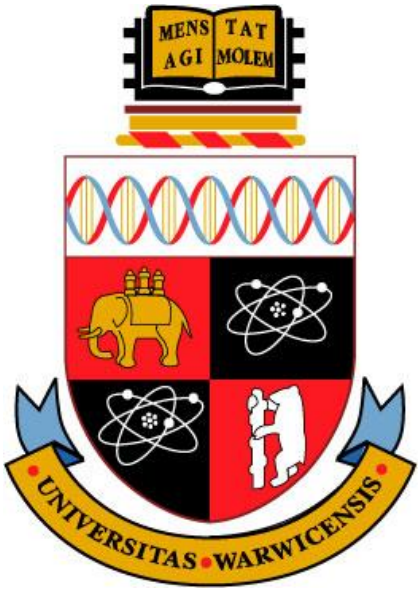

Manual and Automatic Authoring for Adaptive Hypermedia

By

Jonathan Gregory Kenneth Foss

A thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy in Computer Science

Supervisor: Dr A. I. Cristea

University of Warwick, Department of Computer Science

September 2012

THE UNIVERSITY OF WARWICK

Contents

List of Figures

List of Tables

Acknowledgements

Sincere thanks to my family, friends and colleagues who have supported me during my studies. I am particularly grateful for the support and supervision I have received from Dr. Alexandra Cristea.

Declarations

This research was partially supported by the GRAPPLE IST (FP7 STREP) project IST-2007-215434 and the University of Warwick's IATL funded APLIC project.

Some of the material in this thesis is based on papers that I have previously published (as first author, with a contribution of over 80%) as described below.

The introduction presents my own research questions, whilst Chapter 2 describes related research.

Chapter 3 documents the creation of MOT3.0 based on the pre-existing MOT1.0 system (which is described as related research), and is loosely based on the following workshop paper:

Foss, J.G.K. and Cristea, A.I. (2009) Adaptive Hypermedia Content Authoring using MOT3.0. *In: 7th International Workshop on Authoring of Adaptive and Adaptable Hypermedia, 29 Sep - 02 Oct 2009, Nice, France.*

Chapter 4 was created especially for this thesis however it is based on ideas that were published in my other papers. Parts of the evaluation section were previously published in:

Foss, J.G.K., Cristea, A.I. and Hendrix, M. (2010) Continuous Use of Authoring for Adaptive Educational Hypermedia: A Long-term Case Study. In: IEEE International Conference on Advanced Learning Technologies (ICALT 2010), 5-7 Jul 2010, Sousse, Tunisia.

Chapter 5 presents an extended version of the following paper:

Foss, J.G.K. and Cristea, A.I. (2010) The Next Generation Authoring Adaptive Hypermedia: Using and Evaluating the MOT3.0 and PEAL Tools. *In: ACM Conference on Hypertext and Hypermedia (Hypertext 2010), 13-16 Jun 2010, Toronto, Canada.*

Chapter 6 presents an extended version of the following paper:

Foss, J.G.K. and Cristea, A.I. (2010) Transforming a Linear Module into an Adaptive One: Tackling the Challenge. In: 10th IEEE International Conference on Intelligent Tutoring Systems (ITS 2010), 14-18 Jun 2010, Pittsburgh, USA.

Small parts of the description of MOT3.1/PEAL2 in Chapter 7 and small parts of the comparison between MOT3.1/PEAL2 and GAT in Chapter 9 appeared on the poster:

Foss, J.G.K. and Cristea, A.I. (2011) Two competing approaches in Authoring Adaptive Hypermedia: MOT versus GAT. *In: Hypertext 2011, June 2011, Eindhoven, The Netherlands.*

The first part of Chapter 8 is based on related research that was carried out by other members of the GRAPPLE project (as described in the chapter). My improvements to the course tool and shared interface were based on work that was previously carried out by Maurice Hendrix (as described in the chapter). The improvements to the CAM were also documented in the project deliverable:

D3.3c: Final implementation of the CAM definition tool, GRAPPLE project – available from http://www.grapple-project.org/public-files/deliverables/D3.3c-WP3-CAMToolFinal-v1.0.pdf/view

Chapter 11 draws conclusions from the rest of the thesis and has not been published elsewhere.

This thesis has not been previously submitted in any form for a degree at The University of Warwick or any other institution.

Abstract

Adaptive Hypermedia allows online content to be tailored specifically to the needs of the user. This is particularly valuable in educational systems, where a student might benefit from a learning experience which only displays (or recommends) content that they need to know.

Authoring for adaptive systems requires content to be divided into stand-alone fragments which must then be labelled with sufficient pedagogical metadata. Authors must also create a pedagogical strategy that selects the appropriate content depending on (amongst other things) the learner's profile. This authoring process is time-consuming and unfamiliar to most non-technical authors. Therefore, to ensure that students (of all ages, ability level and interests) can benefit from Adaptive Educational Hypermedia, authoring tools need to be usable by a range of educators. The overall aim of this thesis is therefore to identify the ways that this authoring process can be simplified.

The research in this thesis describes the changes that were made to the My Online Teacher (MOT) tool in order to address issues such as functionality and usability. The thesis also describes usability and functionality changes that were made to the GRAPPLE Authoring Tool (GAT), which was developed as part of a European FP7 project. These two tools (which utilise different authoring paradigms) were then used within a usability evaluation, allowing the research to draw a comparison between the two toolsets.

The thesis also describes how educators can reuse their existing non-adaptive (linear) material (such as presentations and Wiki articles) by importing content into an adaptive authoring system.

List of Abbreviations and Acronyms

ADE	Adaptive Delivery Engine
AEH	Adaptive Educational Hypermedia
AH	Adaptive Hypermedia
AHAS	Adaptive Hypermedia Authoring System
AM	Adaptation Model
CAF	Common Adaptation Format
CAM	Concept Adaptation Model
CI	Complexity Imperatives
CRT	Concept Relationship Type
DM	Domain Model
DP	Design Principles
GAT	GRAPPLE Authoring Toolset
GM	Goal Model
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
LAF	Learning Adaptation Format
LAG	Layers of Adaptation Granularity
LMS	Learning Management System
MOOC	Massive Open Online Course
MOT	My Online Teacher
PM	Presentation Model
PRT	Pedagogical Relationship Type

RI	Requirement for Import
SD	Standard Deviation
SI	Support Imperatives
UM	User Model
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VLE	Virtual Learning Environment
WYSIWYG	What You See Is What You Get
WYSIWYM	What You See Is What You Mean

1 Introduction

Adaptive hypermedia [1] (AH) can be used to provide relevant content according to the profile of the user. To achieve this, the content needs to be authored in such a way that sections of it can be shown or hidden according to various conditions. This authoring process is different to the processes that many non-technical content authors are used to, and so has not become widely popularised amongst a diverse range of educators. A lack of content is therefore preventing students from being able to utilise the benefits of adaptive hypermedia [2].

The research in this thesis investigates methods of increasing the usability of editors for adaptive hypermedia, and describes how the editors can be extended to allow more flexibility and functionality, thus encouraging a wider demographic of content authors to create adaptive courses.

1.1 Objectives

The research presented in this thesis aims to create more usable authoring tools for adaptive hypermedia. The thesis focuses on two case study authoring tools (MOT [3] and GAT[4]) that employ different authoring paradigms. Through analysis of the existent authoring tools, the research aims to generate a set of requirements for an *Adaptive Hypermedia Authoring System* (AHAS).

The research also investigates ways of extracting content from files that have been generated from more widely used (non-adaptive) authoring tools.

1.2 Research Questions

The overall research question is:

"*How can the authoring process for adaptive hypermedia be simplified?*"

This question is investigated from two angles

- *Can more graphical user interfaces assist the author with…*
    - *structuring content?*
    - *labelling content?*
    - *defining adaptation strategies?*
- *Can automated importers assist the author?*

Based on these, the following sub-questions were also created:

1. Can visual programming methods assist authors with describing adaptation?
2. What role does the granularity of code fragments have in simplifying the authoring process?
3. Can *visual* methods assist the author in separating content into a useful Domain structure?
    a. How does the ability to visualize many types of relationships (in a graph structure) affect usability?
    b. Are multiple relationship types perceived as useful?
    c. Do users use more (non-parent) relationships when shown in a graph structure?
4. Can *visual* methods assist the author in applying pedagogical metadata?
    a. Are multiple labels/weights useful?

   b. Do authors prefer to separate pedagogical metadata from the adaptation strategy or do they prefer to label content using pedagogical rules?

5. Can content creation for AH be simplified through the use of (semi-) automatic import scripts from more traditional/linear (hyper-) media?
   a. Are such imports semantically/pedagogically valid – or is more complex or human processing required?
   b. Do authors perceive importing content from wikis as useful?
   c. Do authors perceive importing content from presentation slides as useful?
6. How does the granularity of static content affect ease of authoring?
   a. What granularity of static content do authors prefer?
   b. Do current tools facilitate the creation of static content at a granularity that is adequate for both author and learner?
7. Should content be authored with the domain tool or should it use external resources?
   a. Do users prefer editing content within the authoring tool to using their own separate tools, and uploading the content separately?
   b. Do such external resources automatically lead to a larger granularity of content?

1.3 Methodology

This thesis attempts to answer these questions through continual refinement of a number of authoring tools. During the development of the tools, a spiral model [5] of software development was employed. This was an iterative process, allowing for requirements to be gathered from adaptive hypermedia authors. Each prototype piece of software was used in a long-term usage evaluation, during which time the participants were invited to give feedback. New features designed to address the feedback were integrated into the following prototype.

As well as qualitative feedback, each evaluation of the MOT toolset collected usage data. These data were statistically analysed for their complexity, and where appropriate the statistical significance of the data was calculated using a *t-test* in SPSS[6] as described by Field[7], [8].

Some of the evaluations involved gathering feedback through questionnaires. In many cases, the questions involved Likert-type scales [9], [10]. This data was also analysed using a *t-test,* however in the case of categorical data, a χ^2 test, as described by Cairns and Cox [11] was also used.

The research also creates a list of imperatives, which are described in the context of background research, and are later refined based on observations and feedback from evaluations.

1.4 Thesis Outline

The rest of this thesis is structured as follows: Chapter 2 presents the related research and describes the initial set of design principles for an adaptive hypermedia authoring system. Chapter 3 describes how these design principles were used to update the usability and functionality of an existing authoring tool for static content. Chapter 4 describes methods of automatically extracting content from linear material for use within an adaptive course. Chapter 5 describes the creation of a pedagogical strategy tool and presents a long-term usability evaluation of the authoring tools. Chapter 6 describes a scenario that explains how real-world teachers could utilize the authoring tools. Chapter 7 describes the updates that were made to the tools in response to user feedback. Chapter 8

describes a different authoring paradigm, as implemented within a European FP7 project. Chapter 9 describes the differences between these two authoring paradigms, and presents a comparison evaluation. Chapter 10 then describes the creation of an updated authoring tool based on the results of the comparison evaluation. Finally, Chapter 11 presents conclusions and recommendations for future work.

2 Related Research

2.1 Adaptive Educational Hypermedia

Adaptive hypermedia [1], [12], [13] allows content to be personalised according to the needs of the user. One of the primary focuses for adaptive hypermedia is in an educational context [14], [15], allowing content to be personalised to the needs of the learner. For instance if '*Page B*' requires knowledge of '*Page A*' (i.e., '*Page A'* is a prerequisite of *'Page B'*) an adaptive educational hypermedia (AEH) system can provide guidance to the user by recommending that they visit '*Page A*' before visiting '*Page B*' [16]. Similarly, AEH can cater for a variety of learning styles [17–19], allowing different users to receive educational content in a way that they are most likely to respond to.

2.1.1 Techniques within Adaptive Hypermedia

Brusilovsky [12], [13] defined a taxonomy of techniques that can be used within adaptive systems. More recently, this taxonomy has been updated by Knutov et al. [20] to particularly focus on the expansion of the presentation and the content adaptation techniques. The taxonomies describe various techniques within three broad areas:

2.1.2 Adaptive Navigation Techniques

Adaptive navigation techniques involve '*link-level adaptation*' [21], and can guide the user's browsing by recommending pages. This can be achieved by changing the colour of the link [22], [23] or the colour of a bullet point next to the link, such as

the traffic light system used by ELM-ART[24]. Similarly, techniques such as link hiding (making the link look like normal text) can be used [22].

2.1.3 Adaptive Presentation Techniques

Adaptive presentation techniques involve '*content-level adaptation*' [21] and can be used to change the way that content is displayed to the user. Typically this involves either hiding or showing different fragments of content (such as in AHA! [25]), or even modifying the system's entire layout (as demonstrated by GALE [26]).

2.1.4 Content Adaptation Techniques

Whereas Brusilovsky's updated taxonomy [13] considers content adaptation as part of the adaptive presentation techniques, Knutov's taxonomy [20] makes a clearer distinction between the two techniques. Like adaptive presentation, content adaptation is also classed as '*content-level adaptation*', and specifically involves the hiding/removing of fragments (such as that made available by Interbook [27] or AHA! [28]) or the altering of fragments (such as that available in GALE [26]).

2.2 Frameworks for Adaptive Hypermedia

To simplify the design of adaptive hypermedia systems, various frameworks have been developed.

2.2.1 AHAM

The AHAM model [29], [30] extends the Dexter hypermedia model [31] to describe adaptation. The Dexter model contains a 'Runtime layer', a 'Storage layer' and a 'Within Component layer'. AHAM extends this principle through the definition of the following three models.

- The '*domain model*' is used to store fragments of information, and describes how these fragments are structured. Specifically, the domain model contains '*concepts*' (representing the subject), with each concept containing a number of attributes.
- The '*user model*' stores information about the user, by defining an overlay model. This means that the user model can separately address each concept, thereby storing properties that describe the user's knowledge of the concept. Moreover, the user model can store other properties that link the user to the concept, such as a property that denotes the concept has been read.
- The '*adaptation model*' [32] (previously known as the '*teaching model*'[30]) describes how the relationships between the concepts (from the domain model) should be interpreted to form pedagogical relationships that interact with the user model and can therefore be used to guide the user around the course.

2.2.2 Generic Adaptivity Model (GAM)

GAM [33] was created to provide a more generic abstraction of AHAM. Specifically, it removes the dependence on hypermedia and expresses the process of adaptation as a state-machine. The model consists of a '*domain model*', '*adaptation model*' and '*user model*', which are all similar to that used by AHAM. One of the main differences between GAM and AHAM is that since GAM is not specifically used within hypertext, the structure of the domain is not specified in terms of concepts. Moreover, the domain model only specifies the usage of attributes, and leaves the

actual structure of the domain model to the designer of the system. Similarly, GAM introduces an '*interface model*' which defines the possible events that can be triggered by the user, and how such events might affect the adaptation model.

2.2.3 Munich Reference Model

The Munich Reference Model [34] uses UML to specify three separate meta-models; the '*domain meta-model*', the '*adaptation meta-model*', and the '*user meta-model*'. Like AHAM, the Munich model's meta-models all occur within the 'Storage Layer' of the Dexter hypermedia model. Each model performs a similar role to its namesake from AHAM, yet the Munich model's specification focuses on a more formal object-oriented approach.

2.2.4 LAOS Model

The LAOS model [35] is based around the AHAM model, and consists of five separate layers. It is designed to reduce the complexity of the authoring process by allowing the author to focus their attention on the design of each individual layer, rather than being concerned with authoring the entire content and adaptation specification in one process. LAOS therefore adheres to the 'Separation of Concerns' principle, described in Section 2.9.1 .

Figure 2.1 (previously published in [35]) shows the relationship between the five models of the LAOS model.

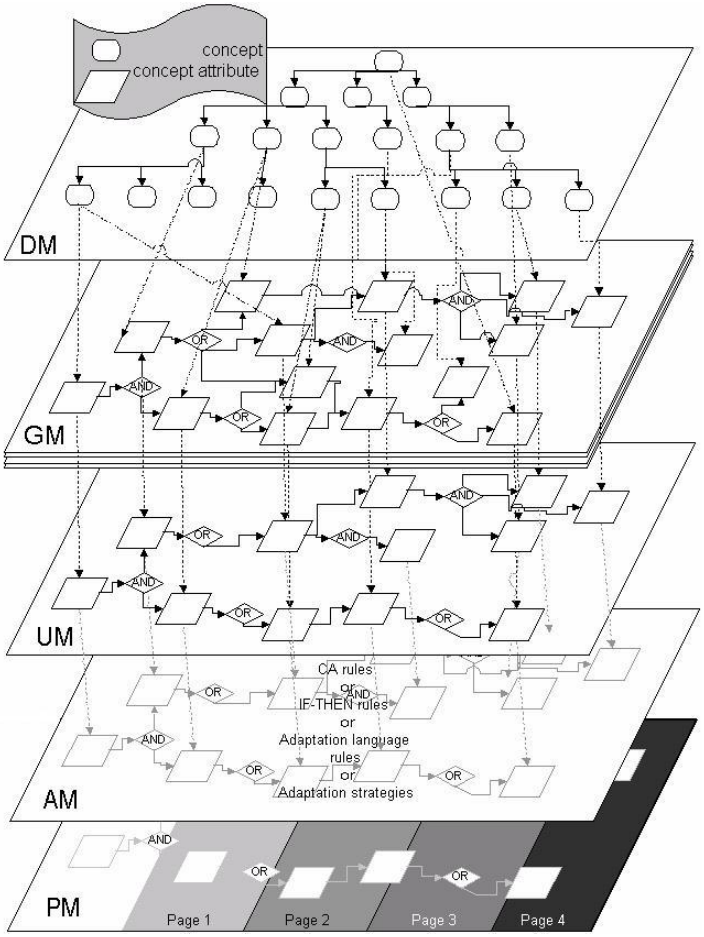

Figure 2.1: The five layer LAOS model

A description of each individual layer is provided below.

- *Domain model (DM)*: Similar to the domain model in AHAM, the Domain layer contains the information content in the form of concepts and attributes. The domain model also defines how the information is structured (but contains no pedagogical information). Although the domain model stores relationships that define the structure of the content, the domain model does not define the order in which the content should be studied (unlike the domain model in AHAM). In the implementations of the LAOS domain model within MOT, an example of a domain model concept would

be a topic such as '*Computer*', whilst an attribute could be '*introduction*' containing some hypertext introducing the topic.

- *Goal and Constraints model (GM)*: The goal model references concept attributes within the domain model so that pedagogical metadata can be defined about each concept. The goal model also stores information about the order and structure of the content. This allows different authors to produce goal maps that use the same domain but order/structure the content in a different way. In the implementations of the LAOS goal model within MOT, authors can add labels and weights to *sublessons* to represent when the corresponding content should be delivered. For instance, the sublesson corresponding to the '*introduction*' domain attribute could be labelled '*beginner*' so that it can be delivered to beginner users (see Section 2.6.2).
- *User model (UM)*: As with AHAM, the user model contains an overlay of the goal model, therefore allowing the storage of information that relates the user to each individual concept. For instance, an implementation of LAOS would be able to store a user's characteristics to label the user as '*beginner*'.
- *Adaptation model (AM)*: As with AHAM, the adaptation model contains the code that determines how the course will adapt to the needs of the user. Within LAOS, the adaptation is usually specified in the form of a LAG strategy[36].
- *Presentation model (PM)*: The presentation model contains information about how the content is delivered, allowing the content to be assembled

into a form that can be interpreted by the browser (usually using HTML). In more recent implementations (described in Chapter 10), the presentation model has also been used to adaptively change the screen layout.

2.2.5 Concept Adaptation Model (CAM)

The Concept Adaptation Model (CAM) [37] was developed as part of the GRAPPLE project [38], and inherits features from both the AHAM and LAOS models. Adaptation is specified using Concept Relationship Types (CRTs). Like both AHAM and LAOS, there is a domain model and a user model. However, rather than specifying adaptation within a single layer, CAM allows adaptation to be spread between multiple layers. The implementation of the CAM within the GRAPPLE project is described in Chapter 8 .

2.3 Authoring Systems for Adaptive Hypermedia

A number of authoring systems have been developed for adaptive hypermedia.

2.3.1 Interbook

Interbook [27] allows content to be authored based on a Microsoft Word [39] file. The domain concepts are automatically identified according to the formatted headings within the document. However, the author must apply a special annotation (a comment in the Word file), to specify related and background concepts (pre-requisites). The document is then converted to an HTML file, whilst the annotations are used to create adaptation rules that are generated using the LISP programming language. The author can also use LISP to specify extra adaptation rules. However, this means that beginner users (non-programmers who cannot use LISP) may have difficulty creating complex adaptation.

2.3.2 AHA!

AHA! [23], [32] follows the AHAM [29], [30] adaptation framework, and allows adaptation to be defined using a graphical tool. Content comes in the form of (X)HTML pages and fragments which can be created using any (X)HTML authoring tool. Content is associated with concepts, which are connected through concept relationships (such as prerequisites) using the '*Graph author*' tool. In addition to this, using special AHA!-specific tags, it is possible to conditionally include/exclude fragments of (X)HTML content [40], [41]. However, adding such AHA!-specific tags would necessitate the mixing of adaptation rules with the domain content, so this is not recommended.

2.4 GRAPPLE

The GRAPPLE [42] project created a delivery tool, GALE[26], and an authoring tool called the GRAPPLE Authoring Toolset (GAT). GAT exports files using the CAM format [43] (see Section 2.2.5), and as such contains three separate tools; the Domain Model tool [44], the Pedagogical Relationship Type tool (also known as the Concept Relationship Type tool) [45] and the Course tool (also known as the CAM tool)[46]. The design and implementation of GAT is detailed in Chapter 8.

2.5 My Online Teacher

This section describes the basic authoring paradigm employed by the My Online Teacher tool [47].

2.5.1 My Online Teacher – MOT1.0 (MOT++)

Unlike Interbook [27] or AHA! [48], which are designed for both delivery and authoring, *MOT++* [47], [49] is a purpose-built authoring tool[1] for use with a separate delivery system and is based around the LAOS framework [36]. Please note that although many previous publications refer to MOT++, it has since been renamed to MOT1.0 for ease of comparison to other tools. The XML-based CAF file format [3] has been defined to store the first two layers of the LAOS framework, the domain model and the goal model. This enables interoperability with compatible delivery engines, such as AHA!, ADE [50], or GALE, the GRAPPLE FP7 STREP project's adaptation engine [26]. This section explains the basic functionality made available by MOT1.0, however actual implementation and usability specifics are described in Chapter 3 .

2.5.1.1 Domain Maps

Domain maps form a hierarchical tree structure of concepts. Each concept must have a title attribute and can have any number of other attributes that describe the concept. Each attribute has a type such as '*text*', '*keywords*', '*image*', '*video*', '*introduction*' or '*conclusion*'. Each attribute can contain the static domain content (either HTML or plaintext). MOT1.0 provides a basic interface to allow authors to author the content of each attribute (Chapter 3 explains the interface in detail).

MOT1.0 also allows authors to copy concepts between domain maps, allowing authors to reuse domain content (either their own, or those made by other users).

[1] http://prolearn.dcs.warwick.ac.uk/MOT/

It is important that the content of each attribute is *standalone* – that is to say that the content does not specifically require the presence of another attribute. For instance, if a concept is described using two attributes '*diagram*' and '*text*' the text attribute should not contain phrases such as "as can be seen in the diagram", since (depending on the adaptation strategy) it is not possible to guarantee that the user can see the diagram.

Similarly, the author must consider the *granularity* of the content [51]. A resource with coarse-grained (also known as 'large' or 'low') granularity has a large amount of content that is stored within a single object/page/concept [52]. However, resources with fine-grained (also known as 'small' or 'high') granularity divide the content into smaller chunks. Using resources with fine-grained granularity means that if the content is reused in other courses the author has greater freedom in choosing which parts of the content to reuse. Similarly, coarse-grained content offers less flexibility in terms of personalisation since the learning object may contain information that is irrelevant to the user but cannot be specifically removed by the strategy. The issues surrounding granularity are also investigated in Chapter 9 .

2.5.1.2 Goal Maps

A goal map is a hierarchical tree structure of groups of '*sublessons*'. Usually, goal maps are created by using MOT1.0's feature for converting domain maps into goal maps. Goal maps initially follow the structure of the associated domain map, with each attribute represented by an individual sublesson. Figure 2.2 shows how

sublessons are arranged into groups based on the domain concept to which they belong.

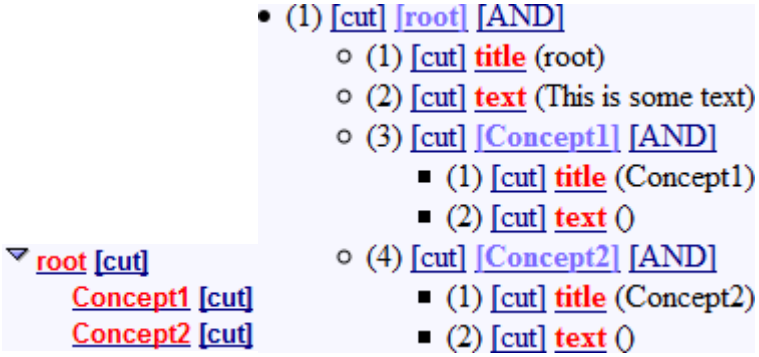

Figure 2.2: A basic domain and equivalent goal map within MOT1.0

MOT1.0 allows authors to rearrange the structure/order of concepts. Goal maps can contain links to any number of domain maps. This means that a course author can reuse domain content in a different goal map.

Authors can add a label and a weight to each sublesson, for use as pedagogical metadata. There are no built-in keywords to define the meaning of weights and labels. The meaning of the weights and labels depends purely on the adaptation strategy.

2.5.1.3 Adaptation Strategy

Within the MOT authoring paradigm, adaptation must be authored within a LAG file [53]. Most strategies make use of the pedagogical metadata defined by the goal map, or even the attribute types used within the domain. The LAG file is responsible for the definition of the Adaptation Model, User Model and Presentation Model. Section 2.6 describes some of the strategies that can be defined.

2.5.1.4 Delivery

As described above, MOT1.0 is specifically designed as an authoring tool, rather than a combined authoring and delivery system. For this reason, the formats generated by the MOT authoring paradigm need to be converted into a format that can be delivered by an external tool. Previous research has created a method of importing content from MOT into WHURLE[54], [55], however the main target for MOT1.0 was AHA![3], [56]. However, subsequent research (described in Chapter 7) created ADE [50], which is a delivery engine that was specifically created to work with CAF and LAG files.

2.5.2 MOT2.0

MOT2.0 [57], [58] was created to research the potential of Web 2.0 style interaction within e-learning. Rather than utilising adaptation strategies, it encourages learners to interact with each other through collaborative tools, such as online chat messaging, tagging and rating. MOT2.0 is therefore not a direct upgrade of MOT1.0, however it does inherit some of its domain model and goal model principles from LAOS[36] to form Social LAOS (SLAOS) [59].

Although MOT2.0 had different functionality to that of MOT1.0, some of the interface and functionality aspects of MOT2.0 could clearly be brought into the MOT1.0 authoring paradigm. Specifically, MOT2.0 introduced a WYSIWYG HTML editor, and a graphical way of displaying the hierarchical structure of a course. Similarly, MOT2.0 introduced methods of importing content from widely used e-learning formats [60]. These features are further described (and used) in Chapter 3.

2.6 Adaptation Strategies in LAG

As described above, the MOT authoring paradigm uses LAG for the definition of adaptation strategies [53], [61]. This section describes the basic principles of LAG, and defines a few sample strategies that will be used as examples throughout this thesis.

2.6.1 LAG Syntax

There are two sections to each LAG file, an *initialization* section, and an *implementation* section.

- *Initialization*: Code within the initialization section is only run when the learner first registers onto the course (i.e., first visits the course). Typically, the initialization code resets the user model variables, and hides all initially unsuitable concepts (via '`PM.GM.Concept.show = false`').
- *Implementation*: The implementation section is run for every concept every time the user clicks on a concept. Note that the expression `GM.Concept` within LAG is actually used to refer to an individual sublesson rather than an entire domain concept. Although the implementation is run for every concept, the expression "`UM.GM.Concept.access == true`" can be used to address the concept(s) that are currently being viewed.

2.6.2 Example Strategies

This section describes some of the basic strategies that are often used to demonstrate the functionality of MOT and AHA!. The example strategies provided

on the website [62] also contain comments that provide natural language descriptions of what the strategy does and the required labels/weights/structure.

2.6.2.1 Beginner-Intermediate-Advanced

The beginner-intermediate-advanced strategy has three separate knowledge variables; *beg*, *int* and *adv*, with the current value stored in the user model variable `UM.GM.knowlvl`. When users first register onto the course, they are classified as a beginner user, meaning that all sublessons that have been labelled with *int* or *adv* will be hidden from the user. In AHA!, this means that the links to the intermediate and advanced topics will be black (looking like normal text – link hiding) with a red bullet point. In ADE, such intermediate/advanced concepts will not be offered on the navigation menu at all.

The initialization part of the strategy looks at all concepts and counts how many sublessons have each label and increments the user model variable counters `UM.GM.begnum`, `UM.GM.intnum` and `UM.GM.advnum` accordingly. When the user views one of these sublessons, the relevant counter will decrement. When a beginner user's `UM.GM.begnum` counter reaches zero (meaning they've visited every beginner sublesson), their level will be changed to *int*, and they will be granted access to the intermediate concepts. Similarly, when an intermediate user's `UM.GM.intnum` counter reaches zero, they will be promoted to an advanced user.

2.6.2.2 Rollout

The goal map for a rollout strategy involves two label types; *showatmost* and *showafter*. The weight associated with each of these labels is also used. The

strategy counts the number of times each sublesson has been shown. If a sublesson has a label *showatmost* with a weight of w_1 (where w_1 is a number greater than zero) it will only be shown the first *w* times that the concept is requested – after which time it will be hidden. Similarly, a sublesson that is labelled with *showafter* with a weight of w_2 (where w_2 is a number greater than zero) will only be shown after the concept has been requested *w2* times.

2.6.2.3 Visual-Verbal Strategy

This strategy uses the label type '*visverb*' (anything that doesn't have this label is shown under all circumstances). The weight belonging to the visverb label is then used to determine how visual or verbal a sublesson is. Verbal sublessons (e.g. long pieces of text or audio clips) are labelled 'visverb' with a weight between 50 and 100. Visual sublessons (images, videos, etc.) have a weight between 0 and 50. When the user first registers, the initialization assumes the user is neutral, so sets their `UM.GM.visverb` preference to 50. Users can then only see sublessons that have a label that is within a predefined threshold of their visverb preference. For example, if the threshold was 30, a neutral user would see all concepts that have weights between 20 and 80. If the user clicks on a sublesson that is labelled with '*setvisverb*', the strategy will set the user's `UM.GM.visverb` preference to the weight of that sublesson. This allows the user to manually choose whether they prefer visual or verbal information; however variations on this strategy allow the strategy to gradually modify the user's visverb preference according to their behaviour within the system.

2.6.2.4 Multimedia Strategy

A multimedia strategy is similar to the visual-verbal strategy; however it can be used to specify a variety of different media (such as text, images, audio, or video). If a concept is represented by a variety of different media styles, the strategy can deliver the appropriate content based on the user's preference, past behaviour or available bandwidth [63]. Rather than using a single label (like the visual-verbal strategy uses), a simple implementation could achieve this using a variety of categorical labels (such as '*text*', '*image*', '*audio*' or '*video*'). Moreover, a '*type-based*' strategy could be used, where instead of relying on the goal model labels the strategy would simply choose the media according to the attribute type.

2.6.3 Reusing Strategies

Keeping all adaptation specification in a separate file (LAG) to that of the static content information (CAF) allows for greater flexibility in reuse. Specifically, this separation is designed so that non-programmers can choose a strategy, read its description, and then use the MOT tool to edit/label their content in such a way that it can be used with the adaptation strategy.

2.6.4 Other 'Custom' Strategies

Many of the above strategies are intentionally generic enough to be applied to a wide variety of content. However, the LAG language aims to provide authors with the flexibility to present their content according to whatever pedagogical strategy they feel appropriate. Indeed, some of these strategies may be domain specific, or contain parameters that must be defined within the LAG strategy.

- *Revision strategies*: A number of authors have implemented strategies that show the '*summary*' of a concept if the main information in the concept has already been read. The theory behind this type of strategy is that if the learner has already read the concept, they only need basic information (such as bullet points, or an image) in order to remember the content. This type of strategy has been implemented in a variety of different ways, with different weights and algorithms to determine when a concept can be considered as 'read'.
- *Quiz strategies*: Various quiz strategies have previously been implemented. A basic quiz could be based around a Type-based strategy, which would only show the sublessons of type '*answer*' when the corresponding '*question´* had been read. However, it is also possible to create a strategy that requires users to choose the answer to a multiple choice question (by clicking on a corresponding concept). If the user clicks on the correct concept, the strategy could increment a `UM.GM.score` variable.
- *Combined strategies*: Many previous authors have combined some of the more common strategies in order to create new strategies. For instance, a Visual-Verbal strategy could be combined with a Beginner-Intermediate-Advanced strategy to ensure that the appropriate type of content is delivered (an 'Adaptive Presentation' technique) whilst providing the learner with guidance about what to study next (an 'Adaptive Navigation' technique).

2.7 Content Authoring Systems

The research presented in this thesis attempts to simplify the process of authoring adaptive hypermedia. Ideally, a large variety of content authors would be able to create their own content in such a way that it could be used in an adaptive system. This would greatly increase the amount of educational material that could be personalised to the needs of the learner. The lack of tools that can be used by an averagely computer literate content author is therefore a major bottleneck to domain authoring [64] and therefore the widespread usage of adaptive hypermedia.

However, there are some non-adaptive tools that are widely used by a wide variety of educators. For instance, many educational institutions use learning management systems (LMSs) such as Moodle[65] or Sakai [66]. Using such systems, educators are able to structure an online course through the creation of web pages and the uploading of lecture slides.

The success of projects such as Coursera[2] demonstrate a growing importance on e-Learning [67]. It is therefore important that e-learning tools can be populated with content from a wide variety of content authors. Although educators may now be familiar with the process of authoring for such linear authoring tools, the process of authoring for adaptive hypermedia requires content to be structured and labelled for use within some form of adaptation (or pedagogical strategy).

[2] https://www.coursera.org/

2.8 Visualizations

2.8.1 Visualization of domain content

As previously described, the process of authoring static content for use within an adaptive hypermedia system is rather different to the process of authoring linear content. The author must think carefully about how their domain content is structured, and what relationships exist between the individual concepts. For this reason, the authoring tool must allow the user to create and view relationships between different pieces of content (concepts).

As described in Section 2.5.1.1 , MOT1.0's approach to this is to display content in a hierarchical tree structure. However, this means that only parent relationships can be clearly displayed to the author. The GRAPPLE project, however, displays content in the form of a graph structure [44], which allows more relationship types to be visualized. A full comparison and evaluation between the two implementations is provided in Chapter 9 .

A similar comparison between graph-based and tree-based visualization of domain maps was carried out by Freire and Rodriguez [68] using the WOTAN tool. Their evaluation involved a small number of participants, with each participant using both a tree design and a graph design over a short time period. They focussed on the time each participant took to complete a number of certain tasks, and found that participants were faster using their second tool rather than the first – i.e., the actual representation appeared to have little effect. However, they did conclude

that the graph tool is "*perceived as harder to use and takes longer to get used to than the tree tool, but seems to provide better understanding of the course layout*".

2.8.2 Visualization of adaptation content

Whilst the above section describes how static content could be visualized, adaptive hypermedia also requires some form of adaptation strategy to be created. As described in Section 2.6 , the MOT authoring paradigm allows non-technical authors to reuse LAG strategies that have been created by programmers. However, in order to provide flexibility, it is important that non-programmers should also be able to specify their own adaptation.

Most current adaptation specifications involve the pedagogy specialist writing code. For this reason, this thesis investigates methods of visualizing an adaptation strategy, and therefore enabling it to be understood and modified by a non-programmer.

This clearly has parallels with tools that allow non-programmers to design their own software. For example, Scratch [69] is a tool that is aimed at introducing children to program. This is achieved by using small fragments of code which are represented as 'building blocks' that can be simply pieced together to form a basic program. Similarly, App Inventor [70] can be used to program basic smartphone apps. Although neither of these tools provides access to a full programming language, they do allow non-programmers to specify their own code. Similarly, Microsoft's on{X} project [71] aims to allow programmers to share '*recipes*' for smartphones

with smartphones, allowing non-programmers to visually change parameters of the recipe for their own personal use.

2.9 Design Principles for Adaptive Hypermedia Authoring Systems

2.9.1 Separation of concerns

The separation of concerns principle is designed to ensure reusability and flexibility. The principle ensures that (e.g.) adaptation specification is separated from the domain content. The layered structure of the AHAM model (and therefore AHA!), is designed to keep the domain, adaptation and user models separate from each other. Wu et al. [29] noted the need within AHAM for separating the adaptation model from the domain model, however adaptation content was also often stored within the static content. For this reason, the goal and constraints layer added to LAOS [35] was designed to ensure that only the factual domain content is stored in the domain model, and pedagogical information is shifted to the goal model. The separation of concerns is inherited from object-oriented programming [72]. It is also closely related to the '*separation of content from presentation'* principle that is employed with XHTML2.0 and CSS [73], which allows authors to change the presentation of a number of web pages by simply redefining the CSS files – thus promoting reuse.

2.9.2 Requirements for an Adaptive Hypermedia Authoring System

Based on this related research, we can identify an initial set of principles that are necessary for an adaptive hypermedia authoring system.

DP1. *Use of Frameworks*. The frameworks described in Section 2.2 demonstrate how the frameworks already implement the separation of concerns principle to some extent. For this reason, it is recommended that any new adaptive (delivery or authoring) system follows a framework. Broadly, the frameworks all specify that there are two different types of content that need to be specified; static content, and adaptive content.

a. *An AHAS should adhere to a Static Content Specification.* As the above background research has demonstrated, most adaptive hypermedia systems use HTML. However, there are no widely-accepted standards for describing the static content used within adaptive hypermedia. There are, however, many standards (such as IMS-CP, IMS-QTI, IMS-LD[3] and SCORM[4]) that are widely used within e-learning systems. In the absence of any static content standards for adaptation, Ghali [60] created methods of allowing such e-learning standards to be utilised within an adaptive course. Moreover, Chapter 4 further explores standards and formats that are frequently used for creating educational content.

b. *An AHAS should adhere to an Adaptive Content Specfication*. Some previous systems (such as Interbook [27] and early versions of AHA! [74]) combined the static content with the adaptation specification. Using some form of specification (such as LAG, or potentially a more

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

[4] http://www.adlnet.gov/Technologies/scorm/

generic scripting language) the adaptive content can more easily be separated from the content, and thus be reused. In this context, we consider all user model and presentation model issues to be directly related to the adaptation – and therefore also covered by this principle.

DP2. *An AHAS should support reuse.* To remove the burden of authoring new content, it is important that the authoring system allows the author to reuse existing content. This requirement was used by Cristea et al. when comparing frameworks in [75] and can be divided into two sub-requirements.

a. *An AHAS should allow linear content import.* As Section 2.7 described, many authors will already have prepared linear resources in various kinds of media and file formats for use within other environments – either real-world universities (such as presentation slides) or in virtual learning environments (such as course pages authored in Moodle or Sakai[66]). The adaptive hypermedia authoring system should accommodate file formats that have already been created for other purposes and, where possible, enhance the value of these files via adaptation. Ghali [60] previously researched integration with learning management systems and Chapter 4 details methods of extracting content from other linear sources.

b. *An AHAS should support the author with reuse*. Within an adaptive system, it is likely that authors will want to reuse educational content that has already been created. The system should provide a way of helping the user to maintain and reuse currently stored content (and searching through it). This could be content that they themselves had previously created, or the author may wish to use content that has been created by another author, thereby allowing a collaborative process such as that used by Baloian et al. [76]. Similarly, it is also important that reuse also applies to the adaptation strategy, thereby allowing non-technical users to take advantage of the code that has been implemented by programmers.

DP3. *An AHAS should ensure an acceptable level of usability*. To encourage non-technical content authors to use the AHAS, the tool must be usable. This principle can be divided into two main sections.

a. *An AHAS should be consistent with other applications*. The authoring system should provide consistency [77] with other well-known applications, especially functions that are common to more generic authoring systems should be easily recognizable to beginner users.

b. *An AHAS must have a shallow learning curve*. The application should also allow users of all abilities to create complex adaptive content that utilizes all possible benefits of adaptive hypermedia. Such functionality will need to be presented simply to the user, requiring a shallow learning curve.

2.9.3 Description of existing tools with respect to these principles

In this section, three adaptive hypermedia authoring tools are described with respect to the above requirements.

Interbook does not use a particular framework (DP1), and therefore has no way of separating static content from the adaptation rules (DP1.a and DP1.b). There is scope for limited adaption reuse in Interbook, through the use of LISP. Interbook inherits many of its user interface features from Microsoft Word. This means authors can import existing Word documents into the system to use as the basis for their material, and also reuse any documents prepared for adaptive hypermedia (DP2.a and DP2.b). Although much of the interface is Microsoft Word (DP3.a), the author still requires understanding of how to apply adaptation rules (DP3.b).

AHA! uses the AHAM [30] framework, but also allows mixing adaptation rules with static content (DP1.a and DP1.b). Content must be authored in (X)HTML, for which many authoring tools exist (DP2.a and DP2.b). AHA! comes with a tool to import content (to upload it to an AHA! server). Adaptation in AHA! can be defined graphically, without any knowledge of the adaptation rule language (DP3.a and DP3.b). However, authors do need knowledge of the adaptation rule language in order to define new complex adaptation rules.

Due to the combination of the LAOS framework, the HTML content, and the ability to export to CAF, MOT1.0 fulfils requirements DP1, DP1.a and DP1.b. In terms of content reuse, HTML can be used as a basis for static content (DP2.a). MOT1.0 also allows static content to be copied and linked to other static content – this makes

reusing information from other domain maps, possible (DP2.b). MOT is a web-based tool, and therefore inherits many user control paradigms from general web design principles. For example, hierarchies in MOT1.0 are rendered using static HTML. However, the interface used in MOT1.0 is inconsistent with other applications, therefore not completely fulfilling requirement DP3.a. Previous evaluations highlighted usability issues within MOT [46], [78] (such as too much complexity in the process of labelling content, and the hierarchy display), therefore highlighting problems with the interface, DP3.b. These usability issues are discussed and addressed in Chapter 3.

Based on these requirements, MOT1.0 demonstrates the most flexibility for authoring. However, the requirements also highlight areas that could be improved within MOT.

2.10 Conclusion

The principles defined in Section 2.9 are the minimum requirements, which have been extracted based on the related research presented in this chapter.

Since authoring for adaptive hypermedia is a complex task, it is vital to have greater acceptance of the tools we propose to the users. The creation of such intuitive authoring tools is therefore an important research challenge. The next chapter details the basic usability and functionality issues that were identified in MOT1.0, and describes the steps that were taken to improve them.

3 Analysing and improving basic Adaptive Hypermedia Authoring: Creating MOT3.0

3.1 Overview

This chapter describes the initial usability issues that existed in MOT1.0 and describes the steps that were taken to create a new version of My Online Teacher named MOT3.0.

The chapter also describes an evaluation that was carried out to compare the usability (and functionality) of MOT3.0 to that of MOT1.0.

3.2 Requirements for Improving MOT1.0

The previous chapter described the motivation for improving the authoring tools. The first step to improving the authoring tools was to examine the feedback from previous evaluations[78], and to make observations about the tools compared with similar authoring tools. The results of this analysis led to the following issues with MOT1.0 being identified.

Issue 1. The usability of MOT1.0 needed to be increased.

a. Visualization was too basic and not always intuitive.

b. The system did not assist the author with creating formatted static content – i.e., the attribute editor only allowed plaintext (or hand-coded HTML) content.

c. The layout of the interface was not appropriate in some cases – specifically common features were hidden away in sub-menus.

d. The *look-and-feel* of the interface was somewhat out-dated.

e. There were inconsistencies in the interface (e.g., some menu items had different positions, depending on which page the user was on).
f. There were few usability conventions/affordances[79] employed.
g. The existent functionality was not always obvious.
h. The search facilities were very minimal.
i. There were a number of security flaws in MOT1.0.

Issue 2. The functionality of MOT1.0 needed to be extended.

a. The number of weights and labels for pedagogical metadata annotations was limited. This limited the opportunities for adaptation strategies.
b. The interoperability with other formats was limited – some limited integration with the semantic desktop had already been implemented [80], however there was no method of automatically populating content with teaching materials that originated from other sources.

Issue 3. The complexity of MOT1.0 needed to be addressed.

a. MOT1.0 was limited with respect to standards that could be used. Specifically, only HTML was supported as a presentation format, with no support for commonly used e-learning standards.
b. It was not clear whether the separation of concerns (defined by the LAOS model) was understood by and useful to the different types of authors.

To address the limitations found in MOT1.0, MOT3.0 was developed. The existing Perl code was hard-coded to the needs of the MOT1.0 interface. It was therefore decided to design MOT3.0 as a complete rewrite of the previous system using PHP. This was because PHP provides a wide variety of modules that are readily available on many server installations and is directly compatible with many existing web-based tools. These modules provided simple integration with external tools such as MySQL, and allowed for the more advanced client-side scripts (AJAX/JavaScript) to be easily generated.

The basic principle of MOT was retained, keeping desired functionality and improving on some of it, as well as adding new required functionality.

3.3 Usability Improvements

3.3.1 Layout

One of the most immediately obvious usability issues with MOT1.0 is its frames-based layout (Issue 1.d – see Figure 3.1). The frames-based layout was originally designed to allow concepts to be shown alongside the current domain map without needing to refresh the entire page (and thus regenerate the entire domain map).

There are two main reasons why a page refresh is undesirable in this context:

1. It transfers a lot of data, and causes the web server to perform database queries and text processing to generate the data – this presents problems with scalability.

2. The time taken for the network, web server and web browser to transfer, process and render the HTML data can be frustrating, and visually distracting for the user.

Using the frame layout helps to minimize the number of page refreshes, thus reducing some of the problems caused by the data transfer issue. However, the speed and visual distraction effect still occurs when using frames, since the frame on the right still needs to update frequently.

Figure 3.1: The frame-based layout of MOT1.0

To improve the layout, it was decided to use Ajax [81][82], which allows small fragments of the page to be requested and downloaded in the background (asynchronously), and only shown to the user when the information is ready. The updated layout is shown in Figure 3.2.

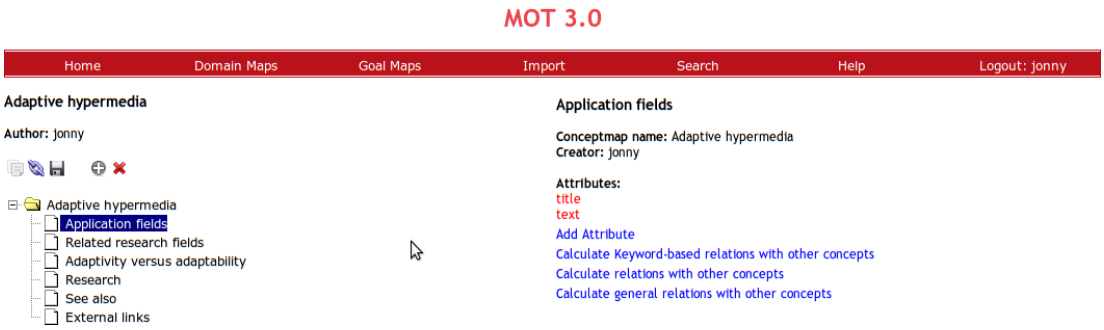

Figure 3.2: The Ajax based layout of MOT3.0

3.3.2 Menus

Another problem arising from the use of frames is the distance between the two menus (Issue 1.c). Figure 3.1 shows that the main Domain map menu is on the far left of the screen, whilst the menu relating to the selected concept is on the left of the right frame. This means that if a user doesn't know where to find a particular menu item, they will have two separate menus to search through. Moreover, some of the items (such as '*Copy concept from other DM*') actually spanned two lines. This makes the item look like two separate items ('Copy concept from' and 'other DM'), which could make it more difficult for the user to find the correct item. There was also an inconsistency with the terminology used by the menu, with the first two items referring to *Domain Maps* and the other items referring to *DMs*. Providing a clearer terminology within the MOT3.0 menu, and making all top-level functions (such as Loading/Creating a Domain or Goal map) available from the main menu was designed to demonstrate the functionality that was available within MOT3.0 (and thus improve the system with respect to Issue 1.g).

Furthermore, in MOT1.0 the options on the menu were hard-coded into each page, and so the menu layout changed slightly on every page (Issue 1.e). For instance, the two menus shown in Figure 3.3 are from two slightly different parts of the domain

tool (list of domains, and the domain editing pages respectively). In this instance, the '*Create Domain Map*' link changes from a button to a textual link. Moreover, the two icons at the top change appearance and functionality between the two pages. In some cases this was an attempt to provide an adaptive menu – however feedback provided during usage had shown that it confused the user, with the lack of a global navigation menu disorientating some users. Similarly, having the "*My On-line Teacher (MOT)*" title at the top of each frame made the two frames look rather separate from each other.

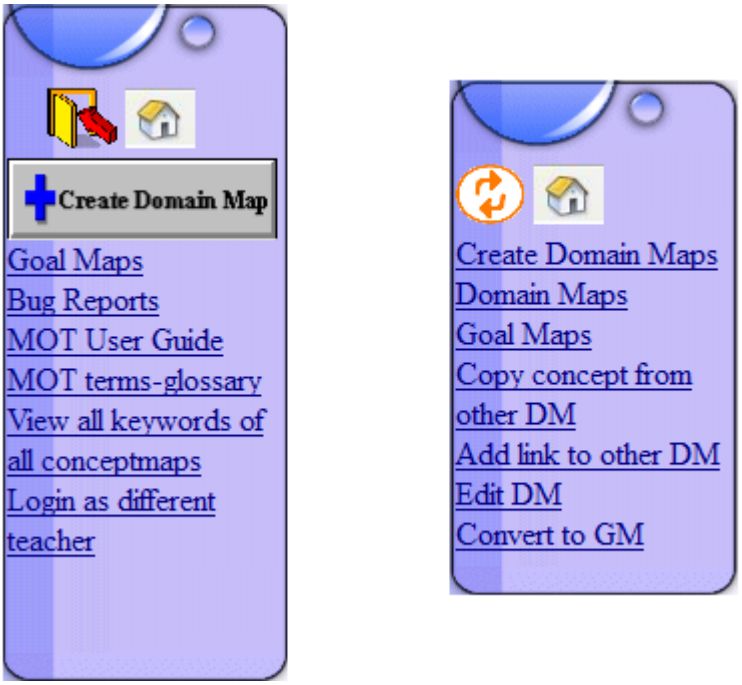

Figure 3.3: The separate navigation menus in MOT1.0

For MOT3.0, it was decided to create the menu in a single PHP source file, which ensures that the menu appears in the same way on every page. This menu was designed to provide a consistent navigation menu that appears on all screens and provides access to all major features of the system, as shown in Figure 3.4.

Figure 3.4: The navigation menu in MOT3.0

The menu provides quick-access to all major areas of MOT3.0, with the structure of the menu designed to help the user understand the structure of the site [83]. Functions that are specific to the current context are kept separate to the main menu and closer to the area of the screen that is affected by the action. These functions are made available through the use of graphical icons.

Figure 3.2 also shows the five icons that appear above the domain map hierarchy; '*Copy Concept*', '*Link to Concept*', '*Save Hierarchy*', '*Add Concept*' and '*Delete Concept*'. Only icons that have readily understandable graphical metaphors [84] were chosen so that they could be immediately recognised by the user. Furthermore, each icon was given an HTML title attribute, so that the user could hover over the icon to discover its meaning. Other (more specialized) functions were kept as text links (such as those relating to calculating relations), since their meaning could not be easily conveyed in an icon.

3.3.3 Visualization

The visualization in MOT also needed to be improved for the display of both Domain maps and Goal maps (Issue 1.a). MOT represents both types of map as a tree structure, but in MOT1.0 this was displayed as static HTML. This meant that every time the user wanted to rearrange a hierarchy, s/he had to click the '*cut*' (textual) link next to the source concept, and then click on the '*paste*' link next to the concept that was to become the parent of the source concept. This caused

many page refreshes, causing similar scalability and time delay issues as described for the frames interface.

Figure 3.5: The two stages of changing a Domain hierarchy in MOT1.0

When cutting and pasting within the same Domain map, the above process all occurs within a single frame.

MOT3.0 uses the JavaScript component dhtmlxTree[5]. A similar JavaScript tree structure had also been implemented by Ghali in MOT2.0[57], but was mainly for the purposes of the presentation of content, rather than for assisting with the authoring process.

[5] http://www.dhtmlx.com/docs/products/dhtmlxTree/index.shtml

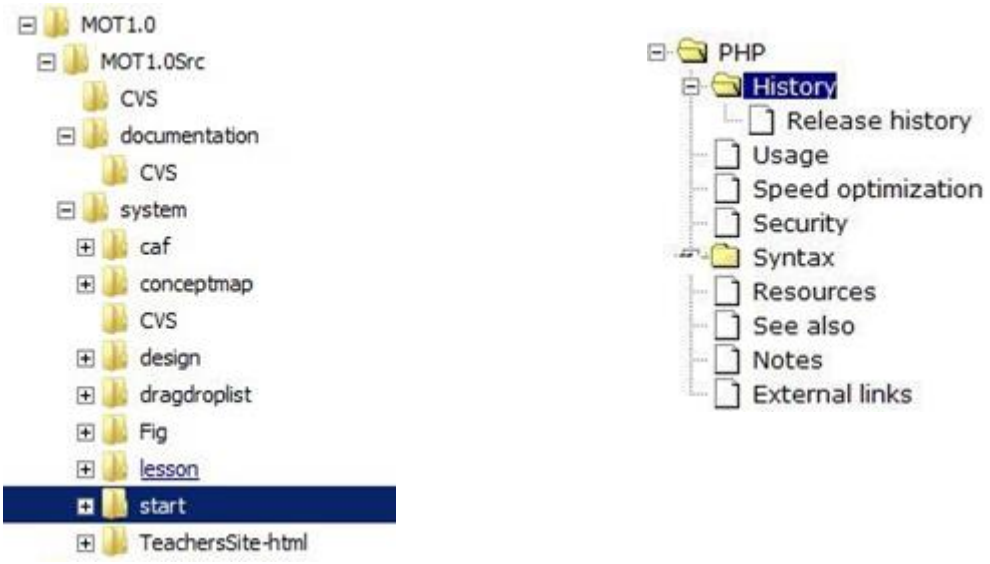

Figure 3.6: The Windows 7 file manager (left) with the MOT3.0 Domain map (right)

This type of hierarchy visualization is more familiar to users, as it is similar to many file manager applications, such as Explorer used in Windows 7[6] (see Figure 3.6), this therefore makes the interface more consistent with other applications, employing more usability conventions and affordances (Issue 1.f).

Nodes (concepts in the case of domain maps, and sub-lessons in the case of goal maps) can be dragged-and-dropped within the hierarchy. As with MOT1.0, MOT3.0 supports the copying/linking of concepts between different domain models. A similar drag-and-drop technique is therefore implemented to allow concepts to be copied between domain maps – see Figure 3.7. Similar rearrangement techniques were created to allow the rearrangement of Goal Maps.

[6] http://windows.microsoft.com/en-GB/windows7/products/home

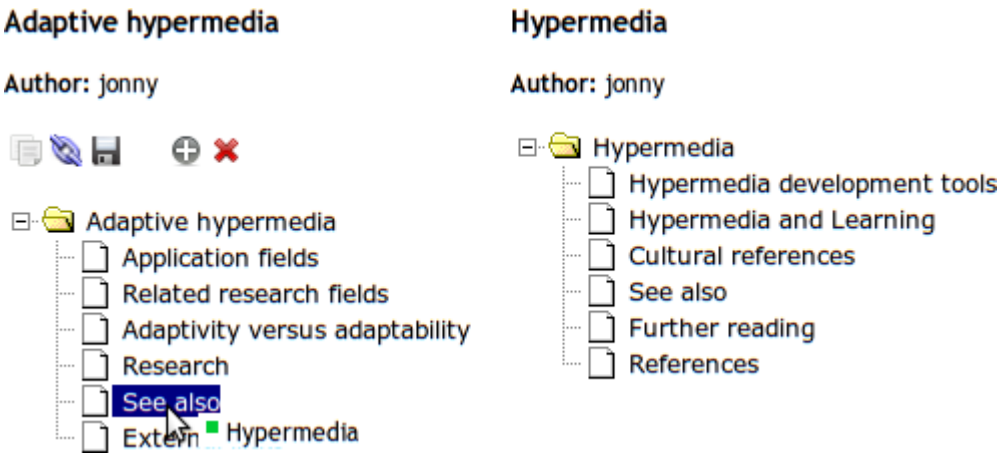

Figure 3.7: Copying the 'Hypermedia' topic into the 'See Also' concept within 'Adaptive hypermedia'

The new hierarchy is sent back to the server using Ajax when the user clicks on the *Save* button. This allows the user to change their mind about the structure, whereas the rearrange methods in MOT1.0 had instantly overwritten the previous structure (with no *Undo* feature).

Another visualization issue that was improved in MOT3.0 is that of visualizing the labels and weights within the hierarchy. Figure 3.8 shows the interface for adding labels and weights within MOT1.0, whereas Figure 3.9 shows the interface for the same functionality within MOT3.0.

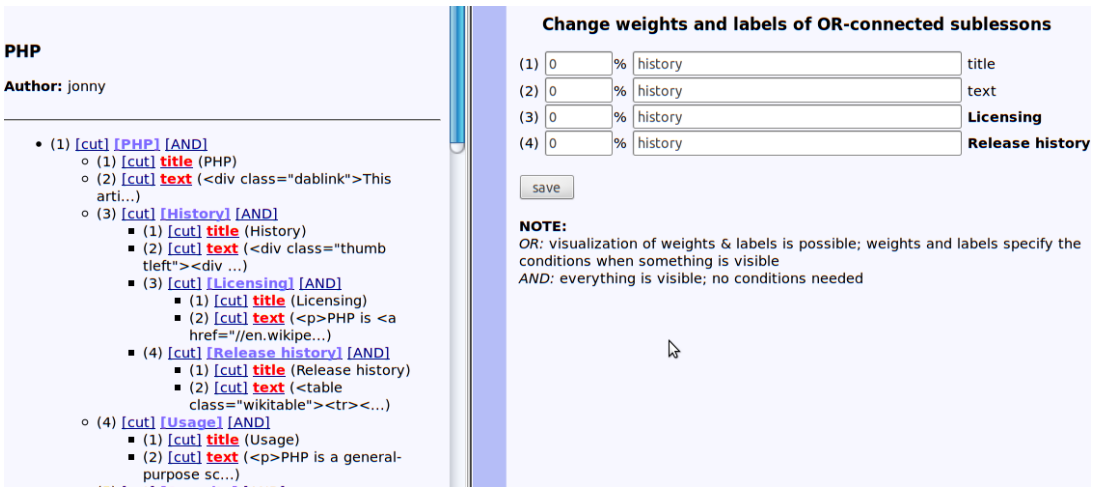

Figure 3.8: Changing weights and labels in MOT1.0

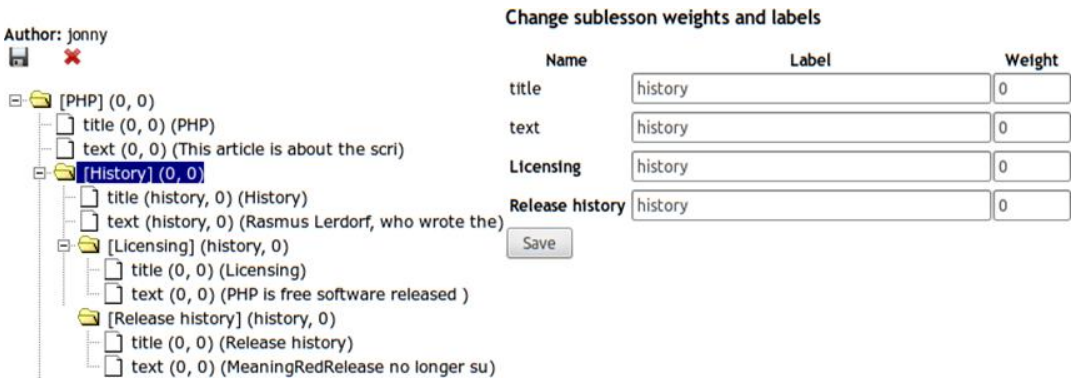

Figure 3.9: Labelling within MOT3.0

Specifically, the MOT3.0 tree allows the labels and weights of sublessons to be easily viewed, whereas in MOT1.0 the author needs to actively select a sublesson and choose to change its labels before being able to discover the current values of the labels and weights. MOT3.0 also removes the AND-OR relations that existed in MOT1.0 (and are described in [35]) since they had no meaning within the implemented LAG language. Similarly, the percentage sign next to the weight textbox was removed because the weight need not be considered as a percentage, and can represent any numerical value depending on the selected strategy.

3.3.3.1 Editor

MOT3.0 adds a WYSIWYG HTML editor (FCKEditor[7]) for editing attributes in the domain model. The editor (shown in Figure 3.10) had been previously used by Ghali in MOT2.0 [85]. As described in Chapter 2, MOT2.0 had different functionality to that of MOT1.0 – MOT3.0 therefore brings this high usability editor into the mainstream adaptive authoring implementation. MOT1.0 had simply provided a plain textbox for users to input either plain text, or hand-coded HTML.

[7] http://www.fckeditor.net/

Adding the WYSIWYG HTML editor ensures all authors are able to create courses with HTML content, thus improving MOT with respect to Issue 1.b**.**

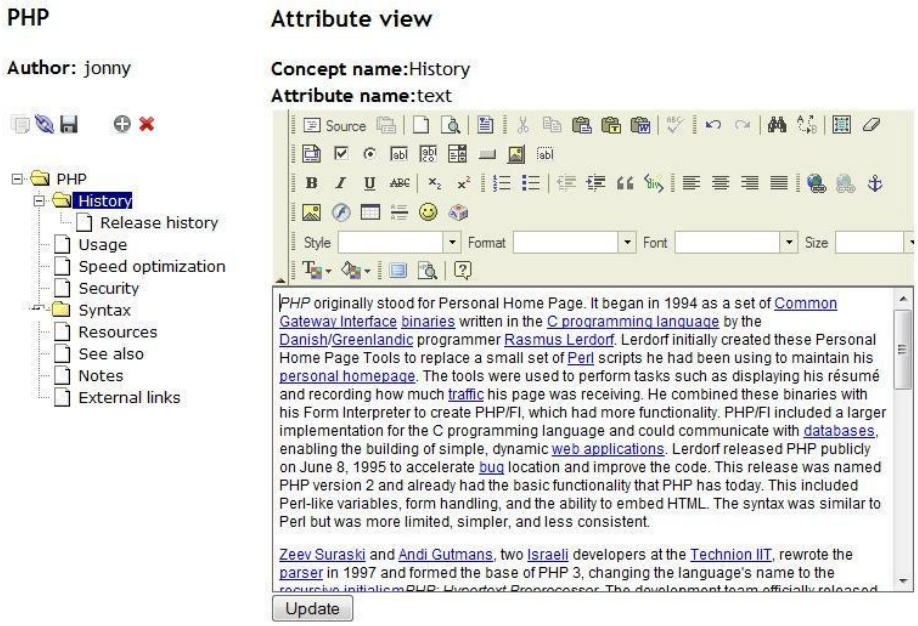

Figure 3.10: Editing an attribute in MOT3.0

3.3.3.2 Search functionality

MOT1.0 contained no built-in search feature, although it did contain a feature called '*View all keywords of all conceptmaps*', which generated a page that contained the contents from every attribute named '*keywords*' and then allowed the user to use their web-browser's '*Find*' feature to look for the appropriate topic. There are four main disadvantages to this.

1. It provides a (non-scalable) CPU overhead for the server.
2. It relies entirely on the author populating the *keywords* attribute.
3. It only considers Domain maps, and not Goal maps.

4. The large list of generated content could be confusing, overwhelming and difficult for the user to interpret.

MOT3.0 provides a search page, which allows users to type a search term, where the user can select any combination of a number of search scopes (Concept Maps, Goal Maps, Keywords and Concept Titles). The result is then displayed to the user using Ajax. This makes searching for particular pieces of content easier, and thus the system more user friendly, and therefore contributes towards Issue 1.h.

3.3.3.3 Security Improvements

Various security features were added to MOT3.0. For example, the *teacherid* variable was passed in plaintext over a GET request [86]. This meant that when a user navigated to a concept map, they would see a URL such as:

```
MOT/conceptmap/concept_frameset.cgi?teacherid=203&mapid=
379
```

However, a malicious user could easily change the teacherid parameter from their own id (which in the above example is 203) to that of another account.

In MOT3.0 the teacherid is stored in a session cookie, and it is therefore harder for the attacker to switch to another user's account, thus improving on Issue 1.i.

3.4 Functionality Improvements

3.4.1 Compatibility with Other Systems and Standards

MOT3.0 retains support for the CAF format. Moreover, MOT3.0 can now also use standard formats for e-learning content, such as IMS-CP[8] and SCORM[9], and the standard format for testing and questionnaires, IMS-QTI[10], through integration of the import scripts that were originally created by Fawaz Ghali for MOT2.0 [58], [60], [87]. The scripts were designed to enhance interoperability with more commonly used (non-adaptive) LMS systems such as Sakai[11] or Moodle[12]. This extension is part of the work in support of Issue 2.a. Further work on this requirement, and work to address Issue 2.b is presented in Chapter 4 .

3.5 Evaluations

To determine whether or not these features have enhanced the functionality and usability of the MOT3.0 system, user testing has been carried out. The first evaluation compared the MOT3.0 system with the MOT1.0 system[13].

---

8 http://www.imsglobal.org/content/packaging/

9 http://www.adlnet.gov/Technologies/scorm/

10 http://www.imsglobal.org/question/

11 http://www.sakaiproject.org/

12 http://www.moodle.org/

13 Please note we don't compare to the MOT2.0 system since it was designed to research Web2.0 principles – which is not the purpose of MOT3.0

3.5.1 Hypotheses for the MOT3.0 comparison to MOT1.0

For the purpose of the evaluation a series of hypotheses were created, as follows.

H3.1 Browsing Domain Maps is easier in MOT3.0.

H3.2 Browsing Goal Maps is easier in MOT3.0.

H3.3 Creating Domain Maps is easier in MOT3.0.

H3.4 It is easier to add and delete concepts in MOT3.0.

H3.5 It is easier to modify the hierarchy of domain and goal maps in MOT3.0.

H3.6 Searching through Domain Maps and Goal Maps is easier in MOT3.0.

H3.7 It is easier to copy domain maps concepts and link goal maps in MOT3.0.

H3.8 It is easier to manually build a goal map by inserting sub-lessons from concept attributes in MOT3.0.

H3.9 The more consistent menu style in MOT3.0 makes it easier to find the correct function (such as converting/exporting maps).

H3.10 The FCKEditor is useful within the context of editing attributes.

3.5.2 Testing the Hypotheses

The first phase of the evaluations was carried out with two groups of students from the Department of Engineering Studies in Foreign Languages, Politehnica University of Bucharest, Romania. One group contained 25 3rd year students registered for the course 'Web Application Development', and the other group had 39 4th year students from the 'Semantic Web' course. Although the MOT3.0 system is primarily

targeted at teachers, it is important that the system is able to appeal to a wide variety of content authors. Due to the web-related nature of their course, they were expected to be proficient at traditional web design/development. It is therefore important to gain the approval of this demographic when developing adaptive hypermedia authoring tools.

Sets of similar[14] scenarios were created for both MOT1.0 and MOT3.0. In order to remove any bias, each group was divided into two subgroups. The first subgroup began the evaluation using MOT1.0, whilst the other began with MOT3.0. When a student had completed each of the scenarios, their work was checked by a lab assistant, and the participant was asked to complete a questionnaire about the functionality and usability of the system. The student was then asked to perform the same tasks using the other system, and then complete a questionnaire. Answering the questionnaire was not obligatory, 13 of the 25 Web Application Development students and 18 of the 39 Semantic Web students responded. Each question in the questionnaire used a very simple four-point Likert scale (Very Easy, Easy, Difficult and Very Difficult). After each question, students were asked to provide comments about their choice of answer. This is due to the fact that non-native English speakers tend to find other scales unclear – e.g., words like “quite” or “cumbersome” that appear on the popular System Usability Scale questionnaire (SUS) [88], are not appropriate for such students. It was also decided to not provide an ‘average’ option, to encourage students to state a preference between the two

[14] Where possible, the scenarios were kept identical. Only where the functionality of MOT1.0 and MOT3.0 differed greatly were there deviations in the scenarios.

systems. To eliminate bias, the questions all started with "*What do you think about…*" (rather than asking directly whether they liked or preferred a particular system):

Q1. … browsing other author's a) domain maps and b) goal maps?

Q2. …creating new a) domain maps and b) adding/deleting concepts?

Q3 …reordering the hierarchies for a) domain maps and b) goal maps?

Q4 …searching for concepts?

Q5 …copying concepts and linking between concept maps?

Q6 …manually creating goal maps?

Q7 …navigating around the menu?

Q8 …using the HTML editor to edit attributes?

Each Likert item was assigned a value (1 represents 'Very Easy', and 4 'Very Difficult'). These values assume a monotone scale, which is a reasonable assumption since the natural language and the presentation of the options within the questionnaire should lead the participants to understand that (e.g.) 'Very Difficult' is the opposite of 'Very Easy'.

To evaluate the differences between the two systems, a two-tailed paired t-test was performed for each question. The t-test assumes equal variances for the results of both systems, and it assumes a normal distribution. For each question, the

MOT3.0 score was subtracted from the MOT1.0 score, with the difference reported in Table 3.1.

Question	***Average Difference***	***Standard Deviation of Difference***	***T-Test Result P(T <= t) two-tail***	***Hypothesis***	***Hypothesis Confirmed?***
Q1a	0.06	0.68	0.6	H3.1	
Q1b	0.16	0.73	0.23	H3.2	
Q2a	0	0.68	1	H3.3	
Q2b	**-0.12**	0.99	0.47	H3.4	
Q3a	0.61	0.88	0	H3.5	**Yes**
Q3b	0.61	0.92	0	H3.5	**Yes**
Q4	0.35	0.91	0.04	H3.6	**Yes**
Q5	0.19	1.14	0.35	H3.7	
Q6	**-0.23**	0.88	0.17	H3.8	
Q7	0.32	0.94	0.07	H3.9	**Yes**

Table 3.1: Comparison of answers to questionnaire about MOT1.0 and MOT3.0

Based on the results of the t-test and applying a 95% confidence level, we can accept hypotheses H3.5 and H3.6. This suggests that it is easier to modify the

hierarchies of domain maps and goal maps in MOT3.0. It is also easier to search for concepts in MOT3.0. The menu layout for MOT3.0 is also preferred, but the hypothesis for this (H3.9) can only be confirmed when applying a 90% confidence level.

There is no statistical evidence to show that MOT3.0 improves on MOT1.0 with respect to browsing Domain/Goal maps (H3.1 and H3.2), creating Domain maps (H3.3), copying/linking between goal maps (H3.7). Moreover, with regard to adding/deleting concepts (H3.4), and manually building goal maps (H3.8), there is a small (but statistically insignificant) reason to suggest that students prefer MOT1.0. For adding and deleting concepts, some users reported JavaScript errors in MOT3.0, and they also felt that the buttons "*were not obvious*" – see Figure 3.11. Similarly, some users stated that the icons did not represent their function very clearly. Some users also reported errors when manually creating goal maps, which seemed to be related to the JavaScript tree display.

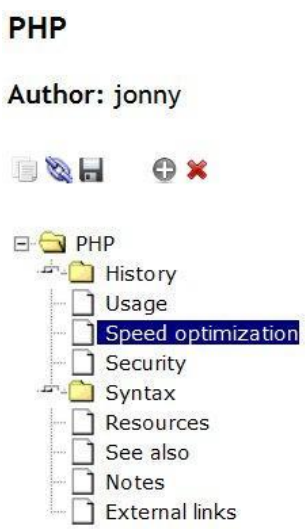

Figure 3.11: The position of the buttons in MOT3.0

The students were also asked for their opinions about the HTML editor (Q8). Since there was no HTML editor in MOT1.0, the students were only asked for their opinions about the editor in MOT3.0, and therefore had nothing to compare it against. However, 20 out of 31 students said they thought the editor was *Easy* to use, a further 9 thought it was *Very Easy* to use, and only 2 said it was *Difficult* (with 0 saying *Very Difficult*).

Many users appreciated the ability to use the FCKEditor editor to create HTML content, but some users said they found the editor "slow to load". These comments provided a good basis for the next iteration of development on the system.

3.6 Updating MOT3.0 based on Evaluation Results

The initial Romanian evaluation provided some valuable feedback, which allowed MOT3.0 to be updated. Firstly, the issues with the JavaScript tree display were corrected by replacing dhtmlXTree with jsTree[15] (as shown in Figure 3.12), which provided more functionality.

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

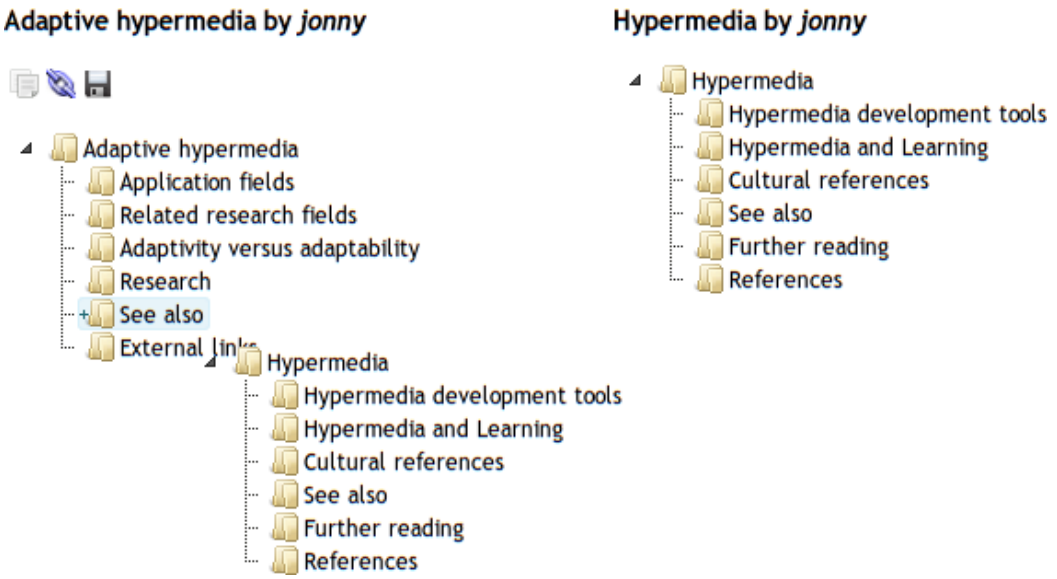

Figure 3.12: Copying a sub-tree of concepts

This allowed a new feature to be added, allowing multiple sublessons to be selected at once (using the *Control* key), thus making it easier to apply the same label to more than one sublesson (see Figure 3.13).

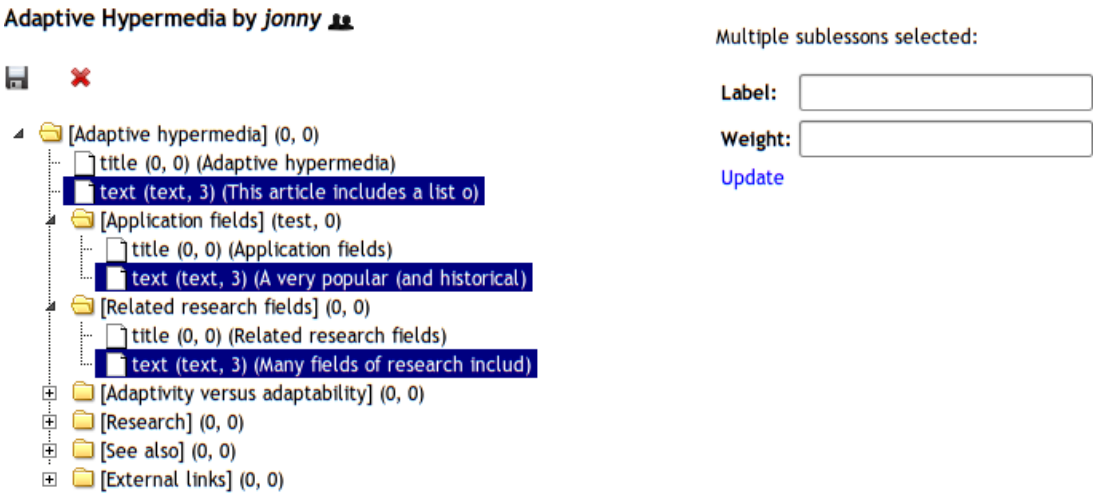

Figure 3.13: Applying the same label to more than one sublesson

Similarly, the issues with the HTML editor were improved by moving to the HTML editor component of the YUI[16] library. These changes were implemented to create MOT3.0.1, which is evaluated in Chapter 5 .

3.7 Conclusion

This chapter has described the initial set of usability and functionality issues that existed in MOT1.0 and then describes how the implementation of MOT3.0 improves on these usability issues.

The chapter also presented the results of a usability evaluation that was performed to assess the usefulness of these modifications. The evaluation results have shown that overall MOT3.0 progressed according to the requirements, as well as identified areas for further development.

This chapter has primarily focussed on the improvements made to the usability of the tool. However, the initial requirements (described in Section 3.2) also identified ways of improving the functionality of MOT that have not been fully explained within this chapter. Similarly, Issue 3 identified issues with the complexity of the authoring process, which will be further investigated in Chapter 5. In particular, the separation of concerns provided by the LAOS model is compared with another model in Chapter 9. The next chapter discusses options for enhancing the interoperability of MOT with other formats (as suggested by Issue 2.b and DP2.a from Chapter 2).

[16] http://yuilibrary.com/

4 Importing Content from Linear Sources

4.1 Overview

This project's overall aim is to simplify the process of authoring adaptive hypermedia. In particular, we focus on the creation of adaptive courses, and so the target users will be experienced teachers. This chapter investigates ways of allowing experienced teachers to reuse existing teaching materials.

4.2 Motivation

Analysing previous systems, as described in Chapter 2 , shows that it is important to be able to import content from sources that authors regularly use, in order for the authoring system to be really useful and relevant for authors (as described in Chapter 3, Issue 2.b).

As described in chapter 2 there are currently no standards for adaptation. However, Ghali created methods of converting from the e-learning standards IMS-CP, IMS-QTI and SCORM to CAF[58], for use within the MOT2.0 system. In order to illustrate Issue 2.b, as described in the previous chapter, these import scripts were integrated into MOT3.0. Since these standards are widely used by popular e-learning systems (in fact, learning management systems – LMSs), such as Moodle [65] and Sakai [66], their usage clearly extends the usability of the authoring system, allowing authors to import material they might have created in the past in different systems, or even material created by other authors. On the other hand, these LMSs do not provide for personalisation and adaptation, so what is imported is only static content.

Similarly, Hendrix et al. [89] described a method of enriching MOT1.0 content with content from the Beagle++ semantic desktop environment[90]. Beagle++ stores semantic metadata (via RDF [91]) about the files on the author's computer. The enricher can therefore assist the author with finding files that are relevant to the domain, and upload the files as alternative resources. The tool can then also utilise the semantic metadata to add extra properties (attributes/sublessons) to the domain and goal models. However, the resource files uploaded by this system were static – i.e., the content within the file was not divided into small fragments for adaptive presentation.

Beside allowing import-export from given e-learning standard content, when designing MOT3.0, we also explored other de facto 'standards' which are widely used, and could possibly enhance an authoring tool. Many content creators will have already written their learning material using other non-adaptive tools. For instance, many lecturers will already have created lecture slides in presentation files, or maybe created learning resources in documents. Below, we explore such de facto 'standards'.

4.2.1 Familiar Authoring Tools

In this section, a list of commonly used authoring formats is presented. It is likely that teachers will have already created content in at least one of these formats.

4.2.1.1 Microsoft Word

Microsoft Word [39] allows authors to create documents. It is likely that many documents (especially formally written documents) make use of the *Styles* feature, which applies a particular formatting to certain elements of the document.

Although such styles are primarily intended to modify the *presentation* of the document, systems such as Interbook [27] have used the style information as a semantic markup to identify elements of the document.

4.2.1.2 Microsoft PowerPoint

Microsoft PowerPoint [92] (and similar programs such as OpenOffice Impress [93]) allow authors to create presentations as a series of slides. Although there are often template layouts for the slides, the author is not constrained into ensuring that their presentation conforms to a particular structure.

4.2.1.3 LaTeX

LaTeX [94] is a markup language, which is widely used within academia for the presentation of scientific documents. LaTeX code is usually written directly – rather than using a WYSIWYG[17] editor. This is intended to allow the author to focus on writing the content, rather than the presentation of the content.

4.2.1.4 Wikipedia

Wikipedia [95] is an online collaborative encyclopedia which uses the MediaWiki software [96]. MediaWiki is a collaborative authoring tool, which provides a simple to use interface for authors. MediaWiki articles are written in WikiText, which provides basic formatting such as hyperlinks, references, tables and headings. MediaWiki (and therefore Wikipedia) also contains an editor for WikiText, called WikiEditor [97]. Rather than being a WYSIWYG editor, WikiEditor automatically inserts markup templates, to assist the user with content editing.

[17] What You See Is What You Get

4.2.1.5 HTML

Hypertext Markup Language (HTML) [98] is the markup language that describes the content of web pages. HTML is ubiquitous on the web, and there are a plethora of WYSIWYG editors designed to help content creators of all skills and abilities. For this reason, there is a wide variety of online learning material authored in HTML.

4.2.1.6 Portable Document Format

Portable Document Format (PDF) [99] is a commonly used file format for documents. Many teaching materials and journal/conference articles are commonly distributed in via PDF. It is not itself an authoring format, so it is likely that the content was authored in one of the above formats.

4.3 Requirements to import linear material into an AHAS

This section describes the type of data that needs to be extracted from the linear material in order for the resulting content to be of use for creating an adaptive course. For this purpose, we have defined a set of principles (or *Requirements for Import*: *RI*), based on the most common implementation types of adaptive hypermedia, as well as the requirements extracted in Section 2.9 . These principles are to be used as a basis for creating adaptive content from many different file formats. The principles should also be generic enough to be used as a basis for importing content into other adaptive authoring systems. As the generic requirements (Section 2.9) define the need of a framework as a basis, we had to opt for a specific framework. Thus this section describes the conversion process in terms of layers within the LAOS framework [35], whilst forming a set of

requirements for such an importer. Whilst the principle remains the same, some variations are to be expected when applying a different framework.

RI1. Domain Layer

As described in Chapter 2 (via DP1), the first stage of the adaptive hypermedia framework is the definition of the static content for the course (or references to it).

RI1.1. Domain Content should be able to match the content of the input

It is vital that the imported content maintains the content of the input. For adaptive presentation (certainly within MOT3.0 and most current adaptive delivery engines such as GALE [26], AHA! [23] and ADE [50]), the static material needs to be converted into HTML. However, it is also theoretically possible that other content types could be adaptively presented as part of an adaptive course. The importer script needs to find a way of interpreting the input format, and, in most cases, this means converting the format to HTML in a reliable way that preserves the information content of the original input.

RI1.2. Content should be able to be automatically separated into several alternatives (ideally, these should be of sufficient granularity, and 'standalone' pieces)

If the first requirement is fulfilled, at least reproduction of the input material is guaranteed. This requirement ensures that some form of adaptation is possible: if the whole content is converted as a single block, and cannot be divided, then no content alternatives will be present for the adaptation strategy, so all end-users will

by default need to see exactly the same content. Hence, this requirement is essential to ensure at least some degree of personalisation.

The exact degree depends on a number of factors, one of them being the granularity of the extracted material. The main challenge here is that the concepts and attributes automatically extracted should be of a sufficient granularity for adaptation [51], as described in Section 2.5.1.1 .

With static material, most authors are comfortable with the idea of dividing a document into sections with headings. Using these headings it is often possible to divide the content into concepts – such as the way that Interbook uses a document's header styles [100]. However, depending on the author, there is no way of guaranteeing that these pieces are of the required granularity (the sections could be quite large, and hence personalisation in the form of showing or hiding a particular section, or replacing it with another, might not make sense).

Moreover, another issue is that the extracted concepts might not be of a 'standalone' type. In many documents, there may be sections that make references to a previous section or to a section that is still to come. In a linear presentation (such as a static website, or document) such linking phrases do not matter. However, if these sections are to be presented in an adaptive manner, their order may depend on personalisation parameters. In such a situation, it is quite possible that a reference of the type 'see previous section where this notion is explained' is not at all relevant for the personalised order of the sections (the section referred to may not have been seen by the user yet, and may never be shown to the user, if so

decided by the adaptation algorithm). In the process of manually authoring for adaptation it is a known error to include embedded, implicit[18] references. However, when importing previously created linear material, such implicit references cannot be disregarded.

RI1.3. Relationships between concepts should be able to match relationships between elements of the input

Usually, static documents, regardless of the input format, do not have any explicit relationships between their elements. On the other hand, adaptive hypermedia content is mostly represented as a map of concepts, linked via various relationships (depending on the complexity and flexibility of the model used).

However, static documents often have one implicit relationship structure: most of them have a chapter-(sub-chapter)-section structuring, which allows for the extraction of a hierarchical relationship structure. From a more technical point of view, depending on the file type of the static document, it is often possible to extract a structure that is based on the importance (size) of the headings – thus creating a domain with a simple logical structure that matches the layout of the input file.

Although the structure of the course in terms of prerequisites is of a pedagogical nature (and is to be maintained at a '*goal*' level in any system based on the LAOS framework), the Domain should define the hierarchical structure of the course. Concretely, in the case of MOT3.0, this means that the Domain should be divided

[18] Such as 'see below'

into a hierarchical structure of concepts, with each concept containing a number of attributes.

RI1.4. The importer should provide a variety of alternative attribute styles

Another important aspect of creating a Domain is the number of different attribute types that can be extracted for each concept. As described in Chapter 2, attributes should be used to describe concepts in a variety of different ways, depending on the adaptation strategy used. For instance, when dealing with a strategy that caters for different learning styles [17], [18], [101], it may be advantageous to provide (for instance) separate text, image and video attributes as alternatives to be displayed for each concept.

To maximise the utility of the static content, therefore, the importer should be able to extract (and identify) different types of content. Some media-based attributes (e.g., *text*, *image, video*, etc.) can be easily identified based on the way they are included in the source document (for instance, via an <img> tag or MIME type). However, other sections – such as *introduction, conclusion* or *exercise* – can have more subtle distinctions, and are therefore more difficult to detect and automatically extract.

RI2. Goal Layer

The LAOS model recommends that the next stage of the authoring process is to create a *Goal Map*. This results in a number of requirements, as below.

RI2.1. The resulting Goal Map should have a pedagogically useful structure

As described in Chapter 2, the Goal Map imposes an order and pedagogical labels onto the Domain content. In its simplest form, the Goal Map can replicate the hierarchy (and order) of the Domain Map – based on the 'parent' relationships. This mimics the behaviour of the 'Convert to GM' function within MOT3.0, when creating non-imported Domain and Goal Maps. The semantics of this is to convert hierarchical relations (which are inherently domain relations) into prerequisites (which are inherently pedagogical relations). In simple terms, this would suggest, in general, reading higher level chapters first, before reading sections. However, the exact pedagogical meaning of such a hierarchical goal-level structure would be determined by the adaptation strategy.

4.4 Recommendations for creating adaptation from linear material

The requirements RI1.1 to RI2.1 are reasonably straightforward to implement, since they only require some simple conversions to be applied to the original input document. However, by definition, the static content has no way of describing how the adaptation should be implemented. Indeed, an important principle of the content importer (and of LAOS in general) is that the static content can be reused by many adaptation strategies. For this reason, the adaptation side of the import process is not as rigidly defined, and may require more *creative* input from the author.

On the other hand, imported static content cannot directly be deployed in adaptive hypermedia, without an adaptation strategy allocated to this content. In its simplest form, a basic strategy that shows everything can be applied. This is of course not adaptive, because it will show all content to all users regardless of their

profile. However, it will at least render the imported content usable, and in fact be no worse than the original content, which was also static and linear. If authors are not satisfied with this simple presentation, they would have to create the adaptation specification manually. Here we introduce requirements that move away from this baseline, adding actual adaptivity automatically, and thus potentially requiring less manual intervention and manual authoring.

RI2.2. Where possible – labels should be added

Depending on the strategy, the goal map also needs to be labelled with semantically-rich labels and weights. The static material is unlikely to contain any usable metadata. However, depending on the format of the original document, it may be possible to use data-mining techniques to interpret the input and generate labels for each section (and therefore, groups of sublessons). This could be further enhanced by using Semantic Web technologies (such as RDF [91] and OWL [102]) to crawl the web and identify metadata about the static document's concepts.

RI3. Adaptation, User and Presentation Layers

To fully automate the process of creating an adaptive course based only on static materials, the system would need to choose (or even generate) an adaptive strategy. In this context, the strategy that would need to be created for the importer would deal with the final three layers of the LAOS framework: the Adaptation, User and Presentation layers, which for our purposes are all specified within a LAG file [61].

RI3.1. The importer should provide one or more template strategies

Static content contains no information about how to automatically determine which strategy should be used. Indeed, the static content should be usable by a variety of strategies. One solution to this problem would be to default to a '*Show all*' strategy, which would be able to show the content with no adaptation (therefore resembling the original static document), as described previously.

However, it is possible that the static content might describe concepts in a variety of different ways (see requirement RI1.4). For instance, if an input source clearly has image, text and video elements within each section, it might be appropriate for the importer to automatically select a multimedia strategy (as defined in Section 2.6.2.4). Alternatively, if enough semantically rich labels can be acquired for the content (see recommendation 2.1), it may be possible to create a template (or suggested) strategy, based on these labels, such as a beginner-intermediate-advanced strategy (as defined in Section 2.6.2.1).

4.5 File Formats

It is also important to consider exporting to useful[19] file formats for such an importer, primarily to ensure the possibility of reuse. For instance, although the case studies presented later on in this chapter are extensions to MOT3.0, it is

[19] 'Useful' is used here instead of 'standard' or 'commonly used', etc., because there are no standards for adaptation yet. Ideally, popular formats should be used, but importantly, formats without information loss should be applied (i.e., if adaptation functionality has been generated in the import, this should still be present in the output format).

important to note that rather than importing the static content directly into the database, the extension works by transparently creating a CAF file [3], and then importing the CAF file into the system.

This principle of this requirement demonstrates the necessity of using an established framework for static content description as advocated in Section 2.9.2 , and ensures that if future development of the MOT system requires the database structure to change, the extensions will be future-proof as long as the new authoring (or delivery) system can import CAF files (and this format remains backwards compatible).

To some extent, this requirement is implicitly available if the authoring tool already has export facilities for relevant, useful formats, so we will not go into further details here.

4.6 Semi-Automatic Authoring

Although the factual Domain content can be imported from static materials, it is likely that authors will still need to edit the course, for fine-tuning. The semi-automatic authoring process has two principal uses, which can be characterized by pre-import and post-import editing.

4.6.1 Pre-import Editing

Pre-import editing occurs when the author modifies (or creates) the content of the original document, with the primary purpose of converting the document into an adaptive course. This requires the author to understand how the importer will interpret the content. For instance, Interbook[27] primarily supports pre-import

editing (although advanced users can edit the HTML/LISP [103] code directly), since the author normally creates/edits the source Word document with the knowledge of how Interbook will interpret the document. Moreover a lot of the content of the document (i.e., the comments) is specifically created to express the adaptation.

4.6.2 Post-import Editing

By contrast, post-import edits can be made after the author has imported the document. This would be especially useful when used in conjunction with *pre-existing* learning materials that were created for non-adaptive delivery (e.g., existing lecture slides). After the static material has been processed by the importer, it is often likely that the author would need to manually edit the content, or at least label the content to apply an adaptation strategy. As explained in Section 4.4 the importer will be unable to definitively apply metadata or a complex strategy to the static material, so it is at this post-import editing stage when the author can edit the metadata and choose (or create) an adaptation strategy.

4.6.3 Two Use-Cases for Semi-Automatic Authoring

To illustrate the requirements above, we present in the following two use-cases for semi-automatic authoring.

4.6.3.1 Reusing Existing Content

Rather than recreate entire modules from scratch, many content authors may want to reuse existing content [104]. For instance, a lecturer might want to create an adaptive course based on a module that has been previously taught (offline) in a university (for instance, as revision material [105]), or content that has previously been delivered using a non-adaptive LMS. Alternatively it is possible that an

adaptation specialist (with a background in pedagogy) might acquire a repository of static content, and want to create a series of adaptive courses based on this content. The semi-automatic importer described in this chapter can assist with this reuse process. The process is called semi-automatic, and not automatic, because, as described above, the author might decide to do some post-import editing on the resulting Domain and Goal Maps, and would have to select (or create) a relevant strategy, and apply the appropriate metadata.

4.6.3.2 Authoring Content using more Familiar Tools

Rather than learning how to edit content within the AHAS, it is likely that a content author would like to author using the same software that they usually use. This removes the requirement from the author to change their working routine. If the content author understands how the static material will be interpreted, they should be able to do most (or potentially all) of the content creation within the more familiar word processor (pre-import editing), with or without some (minimal) post-import editing.

4.7 Case Studies for Semi-Automatic Authoring Importers

This section describes the creation of two semi-automatic authoring filters, which can interoperate with some of the most frequently used authoring tools. However, it should be relatively easy for programmers to create converters for other content types, based on the principles described in this chapter. The filter should then be documented to allow the author to structure the document in a way that can be delivered adaptively (pre-import editing).

4.7.1 Import Case Study 1: MediaWiki

MOT3.0 provides the ability to download the contents of any named MediaWiki [96] article (e.g., from Wikipedia [95]) and split the article into different concepts according to its structure. We particularly chose to target MediaWiki because of its popularity, and its hierarchical article structure which can be easily converted to a domain map.

As described by the sections above, the purpose of this MediaWiki importer serves the two important roles.

4.7.1.1 Reuse of existing content (post-import editing)

We considered that, due to the popularity of Wikis, and the extent to which they cover various subjects, many authors may find it useful to base a particular adaptive course on pre-existing content from a Wikipedia page. For instance, if an author wants to teach a course about *PHP*, they might find it useful to divide the content of the course into sections. By importing the already existent PHP article [106] from Wikipedia into MOT3.0, these concepts will be automatically created from the sections (including *History*, *Usage*, Security and *Syntax*) along with the text content of the different sections. The author could then do some post-import editing within MOT3.0 to change the content (or structure) of the article, if required. Of course, this being Wikipedia, it would be strongly recommended to the content author to at least check the factual accuracy of the article, in a post-import editing phase.

4.7.1.2 Creation of new content (pre-import editing)

There might be some authors who regularly contribute to Wikipedia, and are therefore more comfortable with the idea of using the wiki mark-up language to

create formatted content. One way of catering for these authors would be to install a local MediaWiki [96] on the authoring server, which has the primary purpose of feeding content into adaptive courses. The author could then freely write their own content (or modify existing content) within their installation, without fear of rendering the common version unusable for generic purposes. The author could then use the importer to transform the article into a Domain Map and Goal Map.

4.7.1.3 Interpreting a Structure from MediaWiki

The first iteration of this importer script uses a PHP script to download the WikiText contents of the named article, which then parses the wiki markup[20] to extract the individual sections of the article, and their level within the article by counting the number of '=' signs around the header. An example of this syntax is shown in Figure 4.1, with the extracted hierarchy shown in Figure 4.2.

```
==History==
PHP originally stood for...
===Release History===
Beginning on June 28, 2011...
==Usage==
```

Figure 4.1: Some sample WikiText from the PHP article [106]

Separating the content of the article in this way fulfils requirement RI1.2 (in the sense that separation is guaranteed and that a variety of content alternatives is created, in the form of sections) since the granularity of the concepts matches the

[20] http://en.wikipedia.org/wiki/Help:Wiki_markup

granularity of the article's sections. This assumes that the smallest indivisible unit of content is the article's sections, otherwise more division would be required to fully satisfy RI1.2.

Interpreting the level of the headings allows hierarchical relationships to be generated that match the outline of the original article (thus satisfying requirement RI1.3, by generating at least one type of relationship between extracted domain concepts). An example of an imported Wikipedia page is shown in Figure 4.2.

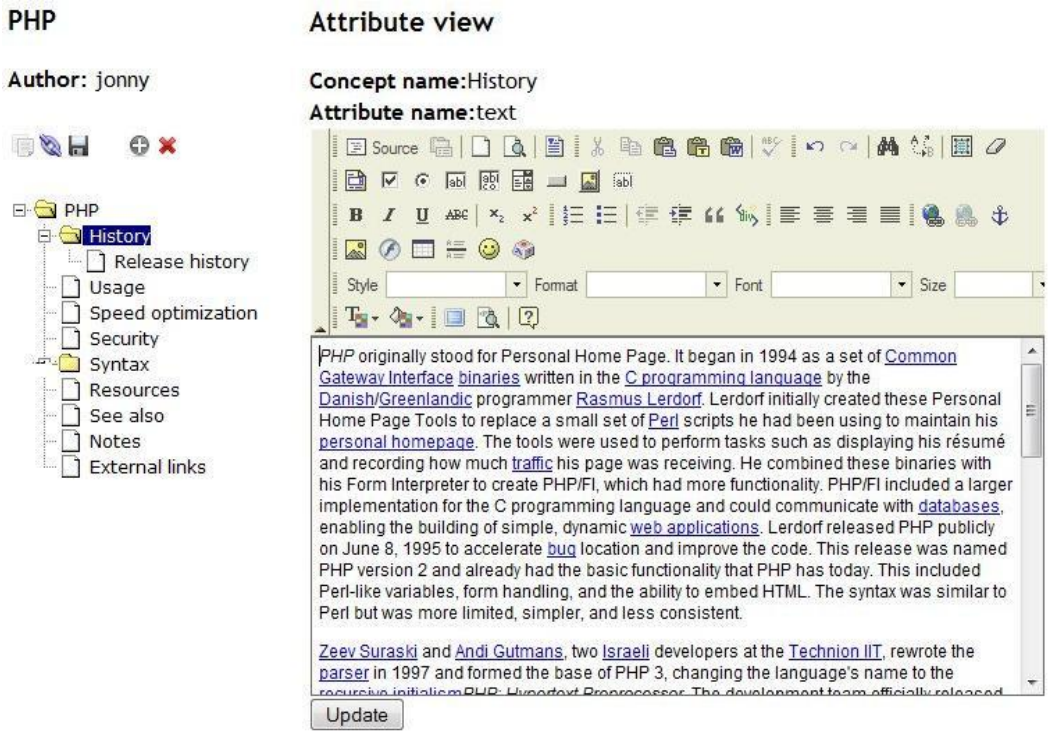

Figure 4.2: Imported Domain map in MOT3.0 based on the PHP article [106]

The importer script created in this first iteration automates other PHP scripts from a local installation of MediaWiki to convert each section of the article into HTML. However, the first iteration of this importer could not import the images – thus only partially fulfilling requirement RI1.1. Please note that this issue was later fixed, and is documented in Section 4.7.1.7 .

The required local installation of MediaWiki could also be used directly by the author for writing the content (as described in Section 4.7.1.2).

The accompanying Goal Map also matches the structure created for the Domain (interpreting hierarchical relationships as pedagogical relations, as explained previously), which can be used with a *show-all* strategy (fulfilling RI2.1 and RI3.1). If the authors wish for a more complex strategy, it would require them to manually add their own labels (and weights), and choose (or create) an adaptation strategy.

4.7.1.4 Evaluation

As part of the evaluation with the Romanian students described in Chapter 3, the students were also asked about the Wikipedia importer[21]. As with the previous questions, the Wikipedia specific questions were based around a Likert scale. The questions were:

1. Do you like the theoretical idea of using Wikipedia content in Domain Maps and Goal Maps? The ability to import content from Wikipedia would be:

   Very Useful, Useful, Not Useful, Useless

2. What do you think of the structure of an imported Wikipedia article in MOT3.0? The structure created by the importer is:

   Completely Preserved, Mostly Preserved, Partly Preserved, Mostly Destroyed, Completely Destroyed

[21] Although the importer supports any compatibly MediaWiki installation, for simplicity the evaluation limited usage of the importer to Wikipedia

3. What do you think about the speed of importing a Wikipedia article? Speed of importing is:

   Fast, Adequate, Slow, Too Slow

4.7.1.5 Evaluation Results and Discussion

As described in the previous chapter, the students were not obliged to complete the questionnaire. 13 of the 25 Web Application Development students and 18 of the 39 Semantic Web students responded. Out of the 31 responses to the questionnaire, 20 said they thought the ability to import from Wikipedia would be *very useful*, with 10 saying it would be *useful*. The remaining respondent said it would be *not useful*, adding a comment which said that given that Wikipedia contains some information which isn't correct, maybe the importer could help to propagate erroneous information. This comment raises a widespread concern [107], [108]. However as described in the above sections, it is hoped that a responsible content author would only use the imported content as a starting point for the lesson, rather than simply publishing the contents unchecked and unedited (where necessary).

When asked about the structure of the imported content, 10 out of 31 (32.26%) said they thought it was *Completely Preserved*, 12 (42%) said it was *Mostly Preserved*, and 6 said it was *Partly Preserved*. Of the remaining two users, one said they didn't know "how to check" – which suggests the question may have been misunderstood – whilst the other user said that they didn't manage to import an

article. However, this latter user also said they “thought it was stuck, but it wasn't, maybe it's because of my internet connection”.

13 out of 31 users (42%) said the speed was *Fast*, and a further 11 (35.5%) said the speed was *Adequate*. 2 of the users said the importer was *Slow*, with 5 users saying it was *Too Slow*. However, some of the comments such as “*First, I chose Car as the article. And it stops responding until I restarted my computer and tried another article*”, sound as if it might have been an internet connection problem. Nevertheless, speed was one of the issues that remained to be re-analysed and worked on for the follow-up implementation.

4.7.1.6 Second Evaluation

A second (short-term) evaluation was also performed at the University of Warwick with six volunteer course authors and designers. Each participant was experienced in the process of both teaching and web-design. Further aspects of this evaluation are documented in Section 6.6 . The authors were all already familiar with Adaptive Hypermedia. Each user was given a short demonstration of the system, and a chance to experiment with the different features.

With regards to the Wikipedia importer, the course authors and designers were asked the following.

What do you think about the usefulness of importing other content into MOT3.0?

Being able to import Wikipedia content as domain maps is:

Very Useful, Quite Useful, Slightly Useful, Not Useful

Being able to import Wikipedia content as goal maps is:

Very Useful, Quite Useful, Slightly Useful, Not Useful

In this experiment, all six respondents said they thought that being able to import Wikipedia content as domain maps was *Very Useful*. For importing Wikipedia content as goal maps, 4 out of 6 said it was *Very Useful*, whilst the other 2 said it was *Quite Useful*.

This evaluation also provided useful qualitative feedback from the course authors/designers. Specifically, with regard to importing, one respondent said "*The content needs to be expanded. How about movies (YouTube)? Flash presentations? Websites?*", whilst another said "*This is a very nice feature but needs to be extended. In particular – images!*".

Whilst the above questions ask about the general idea of these import features, the content creators were also asked more specifically "What do you think about the functionality of importing content into MOT3.0?", with the following 4 sub-questions.

The content of the imported Wikipedia article is:

Good, Sufficient, Bad, Insufficient

The number of attributes extracted from an article is:

Good, Sufficient, Bad, Insufficient

The type of attributes extracted from an article is:

Good, Sufficient, Bad, Insufficient

Speed of importing an article is:

Good, Sufficient, Bad, Insufficient

Question	***Mean***	***StdDev***	***T***	***P***
The content of the imported Wikipedia article	1.667	0.516	7.91	0.001
The number of attributes extracted from an article	1.5	0.548	6.71	0.001
The type of attributes extracted from an article	1.333	0.516	6.32	0.001
Speed of importing an article	0.333	1.033	0.79	**0.465**

Table 4.1: Results and t-test statistics of the Wikipedia question

Table 4.1 shows the results of a one-sample t-test against an expected score of 0. The content creators appreciated the content of the imported article, with 4 out of 6 saying the content was *Good*, and the other 2 saying it was *Sufficient*. Similarly, 3 out of 6 said that the number of attributes imported was *Good*, with the rest saying it was *Sufficient*. However, only 2 out of 6 thought the type of attributes extracted from the article was *Good*, with the other 4 feeling it was *Sufficient*. The comments about this recommended that images from the article should also be imported (which was also requested in the Romanian evaluation).

There were, however, a few concerns about the speed of the import process, with 4 out of 6 saying it was *Sufficient*, and the other 2 saying it was *Bad*. Since the time

taken for importing an article is proportional to the size of the article, one comment suggested that it would be beneficial to inform the user of how long the article is before the import. Another way of improving this would be to provide a progress bar.

4.7.1.7 Importing Images

In response to the user feedback from the above evaluations, a second stage of the importer was implemented, which also allowed for extraction of the images from the MediaWiki article. This was achieved by sending a request to the MediaWiki API [109] – in this case to Wikipedia, using a URL such as:

```
http://en.wikipedia.org/w/index.php?title=$articleName&action=render
```

where `$articleName` is the name of the article to be imported.

The above URL returns the rendered HTML content of the page, without the Wikipedia skin. This removes the necessity for the local installation of a MediaWiki, and ensures that the article is rendered using the most accurate (up-to-date) version of the MediaWiki rendering engine – therefore guaranteeing that it will be as close to web-based version of the article as possible. This import method also provides a speed increase over previous versions.

For each section heading of the article, a new concept is created, with a *title* attribute. A text attribute is then also added, containing the HTML content of the section (the HTML between the current heading and the next heading).

This method of importing does not download the source WikiText of the article (only the rendered HTML) – so a change in the way that relationships are extracted was also required. Instead of interpreting the structure from the number of '=' signs around the heading, the new method parses the HTML for the '*Contents*' box at the top of the article (an example of the HTML is shown in Figure 4.3).

```
<ul>

<li class="toclevel-1 tocsection-1"><a
href="#History"><span class="tocnumber">1</span> <span
class="toctext">History</span></a>

<ul>

<li class="toclevel-2 tocsection-2"><a
href="#Licensing"><span class="tocnumber">1.1</span>
<span class="toctext">Licensing</span></a></li>

<li class="toclevel-2 tocsection-3"><a
href="#Release history"><span
class="tocnumber">1.2</span> <span
class="toctext">Release history</span></a></li>

</ul>
```

Figure 4.3: Extract from the HTML of a Wikipedia contents box – extract based on PHP article [106]

4.7.1.8 Further improvements to the MediaWiki Importer

Although the first and second version of the importer create only *title* and *text* attributes for each concept, a third version was created that could separate the text from the images. This could then be automatically paired with a multimedia strategy (defined in Section 2.6.2.4) that (depending on the user's preference) could show either the images, the text or both.

Similarly, version 3 of the importer is able to download the 'information box' of the article. It is then inserted as an '*infobox*' attribute into the root concept of the article – or alternatively it could be appended to the text content of the root

concept. Separating each of these types (*text*, *image* and *infobox*) into different attributes would help to fulfil requirement RI1.4 to a greater degree.

Note that because of the variety of content available on Wikipedia, there was no obvious way of specifying the adaptation labels, and therefore RI2.2 has not been implemented. This also means that the default adaptation strategy for an imported Wikipedia article is a show-all strategy (RI3.1).

In 2010, as part of the 'Dynamic Web-based Systems' module, a fourth year project group implemented some of the principles described in this chapter by extending the Wikipedia importer to create a way of crawling an entire category page[110].

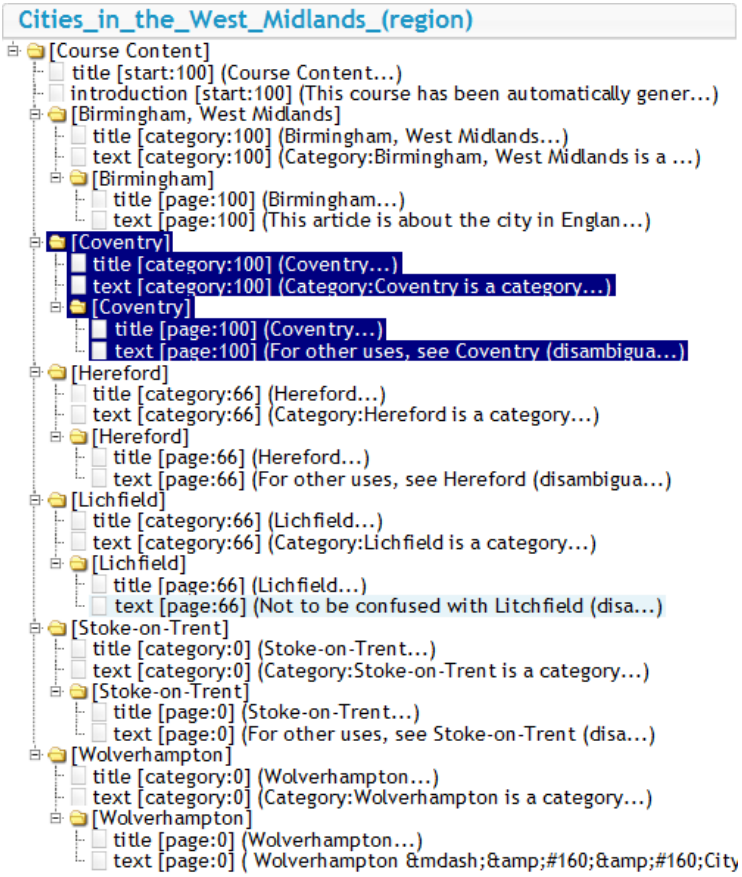

Figure 4.4: An imported Wikipedia category page, with labels and weights according to the importance of the page

In their implementation, when the user imports a category page, the script (recursively) imports all pages that belong to that category, and also looks at how many links there are to that page, therefore providing a form of measurement of the importance of the page. This is a similar technique (although extremely simplified) to that used by search engines to rank the importance of pages [111]. This importance is then scaled as a percentage, and is added automatically as a weight to the page's sublesson by the importer. This means that the imported course does have some interesting semantic pedagogical labels and weights (thus implementing RI2.2), and allows a strategy to be created which only shows the most important pages in a particular category (RI3.1).

Chapter 11 describes another method of extending the Wikipedia importer to extract semantically meaningful relationships between domain concepts.

4.7.1.9 Future improvements to the MediaWiki Importer

Note that – as with all imported content – besides checking for accuracy, we are expecting the author to check for copyright issues surrounding the imported images. For a more automatized solution, future improvements could look up the licences of the images from the Wikipedia file metadata, and inform the user of how the image may be used.

Future research could also add text mining techniques to the importer. For instance a *keywords* attribute could be created for each concept, based on analysis of the text content. Moreover, a follow-up version of the system could allow the links in the MediaWiki article to be crawled, thus generating a larger domain map. The

question here is what the ideal depth of this import is (e.g., links in the article are of depth 1, links of links are of depth 2, and so on). It is clear that it should be smaller than six, because this may end up with a very large part of the Internet, as there are clearly parallels to the "*Six degrees of separation*" idea [112]. Dolan [113] has investigated a similar principle using links between Wikipedia articles. Links between articles on Wikipedia provide a way of inferring a relationship between these articles. Future research could investigate how to crawl between articles, and therefore be able to create more comprehensive domain maps that contain data that span over several related MediaWiki articles.

4.7.2 Import Case Study 2: Presentation Importer

Presentations are commonly used in education to display lecture slides. Here, we define a presentation to be any series of slides which can be read by Apache OpenOffice Impress. This includes the Microsoft PowerPoint (.ppt), Office Open XML Presentation files (.pptx), OpenDocument (.odp) and the older OpenOffice.org (.sxi) formats. Moodle also supports importing presentations from Microsoft PowerPoint in a similar way [114].

Whereas the MediaWiki format has a predefined structure of headings and subheadings, presentations can have a much wider variety of style (and therefore structure). Indeed, unlike the WikiText format, these presentation file formats do not clearly separate the '*domain*' content from the '*presentation*'. This makes extracting the raw '*domain*' content from the presentations much more difficult – therefore making reuse difficult (and, tangentially, reinforcing the importance of

the ‘separation of concerns’ principle (as described in Section 2.10.1) within (adaptive) hypermedia.

4.7.2.1 Reuse of existing content (post-import editing)

It is likely that many lecturers will have already created lecture slides using some presentation software that is compatible with the above formats. After they have imported the presentation, they are free to edit the imported Domain and Goal maps (post-import editing), possibly enhancing their original ‘linear’, ‘real-world’ presentations by adding extra content, such as hyperlinks, to other resources or videos. Similarly, it is likely that an author may wish to reuse somebody else’s presentation slides. As with the MediaWiki importer, it is left to the author to check for accuracy and copyright issues within the imported material.

4.7.2.2 Creation of new content (pre-import editing)

It is also possible that a content author might wish to create original content for the AH course, using PowerPoint. This is particularly likely if a user needs to draw a diagram, or create a slide based on a pre-existing slide.

4.7.2.3 Interpreting a Structure from Presentations

When the user uploads a file to the server (via the MOT3.0 import interface), the server automates OpenOffice Impress[22] to convert the file into a series of HTML and JPEG files. This is a standard feature of Impress, to allow the presentation to be exported to a webpage, and using an established external piece of software like this ensures that the content can be accurately converted to domain content (RI1.1).

[22] http://www.openoffice.org/

PowerPoint also offers this facility, however it was decided to use Impress so that it could interact with PHP scripts from a Linux-based server.

The design decision was taken that for each slide in the presentation file, MOT3.0 will generate a concept. The presentation author will already have separated the content separate slides, so the content should already be separated into standalone pieces (RI1.2). Each concept will contain the following attributes.

- **Title**: On most slides Impress is able to automatically identify the title of the slide and put it in an <h1> tag (and the <title> tag) at the top of the generated HTML file. Although most presentation applications provide a template layout for each slide, it is likely that the layout of some slides will be drastically changed by the user (for instance to show a fullscreen diagram), which would mean that no title can be extracted. In this case MOT3.0 records the title as 'Untitled'.
- **Text**: The HTML file also contains the HTML representation of any text elements from the slide. Background styles and images are ignored, however paragraph styles, bullet points and most HTML compatible font styles (such as colour, italic and bold emphasis) are preserved.
- **Image**: The exporter automatically generates a JPEG file of the slide. This not only maintains the style of the slide, but also preserves any diagrams or images that could not be exported to the text attribute. As part of the import process, the MOT3.0 PHP script copies the generated images into a publicly readable folder on the webserver, and the attribute created

consists of an `<img>` tag with a `src` attribute that points to the image on the webserver.

- **Notes**: The presenter's notes are also written to the HTML file (the notes occur under a '`<h3>Notes:</h3>`' heading). MOT3.0 can identify the notes part and store it as a separate attribute.

We considered it is important to preserve each of these elements of the slide to ensure that no information is lost. The aim was to extract as many different types of attributes as possible from the original linear content. This provides alternative methods (attribute types) for describing the same concept (slide) – therefore adhering to requirement RI1.4. Content imported in this way could therefore be immediately used, for instance, in a multimedia strategy (as defined in Section 2.6.2.4).

A revision strategy could also be easily created using such imported material. For instance, if a presenter's notes contained a more detailed script about the slide, the learner could initially be shown the concept's *image* and *notes* attributes. However, when the learner returns to this slide after reading the rest of the course, the concept might only show the *text* attribute, to remind the learner of the slide's bullet points, without making the user read the full *notes* attribute. Although such strategies only use content attribute types (rather than pedagogical labels), these example strategies fulfil RI3.1.

Currently, the hierarchies generated by the importer are linear, simply containing a root concept for the first slide, with each other slide as a child concept of the root.

Since there are no direct relationships between slides (other than *Next* and *Previous*), this basic format fulfils RI1.3. As with the MediaWiki importer, the structure of the Goal map simply mimics the structure of the Domain map – thus interpreting (quite straightforwardly) the *Next* and *Previous* relationships as prerequisite relations, and fulfilling RI2.1.

4.7.2.4 Evaluation

The presentation importer was developed after the Romanian evaluation described in Section 4.7.1.4 , however it formed a major part of the evaluation with the six course designers at the University of Warwick (introduced in Section 4.7.1.6). In particular, they were asked the following (the frequency of each answer is in brackets).

1) What do you think about importing Presentation content?
    a) Being able to import Presentation content as domain maps is:

        Very Useful (6), *Quite Useful (0)*, *Slightly Useful (0)*, *Not Useful (0)*

    b) Being able to import Presentation content as goal maps is:

        Very Useful (5), *Quite Useful (1)*, *Slightly Useful (0)*, *Not Useful (0)*

2) What do you think about the functionality of importing content into MOT3.0?
    a) The content of an imported Presentation is:

        Good (2), Sufficient (4), Bad (0), Insufficient (0)

    b) The number of attributes extracted from a Presentation is:

        Good (3), Sufficient (3), Bad (0), Insufficient (0)

    c) The type of attributes extracted from a Presentation is:

        Good (2), Sufficient (4), Bad (0), Insufficient (0)

d) Speed of importing a Presentation is:

Good (0), Sufficient (2), Bad (4), Insufficient (0)

As with the Wikipedia importer, all 6 users thought that importing presentation content as domain maps was *Very Useful*. 5 out of 6 users thought importing the presentation content as goal maps was *Very Useful*, with the other saying *Quite Useful*.

There was also an interesting comment concerning the type of attributes extracted:

"For PPT, different ways of extracting finer granularity need revisited - e.g., extracting individual images from a page, extracting bullet points as separate concepts or attributes, etc."

This comment is interesting because it points towards a way of creating a hierarchy out of an otherwise flat structure, and could be investigated in future research.

The major issue surrounding the importer was the speed of importing. The processing time is proportional to the number of slides in the presentation, and each slide took around 5 seconds to process. As with the Wikipedia importer, this is not scalable if several users try to use the feature at the same time. The responders commented that it would be better to have a progress bar (or some other feedback) to show how many slides need to be imported, and provide an estimated completion time.

Table 4.2 shows the results of a t-test for the presentation question against an expected result of 0.

Question	***Mean***	***StdDev***	***T***	***P***
The content of the imported Presentation	1.333	0.516	6.32	0.001
The number of attributes extracted from a Presentation	1.5	0.548	6.71	0.001
The type of attributes extracted from a Presentation	1.333	0.516	6.32	0.001
Speed of importing a Presentation	-0.333	1.033	-0.79	**0.465**

Table 4.2: T-Test results compared with an expected result of 0

4.7.2.5 Future Improvements to the Presentation Importer

As suggested by the evaluation comments, some progress bar information needs to be relayed to the user. The comment described above about extracting a finer granularity from the presentation slides (e.g. bullet points and individual images), is also very interesting. For instance, if a slide contains animated bullet points, it would be interesting to simulate this animation by separating the bullet points into separate attributes. The corresponding goal model sublessons could then have a label named '*bullet*', with a weight relating to its sequence in the animation. The system could then recommend a strategy which would be similar to a roll-out strategy (defined in Section 2.6.2.2).

Future research will investigate how to analyse the titles of the slides, and structure the generated hierarchy accordingly. For instance, if a presentation contains four

consecutive slides with the same title, the importer could infer a relationship between these slides, and restructure the domain hierarchy accordingly (i.e., all four concepts will be children of a grouping concept). However, since there is no de facto standard for designing slides it might be difficult to reliably interpret such a structure.

4.8 Future Work: Extending this import principle to other formats

The previous section has provided two case studies, however the requirements described in this chapter should also be transferable to other file formats. Section 4.2.1 described other tools that are commonly used in education. This section describes how some of the techniques described in this chapter could also be applied to these other formats.

4.8.1 Microsoft Word

As with the Wikipedia category importer, a group of students from the 2010 cohort of the 'Dynamic Web-based systems' module created a Microsoft Word macro to interpret the structure of a document. It allows the user to add comments (similar to Interbook [27], as described in Chapter 2) to define how the domain content should be split into attributes. The macro then converts the document into a CAF file (using Microsoft Word's built-in HTML export feature), ready for import into MOT, where the HTML content can be edited, and goal model labels can be added.

4.8.2 LaTeX

Based on the recommendations provided in this chapter, it should be reasonably simple to interpret the structure of a LaTeX document to create a well-structured

domain model. This is largely because of LaTeX's focus on a WYSIWYM[23] design [115], which has a clear separation between presentation and content, and encourages authors to divide their document into sections. It should, therefore, be relatively easy to extract domain content from a LaTeX source file. Tools such as Tex4ht [116] could then be used to convert the actual content to HTML.

4.8.3 HTML

The HTML specification contains heading tags such as <h1> and <h2>, which would itself to a clear separation of content parts. In practice, however, the way that content authors actually implement the various HTML standards do not provide a clear distinction between the content and presentation [117], so it is difficult to create a reliable automated method of extracting content and interpreting the original structure from an arbitrary page. For this reason, rather than simply creating a generic importer, it may be necessary to develop a series of templates that can be used to identify the most important bits of content from a web page.

Alternatively, Chapters 9 and 10 describe methods of referencing external HTML pages, and thus reusing existing HTML pages.

4.8.4 Portable Document Format

As described in Section 4.2.1.6 it is likely that most PDF files will have been authored in one of the above formats. In such a situation, it is likely that the import

[23] 'What You See Is What You Mean' – this is a clear distinction from the more familiar WYSIWYG ('What You See Is What You Get') editors that have been mentioned elsewhere in this thesis.

process will be more successful if applied to the original source document. However, it is also possible that an adaptation author would wish to reuse content from a PDF that has been created by another author. In this scenario, the author might not have access to the original document. Future research could investigate how to use tools such as pdftohtml [118] to convert PDFs to HTML files, and then interpreting the HTML.

4.9 Conclusion

This chapter has described the importance of allowing authors to be able to import widely-used static-content file formats into an adaptive hypermedia authoring system. We have defined a number of imperatives to describe how content can be interpreted from its original format.

The formats that have been detailed in this chapter are designed for linear rather than adaptive presentation. As such, the conversion process cannot extract any adaptation information from these file formats. It is hoped that other programmers will be able to use these recommendations to create ways of converting from other file formats into adaptive material.

The recommendations presented in this chapter have suggested that it may be possible to create automatic specific adaptation strategies, depending on the file format used. However, since domain (and goal) content can be reused and taught in a variety of different ways, there is no need to enforce a particular adaptation strategy. Indeed, the author can make a *creative* and *pedagogical* decision about the best way to teach the content.

5 Putting it all together: The MOT Toolset

5.1 Overview

The authoring tools described in the previous chapters have focussed on purely the *static content* aspect of the authoring process. This chapter introduces a new tool, PEAL, which focuses on allowing the author to specify the *adaptivity* for the adaptive course. The chapter also expands on the requirements that were introduced in Chapter 2 to form a set of imperatives. The chapter also describes the usage of these systems by a 4th Year Computer Science module.

5.2 Imperatives for Authoring

Chapter 2 suggested, and evaluated, tools with respect to a series of requirements. Whilst these requirements are useful for evaluating the current state of the authoring tool, the evaluations (in Chapters 3 and 4) also provide clues as to what is required of an authoring tool. It is, therefore, also necessary to revisit and refine the requirements.

Hence this section builds on the design principles that were introduced in Chapter 2 to introduce a set of *imperatives*. These imperatives are divided into two areas; Complexity and Support.

5.2.1 Complexity Imperatives

The following imperatives are extracted based on requirements that address the problem of complexity within adaptive hypermedia authoring systems.

CI1. **Separation of concerns**: The authoring system needs to allow the user to separate adaptive content from static content, and, preferably,

separate other elements of adaptation (such as user model or presentation model data). This provides flexibility and reusability as described in 2.9.1 .

CI2. **Use of frameworks**: As described in 2.9.2 (DP1) frameworks usually already implement CI1, the separation of concerns, so using a framework is a straightforward way of respecting this principle. Frameworks are also 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 (moving away from the early trial-and-error approaches to building adaptation). Basing an authoring system on an established framework is, therefore, a straightforward way of conveying lessons learnt from previous research.

CI3. **Use of standards**: Ideally, standards could replace the use of a framework, if they would be able to reproduce all aspects of the authoring process. However, the adaptation process of adaptive hypermedia is not yet standardized. To ensure a wider range of reuse, standards should be used where possible. Thus, e-learning standards should be used where possible for adaptive e-learning systems.

5.2.2 Support Imperatives

Since authoring for adaptive hypermedia is relatively 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. We have grouped the imperatives resulting from this into the category of support imperatives.

SI1. **Simple access** to (content) pieces that need to be reassembled for different types of adaptation. This requirement is a refinement of the usability principle described in 2.9.2 . In particular the system should allow the author simple access to the following.

- **Reordering** of the domain/goal maps (or equivalent).
- **Labelling** of sublessons (or equivalent).
- **Searching** to allow authors to find pieces of content that can be reused.
- **Copying & pasting** of concepts, attributes, sublessons or parts of content.
- **Linking** between fragments of content.
- **Editing** of (for instance) content and strategies (where applicable).

SI2. **Shallow learning curve**: As with simple access to content, the system should be easy to learn, and should allow beginner authors to use a wide variety of features in simple ways.

SI3. **Familiarity**: The system should be familiar to the author, and thus use as many conventions as possible which are familiar to the author – such as functionality, formats, etc. In particular, the system should provide the following.

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.). This imperative mirrors the principle set out in DP3.

b) **Interoperability with familiar content 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. This imperative is based on the principles identified in Section 2.9.2 (DP2.a) but it is more generic, as it, in principle, encompasses any familiar content creation formats, rather than just linear formats. Moreover, the stress is on the familiarity of the format, and not on the fact that it is a linear format or otherwise. For instance, graph representations (e.g., RDF [91]) should also be considered for import. In reality, adaptive content is rarely familiar to authors, and it is even rarer for such a format to be directly importable. The principle described in DP2.a concerning reuse of content was dropped as a separate requirement, as it is now included in the overall Support Imperatives (described above).

c) **Interoperability with familiar course creation standards**: The system should be compatible with standards used in many other (e-)learning systems, for instance SCORM, LOM, IMS-CP, IMS-QTI and IMS-LD – this is based on the principle described in DP2.a.

d) **Interoperability with familiar course creation systems**: The system should be able to import (and export – possibly with some loss of adaptivity) to Learning Management Systems (LMS), which (as described in Chapter 2) are currently widely used for the creation and delivery of online material.

SI4. **Adaptive functionality**: The authoring system should adapt, where possible, to the needs of the author, suggesting relevant features (note that this is independent of the adaptation to the learner).

5.3 The MOT Toolset

The MOT toolset consists of two separate authoring tools; MOT3.0.1 and PEAL.

5.3.1 MOT3.0.1

MOT3.0.1 provides the authoring environment for the creation and editing of static material that is necessary for the building of adaptive courses. This includes structuring the content, separating it into relevant concepts, and adding metadata and pedagogical labels.

The evaluations documented in Chapters 3 and 4 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 preferences 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, to create MOT3.0.1 (as described in Chapter 3). Please note that this change from MOT3.0 to MOT3.0.1 was a relatively minor change (in response to bug reports and usability feedback), so many of the principles described in this chapter also apply to MOT3.0.

5.3.2 PEAL

To build a complete adaptive course, the adaptation specification has also to be written. For this reason, the MOT toolset also contains PEAL [53], the adaptation strategy editor. PEAL was written by Jon Bevan, an undergraduate student at the Computer Science department of the University of Warwick, as part of a third year BSc project, and provides a basic integrated development environment (IDE) for the LAG language [53]. In theory, the adaptation strategy in LAG could be written in any text editor. However, PEAL offers a number of additional features, as follows.

- **Syntax highlighting**: Different coloured highlighting depends on the type of keyword used (e.g., reserved word, invalid/valid syntax, variable, etc).
- **Auto-Completion**: When a user types a statement, a drop-down menu appears with suggestions from the LAG grammar.
- **Strategy Library**: Users can save their completed strategies to the web-based storage. There are two types of storage:
    - **Private**: With *read-write* permissions given only to the strategies author;
    - **Public**: With *read* permissions given to all users – the author has *read-write* permissions.

  Alternatively, the user can download the strategy (as a .lag file) to their local file system. The strategy library feature, coupled with the ability to exchange LAG files with other users, allows reusability of adaptation strategies.
- **Code Fragment Library**: Similar to the strategy library, the code fragment library allows authors to save small sections of code that can be used again

in other strategies. As with the strategy library, code fragments can be saved as either private or public. This feature extends reusability to smaller snippets of code, thus enabling finer grain reusability.

- **Strategy Wizard**: This provides a basic dialog box to encourage the user to specify a description for the strategy, and allows the user to choose which variables will be used by the strategy. This is aimed at the beginner programmer, and gives them templates for the programs they wish to build.

PEAL can now be combined with MOT3.0.1 to form the '*MOT Toolset',* an adaptive hypermedia authoring system that allows authors to create both static content and adaptation content.

5.4 Authoring Imperatives applied to the MOT Toolset

This section describes the two authoring tools from the MOT toolset (MOT3.0.1 and PEAL) with respect to the imperatives described in Section 5.2 . This evaluation is not based on user feedback, but on the evaluation of the design and implementation aspects and how they map (or not) onto the imperatives. An evaluation with users is done separately, and is presented in Section 5.5 .

5.4.1 MOT3.0.1

This section describes how MOT3.0.1 respects the set of complexity and support imperatives defined in Section 5.2 .

CI1. *Separation of concerns*: As previously explained, MOT3.0.1 is dedicated to authoring and labelling of content only. Adaptation strategies need to be edited via a different tool. Thus, MOT3.0.1 obeys principles of separation of concerns.

Moreover, it also distinguishes between domain (map) authoring, and pedagogical (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 can be different from the adaptation author.

CI2. *Use of frameworks*: As with previous versions of MOT, MOT3.0.1 is based on the LAOS authoring framework.

CI3. *Use of standards*: To ensure interoperability with other, more established, educational systems, MOT3.0 and MOT3.0.1, (unlike MOT1.0), are able to import content from SCORM, IMS-CP and IMS-QTI. 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 still need to be added manually after the import. 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.

SI1. *Simple access*: MOT3.0 provides a new interface for creating and loading Domain and Goal map content. MOT3.0 (and therefore MOT3.0.1) addresses the 'Simple access' imperatives with the following features.

- The JavaScript tree display 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 manoeuvre. This tree aims to make it easier to **reorder** domain maps and goal maps as well as **labelling** them. The tree structure does not require the page to be reloaded after every small operation. This provides a smoother user experience by limiting the number of page refreshes (see Section 3.2) therefore it improves the speed and look & feel of the **copying & pasting** and **linking** aspects of this imperative.

  As described in Section 3.6 , MOT3.0.1 improves on MOT3.0 with regard to labelling sublessons by allowing multiple sublessons to be selected at the same time – thus speeding up the process of labelling items.
- MOT3.0 added a **search** facility, which is also implemented using Ajax. The search facility is employed in many aspects of MOT3.0.

Specifically, MOT3.0 allows users to type a search term, and to choose a search scope (as shown in Figure 5.1). This allows users to find content based on any combination of the following content types: *Concept Map (Domain) Titles*, *Goal Map Titles*, *Keywords* and *Concept Titles*. Rather than searching for entire concept

maps, the *Keywords* and *Concept Titles* scopes allow the user to search for individual concepts.

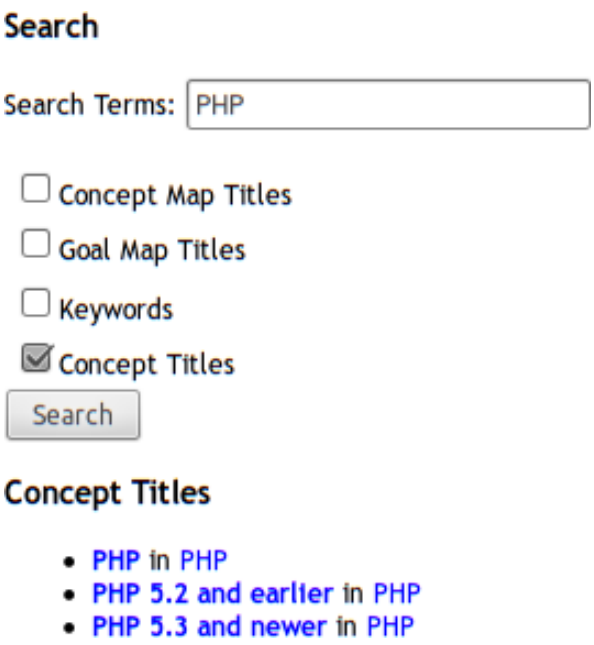

Figure 5.1: Search functionality in MOT3.0

Although the search functionality is available to users at all times through the navigation menu, it is also provided when performing copying/linking operations. For instance, if a user wishes to copy from one domain to another, the user is presented with the search form, allowing them to more quickly find the domain that they want to copy from.

- 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 a WYSIWYG HTML editor, allowing authors to input text either in a rich text (X)HTML editor, or in plaintext source mode. This replaces MOT1.0's simple editing window, which provided no formatting options and only allowed for simple text or manually-written (or pasted from other tools) (X)HTML input.

SI2. *Shallow learning curve*: An adaptive hypermedia authoring system must emulate an adaptive delivery engine, in that it needs to adapt to the user – here, the author or (in the case of adaptive educational hypermedia) the 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, meaning the menu can be smoothly updated to provide context-sensitive options – as described in Chapter 3.

Both MOT1.0 and MOT3.0.1 aim for a shallow learning curve. Functions are kept elementary, even if functionality is complex. Indeed, much of the extra functionality that was introduced by MOT3.0 is designed to simplify the process for the author, thus attempting to provide a shallower learning curve.

SI3. *Familiarity*:

a) *Consistency with existing applications and systems*: The drag-and-drop JavaScript methods of displaying the domain and goal maps were created especially to follow the more usual directory listing programs found in many operating systems (see Chapter 3). Similarly, the WYSIWYG editor provides a basic word processing environment similar to editors that are widely used for

document creation within blogging software such as Wordpress[24], content management systems such as Joomla[25] or e-learning systems such as Moodle[26].

b) *Interoperability with familiar content creation formats*: MOT3.0.1 implements the Wiki and PowerPoint importers that were described in Chapter 4 . This adds to the interoperability with e-learning standards, and allows for a wider range of familiar content creation formats to be used by authors. This is especially useful if they have already created content in these other formats.

c) *Interoperability with familiar course creation standards*: Chapter 4 also described how MOT2.0 had already created methods of importing content from other course creation systems via the SCORM, ITS-CP and ITS-QTI standards[60]. MOT3.0 also implements this importer. These standards are widely used within other systems such as Moodle, Sakai and Blackboard[27], thus also providing interoperability with familiar course creation systems.

SI4. *Adaptive functionality*: As with MOT1.0, MOT3.0.1 can suggest domain concepts that are related to another domain concept. 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). MOT3.0.1 can also calculate

[24] http://www.wordpress.com

[25] http://joomla.co.uk/

[26] http://docs.moodle.org/23/en/Text_editor

[27] http://www.blackboard.com/

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 '*Concept-oriented, relevance ranking method*' is used, as described by Hendrix et al. [80] through the following formula:

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

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 feature is adaptive in the sense that it assists the author to find concepts based on the current concept. However, to be truly adaptive, the system would need to adapt to the needs of the author.

5.4.2 PEAL

This section describes how PEAL respects the authoring imperatives described in Section 5.2

CI1. *Separation of concerns:* PEAL is designed to create adaptation strategies, and thus separates adaptation from content. Moreover, whilst MOT3.0.1 is to be used by the content author, PEAL is designed to be used by the adaptation specification author. This means that PEAL needs an author who has knowledge of programming.

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.

CI2. *Use of frameworks*: The PEAL tool is based on the LAOS framework [35], and moreover, on the LAG framework [119] for adaptation.

CI3. *Use of standards*: The PEAL system does not use standards, as there are no standards for adaptation. However, it uses the LAG adaptation language [53], and thus promotes reuse. This means that any system that can import from the LAG language (such as AHA! [56] or ADE [50]) can use PEAL as an adaptation strategy authoring tool.

SI1. *Simple access*: To assist the adaptation language author in accessing edited adaptation strategies and snippets, PEAL provides the following features for online storage.

- **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.
- **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. In such a way, simple access to programs and snippets of programs is promoted.

SI2. *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 some programming background, learning the PEAL language should, in principle, not be difficult.

By design, the LAG language aims to have a shallow learning curve, since it contains a reduced number of programming instructions, no variable typing, and in general, all simplifications possible in order to keep the learning curve shallow.

SI3. *Familiarity*: In addition to the features enumerated above, PEAL 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.

SI4. *Adaptive functionality*: PEAL helps the authors to complete their work in various semi-automatic ways, as follows.

- **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.
- **Code completion***:* Whilst typing the program code, PEAL allows the user to select the correct statements, operators and variables from a list of suggestions.
- **Strategy Wizard**: PEAL provides a strategy wizard that allows the author to define and initialize variables that will be used within the strategy.

5.5 Evaluations

In the Computer Science Department at the University of Warwick there is a module named “CS411: Dynamic Web-Based Systems”, which runs over two and a half months. This module is taught to a mix of Computer Science MEng and MSc students. Part of the module focuses on adaptive web-systems, and they are therefore required to create some such web-systems, via our set of tools, and effectively become authors of their own adaptive courses. The coursework runs during the majority of the module, and represents a long term, purpose-driven (results are marked) use of our tool. The content of the adaptive courses created by the students has to be also related to adaptive and dynamic web-based systems. Thus there are two major learning outcomes to this coursework:

1. Research into a dynamic-web related topic of their choice and the creation of Domain content.
2. Applying metadata and creating the right adaptation strategies for the personalised presentation of the material.

In 2008, for the content creation and labelling, students worked with MOT1.0 and in 2009 with MOT3.0. The 2008 class was composed of 23 students, whereas in 2009 the number grew to 34. In both years, the coursework involved the creation of (original) adaptation strategies using the LAG language. However, in 2009 the students had the support of the PEAL system, whereas in 2008 they needed to use a text editor, as the PEAL system had not been built yet. 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.

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

- Coursework 1.1 - Simple steps with the content authoring tool (MOT1.0 or MOT 3.0.1)

The students were taught how to use the tool, making sure they have completed the following steps, ordered here based on their estimated difficulty:

1. Exploring Domain Maps and Goal Maps (Browsing, Viewing, Searching);
2. Working with Domain Maps (creating, editing, changing hierarchy, coping concepts, linking concepts, defining concept relations, deleting);
3. 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);

4. Importing a Wikipedia page (MOT3.0.1 only);
5. Importing a Presentation (MOT3.0.1 only).

- Coursework 1.2: Selecting one-two topics for which to create adaptive content

The students had to select some topics they were interested in, to use as a basis for their adaptive course. In 2008, the students had one topic each, whereas in 2009 each group needed two different topics. The topics were to be related to the overall area of the course, and some suggestion list was given (including papers on this topic [20]). However, students were also encouraged to find other topics of interest, as long as they were related to the module, from the Internet or beyond.

- Coursework 1.3: Complex labelling, and usage of strategies

Students were required to perform the complete process of creating static content, labelling 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! [23], [56].

- 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, comprising 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. The students were highly encouraged to create innovative strategies, marks being awarded for (amongst other things) original strategies, and pedagogical use[28].

Other minor differences were that courseworks 1.1-1.4 were marked courseworks in 2008 and unmarked in 2009. This difference resulted from the fact that in 2008 it was considered that marking the small steps leading to the overall end product would encourage students to perform them in time. However, students justly noted that they were marked for partial work, at a time when their overall grasp of the system was not complete. Hence, whilst the steps were kept, and were obligatory work, they were not marked in 2009 anymore. Overall, however, the students had to perform similar quantities of work with the authoring systems, and present at the end four adaptive courses. Even in 2008, the bulk of their mark would be for this end-product, and not for the small initial steps. Whilst working on the coursework, students were encouraged to provide feedback on the suite of

[28] Note that whilst the students were not pedagogical experts, the students were given extra marks for an interesting and educationally coherent strategy.

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.

5.5.1 Usage of MOT

5.5.1.1 Analysis of Submitted Content

Figure 5.2 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) describe 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. This was due to a change in the specification of the module's coursework between 2008 and 2009.

For comparison, Figure 5.3 shows an example of a domain map created by one of the groups in 2009 using MOT3.0.1.

For each year, students were told to create as many domain maps as they needed to generate their adaptive courses. In 2008, students only needed to create one domain map (although they could create more if desired), and then use this domain map as a basis for up to four different Goal maps, with four separate strategies. Similarly, in 2009 students needed to create two domain maps (one for each topic), although they could optionally create more if they felt it was necessary.

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 to the example domain from 2008, in Figure 5.2), and a secondary domain map containing a series of concepts to store questions (see Figure 5.3).

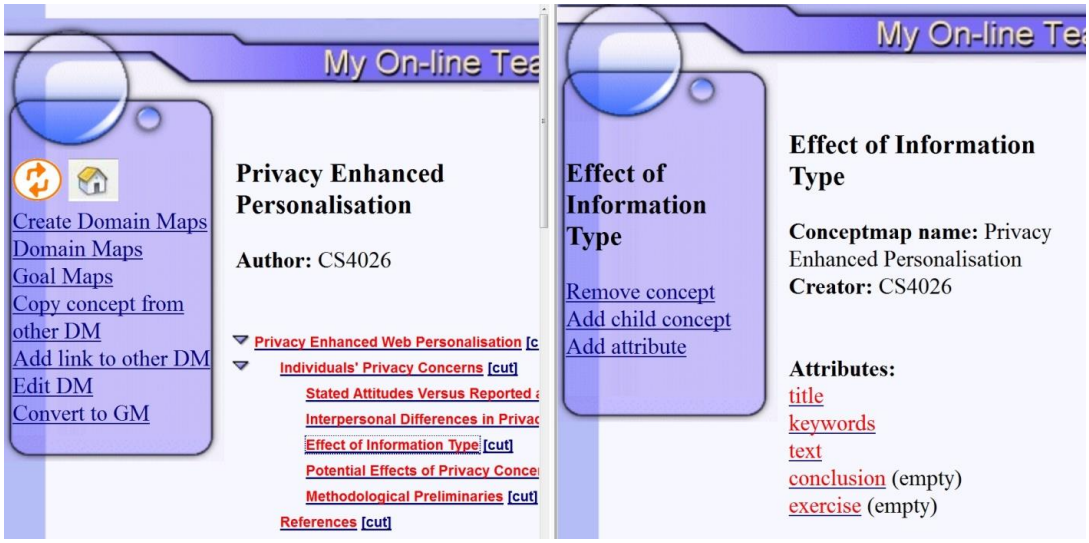

Figure 5.2: A domain map hierarchy created in 2008 using MOT1.0

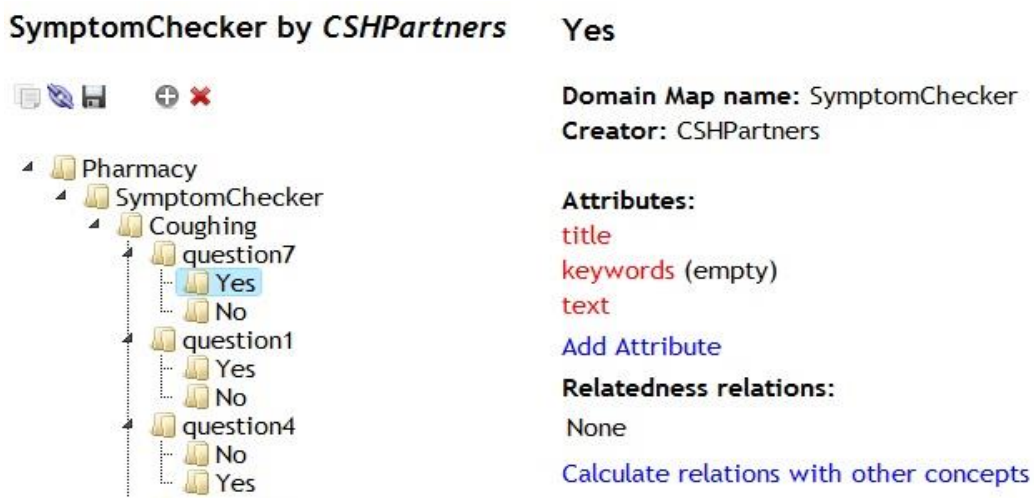

Figure 5.3: A domain map hierarchy created in 2009 using MOT3.0

Each question contained 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 suggests 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. This is further analysed in a quantitative way, as follows.

5.5.2 Quantitative Analysis of Submitted Content

Here we analyse, 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 analyse 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 due to the fact that the 2009 class was larger, that the groups in which they were divided were of smaller size, and that the assignment changed in that it asked for more domain maps (only one topic in 2008, but two topics in 2009 thus resulting in an average of 3-4 domain maps 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 was to create about 4 adaptive presentations.

5.5.2.1 Statistical Significance

For each of the results in this chapter, an *independent samples t-test* is used to test the statistical significance. Unless otherwise reported, any significance is reported at a confidence interval of 95%. The t-test makes four assumptions.

1. Normally distributed data: Since there were a number of different measures (such as numbers of attributes and concepts) tested from a number of different perspectives (as described in the next section), it was impractical to check that every dataset followed a normal distribution. However, an informal test was used to randomly select samples of the data and plot a histogram to check that the data followed an approximately normal distribution.
2. Homogeneity of variance: This means that for each metric, the variance of the samples in each year should be approximately the same. This was verified (in SPSS[6]) using Levene's test when performing the t-test, and where necessary the degrees of freedom in the t-test were adjusted in accordance to the unequal variance t-test[120].
3. Interval data: Each of the metrics used in this chapter is a number rather than a category – therefore this assumption was met.
4. Independence: The samples (group submissions) were independent from each other – therefore this assumption was met.

Where appropriate, the results discussion also reports the effect size (*r*). This is calculated using Pearson's correlation coefficient (as described by Field[7]), and follows the conventions described by Cohen[121] to determine the importance of the effect size – r > 0.1: small, r > 0.3: medium, r > 0.5: large.

5.5.2.2 Analysis of Domain Maps

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 large main domain map, with several smaller utility maps (e.g. to store 'settings' information, as explained previously). This therefore suggests that *creating domain maps* and *reusing multiple domain maps within a single course* became easier with the new toolset.

To ensure that the content from both years can be fairly compared, and to overcome the bias introduced by the smaller domain maps, it is necessary to analyse the MOT3.0 statistics from three different angles.

- *All Domain Maps:* All the submitted domain maps were combined together. Where groups submitted duplicate domain maps, the domain map was only counted once. This ensured that the work performed by a group that only submitted CAF file was counted equally with those that submitted multiple CAF files using the same domain. Where students submitted similar (but not identical) domain maps, each domain map was considered individually.
- *Best Domain Map per group:* For each group, only the most complete Domain Map was considered. This was usually the map with the most concepts, however one of the MOT3.0 (2009) groups submitted two CAF files that had a similar domain structure. One of the domains contained 19 extra concepts that were added to allow a course that provided questions and answers about the topic. In this case, the domain without these extra concepts was chosen, since it more clearly represented how the domain knowledge was structured, rather than being course specific. The average therefore only considered the best of these Domain Maps.

- *Average Domain Content per group:* All the Domain Maps created by each group were combined, grouped by the CAF file they were submitted in. Domain content that was duplicated between CAF files was therefore counted as many times as it was submitted. For instance, if a group submitted two domain maps (one for each subject) and reused them amongst four CAF files, each domain map would be counted twice. Similarly, if a group submitted a CAF file that contained two domain maps (for instance using a utility domain as described above), the CAF file's average domain content was calculated by combining the two domains together.

In all cases, where groups submitted extra files that weren't considered to be part of the assessed work, the extra files were removed from the analysis.

For MOT1.0, as the students only created one domain map per group, the values per 'All DMs', 'Best DM per group' and 'Average Domain Content per group' were identical, and thus not repeated.

	Number of Cases	***Concepts***	***Depth of Average Concept***	***Number of Attributes***	***Attributes per Concept***	***Number of Custom Attribute Types***	***Percentage Attributes (excluding title) containing more than 2 unique HTML tags***
MOT1.0: All DMs	4	22 StdDev: 12.44	2.75 StdDev: 0.31	63 StdDev: 37.28	2.8 StdDev: 0.56	1.25 StdDev: 1.89	2.43 StdDev: 4.86
MOT3.0: All DMs	29	22.24 StdDev:12.22	2.93 StdDev: 0.7	54.62 StdDev: 30.18	2.52 StdDev:0.6	1.75 StdDev: 2.79	19.4 StdDev: 21.93
MOT3.0: Best DM per group	10	27.1 StdDev: 12.3	3.06 StdDev: 0.66	65 StdDev: 32.62	2.45 StdDev: 0.58	1.6 StdDev: 2.17	24.2 StdDev: 25.97
MOT3.0: Average Domain Content per group	10	22.2 StdDev: 10.12	2.85 StdDev: 0.61	54.34 StdDev: 21.5	2.53 StdDev: 0.42	1.64 StdDev: 1.59	20.5 StdDev: 25

Table 5.1: Domain Map Statistics

Note, that the number of attributes per concept includes the title attribute (which is essential for all concepts). A custom attribute type is defined as any attribute that is not in the pre-installed list of attribute types (i.e., *title*, *keywords*, *pattern*, *text*, *explanation*, *conclusion*, *exercise* and *introduction*).

Table 5.1 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).

When considering only the best domain maps per group, there was a larger difference in the number of concepts created by each group (although not statistically significant, $t(12) = -0.699$, $p > 0.05$, $r = 0.2$). As the table shows, the *average attributes number per concept* was slightly lower in 2009, for all categories, when compared to the 2.8 attributes per concept in 2008 – this is also not statistically significant, $t(12) = 1.177$, $p > 0.05$, $r = 0.32$.

As we have seen, some groups created auxiliary domain maps, with information stored in a single attribute for each concept, which influenced the average number per concepts per domain map. Therefore, it might be more accurate to consider the best DM per group, which presents a lower number of attributes per concept (2.45 in 2009, also not statistically significant). One added functionality was that of importing content from Wikipedia, which generated a lot of content (as reflected in the increase of concepts created by groups in the table), however produced only a low number of attributes per concept (two: text and title).

Even if students have added additional attributes the size of the maps may have been too large to add a significant number of supplementary attributes. To alleviate these difficulties, future iterations of the MOT tool will allow for automatic addition of user defined attributes to the standard attribute set (see Chapter 10). 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 by Hendrix and Cristea [89] or the semi-automatic principles described in Chapter 4.

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 when considering the *best DM per group* (although not statistically significant, $t(12) = -0.1$, $p > 0.05$, $r = 0.03$). This increase is in accordance with the fact that the number of concepts also increased, thus showing that the information content in the lessons designed by the students has increased. This supports 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. Considering also that the groups were of fewer students in 2009 when compared to 2008, this statement is further strengthened.

As the table also illustrates, the *depth of the average concept* in 2009 was 3.06 (for the best DM per group), slightly deeper than the average concept created in 2008 (2.75), showing that the domain maps created using MOT3.0 had a more detailed structure.

There was also a very minor (statistically insignificant) increase in the number of custom attribute types that were used. Similarly, only one group utilised the relatedness links feature. Moreover, this group only created one such relation. Similarly, none of the groups in 2008 had created any relatedness relations with MOT1.0. Part of the reason for this could be because the support for such relations in LAG did not provide much flexibility (this was later improved – see Chapter 10). Alternatively, it could be because the interface for the creation of such relations

was insufficient. The support for the creation of such relationships was therefore changed in future versions of MOT (see Chapter 7 and Chapter 10).

Most of the content in MOT1.0 was written in plaintext. However, some attributes contained a few HTML tags when formatting was essential. For instance, some attributes had no formatting within the text, but occasionally added an <*img*> or a <*p*> tag. For this reason, it was decided to investigate the number of attributes that contained more than two distinct HTML tags, since those attributes will have a significantly higher variety of styles. Also, this figure excluded any title attributes, since AHA! does not support HTML within title attributes. The percentage of attributes that used more than two HTML tags in MOT3.0 was significantly higher than with MOT1.0, ($t(10.406) = -2.543$, $p < 0.05$, $r = 0.62$). This shows that the WYSIWYG editor did encourage users to use HTML to describe their content.

5.5.2.3 Goal Maps

A similar analysis was applied to the *goal maps* created by the two classes of 2008 and 2009. As with the domain map analysis, averages were calculated instead of computing the overall productivity, due to the difference in student numbers. Table 5.2 shows some selected results of the analysis.

As with the Domain Map analysis, because each group (and particularly, each year group) was allowed to submit a different number of Goal and Domain Maps in order to create their four adaptive courses, this analysis can be investigated from a number of different aspects.

- *All Goal Maps*: The averages shown in Table 5.2 are based on all the submitted CAF files. Due to the change in coursework specification, there were 41 CAF files submitted in 2009 (one group submitted 5 CAF files), and 13 CAF files in 2008 (one group submitted only 1 CAF file).
- *Best GM per group*: The averages shown are based only on the best Goal Map for each group. The '*best*' in this case is defined as the goal map with the highest number of sublessons. In the event that multiple goal maps had the same number of sublessons, the goal map with the highest number of labelled sublessons was chosen.
- *Average GM per group*: An average goal map was created for each group. The results shown in Table 5.2 are calculated by averaging the average score for each group.

	Number of Cases	*Number of Distinct Labels*	*Number of Sublessons*	*Percentage of Labelled Sublessons*	*Percentage of Weighted Sublessons (i.e., labels with weight other than 0)*
2008 (MOT1.0): All GMs	13	0.92 (SD: 0.862)	118.31 (SD: 60.76)	12.58 (SD: 28.32)	10.35 (SD: 28.58)
2009 (MOT3 0): All GMs	40	3.83 (SD: 3.09)	62.38 (SD: 35.38)	53.25 (SD: 34.73)	37.29 (SD: 34.73)
2008 (MOT1.0): Best GM per group	4	1.75 (SD: 0.975)	101 (SD: 73.63)	38.85 (SD: 43.34)	33.65 (SD: 47.14)
2009 (MOT3.0): Best GM per group	10	4.7 (SD: 3.9)	96.4 (SD: 41.31)	54.89 (SD: 33.16)	42.3 (SD: 31.91)
2008 (MOT1.0): Average GM per group	4	1.13 (SD: 0.72)	101 (SD: 73.63)	17.43 (SD: 17.80)	14.9 (SD: 17.65)
2009 (MOT3.0): Average GM per group	10	3.83 (SD: 2.26)	62.38 (SD: 21.03)	52.29 (SD: 21.39)	38.94 (SD: 24.02)

Table 5.2: Goal Map Production in both MOT1.0 and MOT3.0

Both MOT1.0 and MOT3.0 supported the labelling of both 'groups of sublessons' (shown in Figure 5.4 with a folder icon) and individual sublessons (shown with a file icon). However, the labelling of sublesson group headings in this way has no semantic meaning, since in LAG weights and labels can only be interpreted at the level of an individual sublesson. In the case of Figure 5.4, the 'history, 5' label has no semantic meaning within AHA! or ADE, whereas the 'history, 0' labels can be interpreted. For this reason, this analysis considers only the weights and labels applied to individual sublessons.

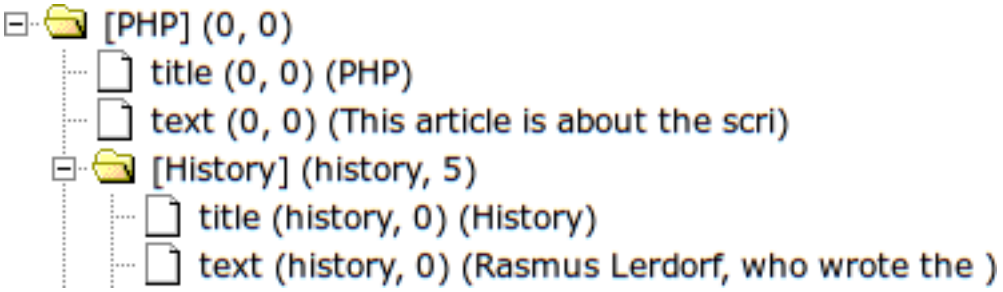

Figure 5.4: Sublessons and 'Groups of Sublessons' in MOT3.0 – extract based on [106]

Because of the significant difference in the number of CAF files between the years (13 in 2008, 41 in 2009), the '*All Goal Maps*' group shows a significantly different variance between the two year groups (verified using Levene's test). This makes it difficult to show any statistical difference when considering '*All Goal Maps*'.

The percentage of labelled sublessons was higher for the MOT3.0 group, and when considering the '*Average GM per group*' metric, can be shown to be statistically significant with a large effect size ($t(12) = -2.867$, $p < 0.05$, $r = 0.64$).

Similarly, the number of distinct labels is higher for MOT3.0. This suggests that the strategies created used a higher number of classifications (this is further explored in the analysis of the strategies in Section 5.5.4.1). Moreover, when considering the

'*Average GMs per group'* this has a statistical significance and a large effect size ($t(12) = -2.295$, $p < 0.05$, $r = 0.55$).

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 MOT3.0.1, whereas this 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. This reduction in the number of sublessons is not statistically significant (using either the '*Best GM per group*' or '*Average GMs per group*' classification).

5.5.3 Qualitative Feedback

The students provided a lot of feedback during their 8 weeks of exposure to the tools. A spiral model for software development [5] 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 (in the MOT3.0.1 implementation, a user 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 – the importer was improved in response to this feedback (see Chapter 4). Overall, from all discussions, we can say that the students understood the ideas behind authoring content.

5.5.4 Usage of PEAL

This section continues to compare the coursework submitted in 2009 with that of 2008 by investigating the submitted LAG strategies. In 2009, students had the support of the PEAL LAG editor, whereas in 2008 the students could only use a text editor.

5.5.4.1 Types of strategy created

Overall, the students in 2008 implemented only very slightly changed strategies based on the pool of strategies we provided for them. Many students opted for modifying 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 labelled concepts before they would usually be shown by such a strategy.

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 a larger number 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 the user 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 5.5 shows a screenshot of PEAL, whilst editing a LAG strategy, as created by one of the student groups of 2009.

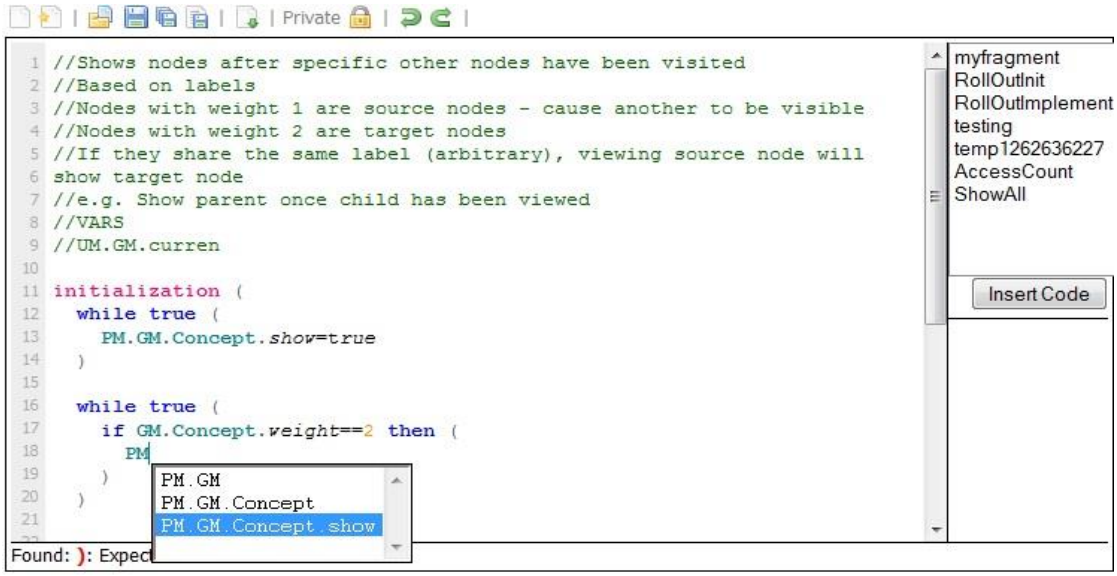

Figure 5.5: Editing a LAG Strategy using PEAL

5.5.4.2 Quantitative Analysis of Submitted Strategies

Whilst all groups were expected to submit four strategies, there were a number of instances where some groups submitted fewer than four strategies. For this reason – as with the Domain and Goal map content – the submitted LAG strategies were analysed from three different perspectives.

- *All LAG strategies*: Every LAG file that was submitted by the students was considered.
- *Best LAG strategy per group*: Only the best LAG file submitted by each group was considered. In this case, the 'best' was defined as the file with the most characters of code.
- *Average LAG strategy per group*: For each group, an average of each measure was taken.

The analysis also considers three different content types.

1. *Code*: Valid LAG code.

2. *Description*: Comments that occur before the *initialization* LAG keyword. These are typically used to describe what a strategy does, what variables it uses, and what labels it requires.
3. *Comments*: Comments (denoted by a '//'). Any line that contains both code and a comment was counted as both a code line and a comments line.

Note, one group didn't actually submit any strategies – only a CAF file (as reported above).

	N	*Lines of Code*	*Lines of Description*	*Lines of Comments*	*Characters of Code*	*Characters of Description*	*Characters of Comments*
Text editor (2008) All Strategies	12	21.17 SD: 8.97	7.25 SD: 5.63	2.83 SD: 2.44	334.75 SD: 188.23	263.17 SD: 283.02	133.42 SD: 122.15
PEAL (2009) All Strategies	38	46.18 SD: 25.65	8.66 SD: 6.78	6.53 SD: 9.92	957.79 SD: 613.274	493.45 SD: 599.37	391.61 SD: 680.928
Text editor (2008) Best Strategy per Group	3	32 SD: 3.606	9.33 SD: 1.16	4 SD: 4.58	599 SD: 89.44	498.33 SD: 245.28	224 SD: 232.41
PEAL (2009) Best Strategy per Group	10	66.4 SD: 22.76	10 SD: 7.5	12.1 SD: 17.15	1457.9 SD: 685.54	680.8 SD: 788.88	757.5 SD: 1182.25
Text editor (2008) Average Strategy per Group	3	21.17 SD: 4.26	7.25 SD: 3.13	2.83 SD: 1.04	334.75 SD: 51.75	263.16 SD: 67.63	133.42 SD: 35.63
PEAL (2009) Average Strategy per Group	10	45.7 SD: 17.02	8.53 SD: 5.51	6.2 SD: 7.04	950.55 SD: 456.59	481.83 SD: 521.36	372.03 SD: 517.02

Table 5.3: Strategies created by 2008 and 2009 Cohorts

The statistics shown in Table 5.3 show that in 2009 there was an increase in the number of lines (and characters) used for strategy *Descriptions*, *Comments* and *Code*. Moreover, Table 5.4 shows the aspects of the analysis that are statistically significant at a 95% confidence level.

	Degrees of Freedom	***t***	***r (Effect Size)***
Best Strategy Per Group: Lines of Code	11	-2.534	0.61 (Large)
Best Strategy Per Group: Characters of Code	11	-2.103	0.54 (Large)
Average Strategy Per Group: Lines of Code	11	-2.404	0.59 (Large)
Average Strategy Per Group: Characters of Code	11	-2.262	0.56 (Large)
All Strategies: Lines of Code	47.328	-5.104	0.6 (Large)
All Strategies: Characters of Code	48	-5.496	0.62 (Large)
All Strategies: Characters of Comments	43.409	-2.227	0.32 (Medium)

Table 5.4: T-Test statistics for LAG Strategies

These t-test results show that there is statistical evidence that users create strategies that contain more code when using PEAL. Similarly, there is some statistical evidence to show that more comments are added, however the effect is slightly smaller.

5.5.5 Discussion

Both the quantitative analysis and a survey of the types of strategies submitted has suggested an improvement in the quality of the strategies submitted in 2009 over those submitted in 2008.

The quantitative analysis showed that there was significantly more code (in terms of number of lines and characters) in the strategies that had the support of PEAL. Whilst the amount of code is not necessarily an indicator of the pedagogical utility of the strategy, it does show that PEAL can support the user in the creation of such complex strategies.

There is some indication that the number of comments also increased with the usage of PEAL, however this can only be shown to be statistically significant when considering all strategies (rather than the *best strategy per group*, or the *average strategy per group*), and only shows a medium effect size. This increase in the number of comments could also be directly related to the increase in the amount of code.

Overall, there appears to be a clear improvement in the quality of the strategies when compared with 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! – this procedure was radically overhauled with the introduction of ADE [50] (see Chapter 7).

5.6 Future Work

The evaluation provided a number of suggestions for future improvement within PEAL. Specifically, greater accuracy of the warning messages was needed (using line numbers to highlight the syntax error), and streamlining the process of previewing changes. This could be achieved by more tightly integrating the MOT and PEAL tools, therefore allowing content to be previewed with a particular strategy. Moreover, work had started at this stage towards 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. This new version of PEAL is described in Chapter 7 .

5.7 Conclusion

This chapter has defined a set of *adaptation authoring imperatives*, extracted from previous experience in design, implementation and deployment of authoring for adaptation, from related research, and the initial set of requirements that were created at the start of this research. Based on this, this chapter 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 improvements 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 can 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. This is unlike much of the related research, which often deliver only tools that can be used for proof of concept, and don't attempt the more difficult and risky process of long term use.

For comprehensive, long-term evaluations with iterative development, these types of evaluations are useful, and have highlighted a number of usability issues that needed to be addressed. Using final year students to evaluate authoring tools is helpful because much of their work consists of gathering and presenting material. This is a similar process to many of the educators that are the ultimate end-users of the adaptive authoring tools. Chapter 6 discusses how these tools can be used by such educators.

6 From a linear module to an adaptive course

6.1 Overview

Over the previous three chapters, we have built and described an enhanced adaptive hypermedia authoring system MOT3.0. The main focus of this effort was on adding and extending the type of functionality that allows the 'lay person', the non-technical author, to efficiently use such a tool. In this chapter we show how this can be achieved. We show in a realistic case, that teachers can start from any course they are already teaching, and transform it, in a number of steps, into an adaptive course, thus targeting various learners and moving away from the 'one-size-fits-all' approach. We then discuss how this apparent simplicity still permits for the building of flexible and complex adaptation, and finally present evaluation results with designers and authors of the tool.

6.2 Scenarios

This chapter considers the authoring process from the point of view of two types of authoring, as illustrated by the two scenarios below.

6.2.1 Content Authoring

Professor Smith is a lecturer in Computer Science, and has presented a ‘Web Development’ course for the last five years. The resources she currently uses are: 30 lecture presentations (written in PowerPoint); 5 videos (each 5 minutes long) and 1 online quiz (authored in Moodle). Although the Professor is keen to embrace the advantages of adaptive hypermedia, she does not want to spend a long time rewriting all of her course material. Nor does she wish to learn a new programming language. Thus she uses the MOT3.0 tool, which will allow her to structure her

existing content in a way that can be integrated into an adaptive course. Her students have previously taken an ILS (Index of Learning Styles) test [122] and have shown clear preferences for two types of learning styles: some of her students are *visual*, some *verbal*. She would also like to classify her students into *beginner*, *intermediate* and *advanced* groups.

Professor Smith selects two adaptation strategies from a pool of strategies (created by her colleague, Professor Jones) that cater for the two types of adaptivity she is envisioning. From the natural language description of the strategies, without reading the code she finds out what type of labelling and annotation she needs to add to the material she has imported into MOT3.0. Because the content has been automatically separated into many reusable pieces, she finds the annotation process simple and fast. Finally, she applies the adaptation strategies to her content and deploys the result in the adaptation engine which will display it to her students, in a personalised way.

6.2.2 Adaptation Authoring

Professor Jones is another Computer Science lecturer, and a colleague of Professor Smith. He understands the pedagogical benefits of adaptive hypermedia, and has recently learnt the syntax of the LAG [53] adaptation programming language. Professor Jones has been appointed by his department to create a pool of adaptation strategies that will be used by his colleagues. He has both pedagogical knowledge and programming knowledge.

However, Professor Jones has not yet had much experience of authoring LAG adaptation files. The web-based PEAL [53] editor will assist Professor Jones,

providing syntax highlighting and code completion. He then creates a good number of relevant strategies in a relatively short amount of time. Importantly, he adds good natural language descriptions to each of the strategies, so that his colleagues may use them without needing to read any of his code.

In the following sections, we will explain, from a technical point of view, how Professor Smith and Professor Jones can collaborate on an adaptive course, utilizing the two scenarios above.

6.3 Importing the Linear Content in MOT3.0

6.3.1 Importing Presentation Slides

Professor Smith starts by using MOT3.0's presentation *importer* to upload one of her existing PowerPoint files on "PHP" to the MOT server. The import script automatically analyses the presentation content, and creates a new domain structure (called a *domain map*) to store her lecture (see Figure 6.1). As adaptation means conditionally displaying or removing content fragments, depending on the learner's needs, the first task for the system is to separate the existing content into reusable fragments (called *attributes*).

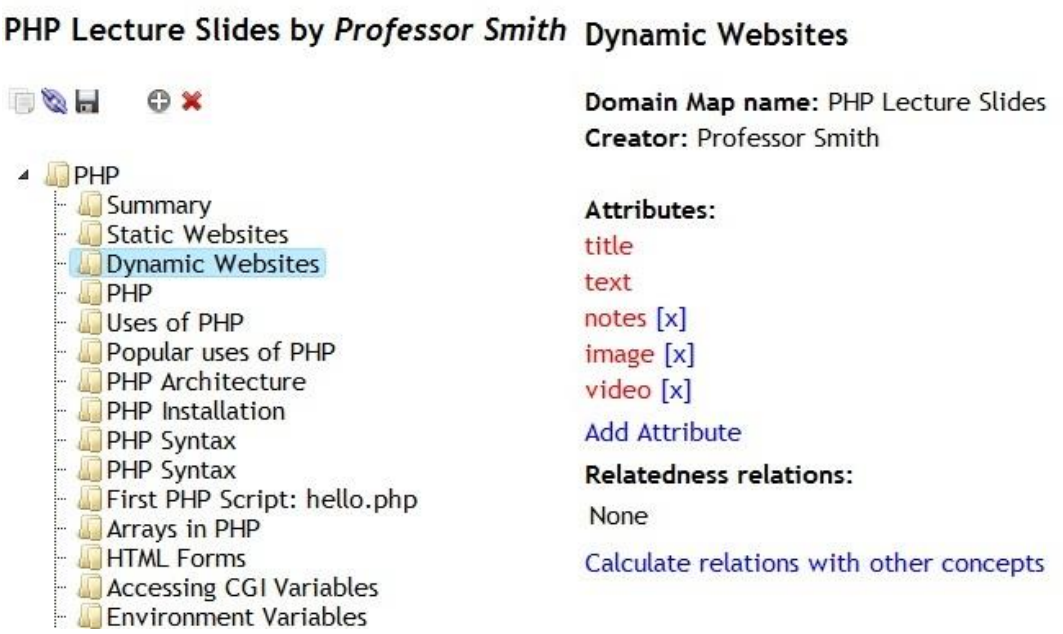

Figure 6.1: Domain model of an imported presentation

Concretely, for each slide in her presentation, the import script creates a new concept in the domain map hierarchy (Figure 6.1, left side), and a number of attributes assigned to this concept (Figure 6.1, right side). The importer generates a slide *image*, and also automates OpenOffice[29] to export an HTML representation of the slide. From the latter, MOT3.0 extracts the *title* of the slide, the *text* content, and Professor Smith's slide *notes*. These attributes are the various information representations for each slide, and thus ensure various adaptations (e.g., slide notes can be used to create an overview; and titles can be used to generate a 'Table of Contents'). For more details about this process, see Chapter 4 .

The actual strategies she will be using are created by someone else, Professor Jones, and will be introduced in Section 6.4.

[29] http://www.openoffice.org

6.3.2 Importing Wikipedia Content

Professor Smith is keen to enhance her lectures by providing information about related topics from Wikipedia[30]. She is aware of the issues surrounding the reliability of Wikipedia content, however, she would like her students to be able to read about the module topics from other sources. She simply types the name of a Wikipedia article (here, "PHP") into MOT3.0's Wikipedia importer, which then downloads the WikiText source code of the article.

As described in Chapter 4 , headings in WikiText are denoted by placing '=' signs on both sides of the heading text. The number of '=' signs denotes the level of the heading (e.g. 2 signs for a level 1 heading, 3 signs for a level 2 heading etc.), which allows the import script to automatically divide the article's content into sections, thus inferring the structure of the article. For each section of the article, the script generates a concept in the *domain map* (Figure 6.2, left side). Each concept is automatically assigned two attributes – the *title* of the section, and the *text* of the section (converted to HTML).

In this way, the import clearly generates a number of reusable, separate concepts that correspond to the many different headings in the original article. Additionally, these concepts are grouped in hierarchies, each with at least two attributes. All these will constitute the alternatives that will be available to the adaptation strategies Professor Smith will apply. As with the previous domain map, she is able to add more content to the newly created domain map.

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

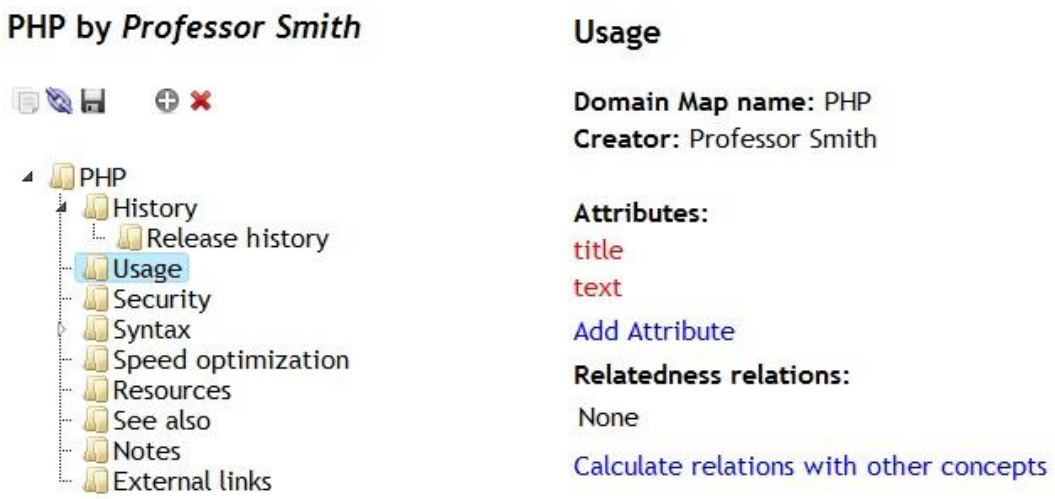

Figure 6.2: Domain model of the imported Wikipedia PHP article [106]

6.3.3 Importing Moodle Content

Another import script Professor Smith can use concerns content from other Learning Management Systems, such as Moodle or Sakai. Professor Smith has already created an online quiz using Moodle, so she exports this content to an IMS-QTI file. She can then upload the IMS-QTI file to MOT3.0, where her content will be converted into another domain map. For each question in the quiz, a concept is created. Each of these concepts contains a *question* attribute, and an *answer* attribute. These questions and answers can be used within an adaptive course. For instance, it would be simple to create an adaptive course that hides all answers until the user has read all the questions. Moreover, the imported questions could be combined with a goal map from another course to create a course that contains both Wiki text content and the questions/answers from the IMS-QTI file. This would allow Professor Smith to create a course that (for example) shows revision material to users after they have viewed both the question and answer attributes for a particular topic.

6.3.4 Editing and Enriching the imported content

The extracted format from the above importers allows Professor Smith also to add additional information to her module, either from HTML content stored previously on MOT3.0 - by simply copying a concept across from a previously authored domain map; or from additional material – e.g., she can add a link to the YouTube[31] videos she was using in her class. She does this by first creating another attribute for the concept 'PHP', and then uses the HTML editor to add some custom HTML. This custom HTML can be easily generated from YouTube by clicking on the '*Embed*' button next to the video, then copying and pasting the generated code into the HTML editor[123].

6.4 Creating Adaptation Strategies

Professor Jones uses PEAL to create a series of pedagogical adaptation strategies. He is confident with simple programming, and creates strategies that can be used by his colleagues.

6.4.1 Beginner-Intermediate-Advanced Strategy

One strategy he creates divides students into 3 groups: *beginner*, *intermediate* and *advanced*, hiding content from learners until they have reached the appropriate level (Figure 6.3 shows an editing snapshot). PEAL suggests *automatic completion* for the current program line (pop-up window). The available library *code fragments,* which can be inserted directly into the current code, appear in the right frame. Figure 6.3 also shows *colour formatting* and *recognition of programming instructions*, as well as *code line numbers* – which help Professor Jones to program

[31] http://www.youtube.com

in the LAG adaptation language. Additionally, PEAL gives access to previously stored strategies (created by someone else and marked for sharing), allows parts of programs to be created directly via a Wizard, and thus overall represents a simple way for Professor Jones to accomplish his task in a short amount of time.

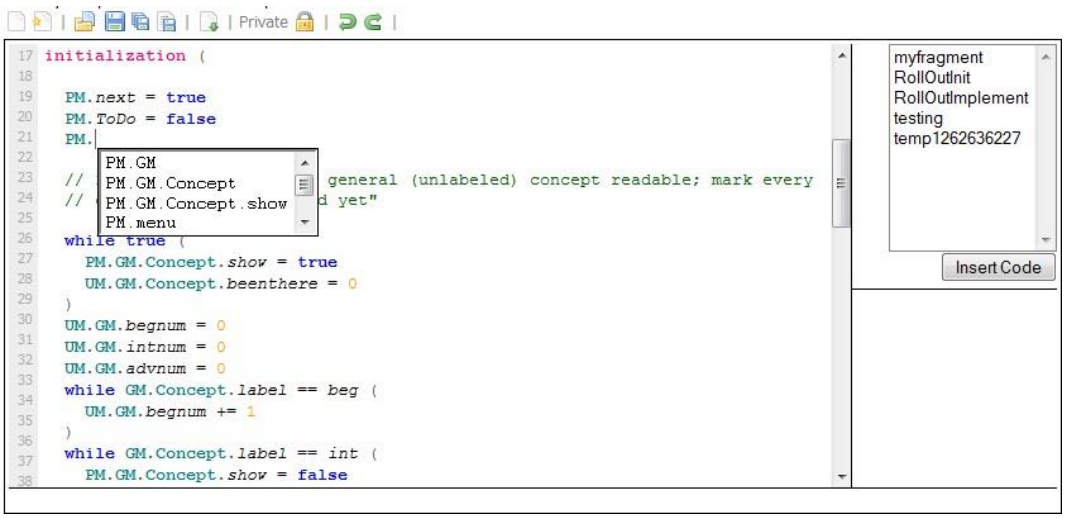

Figure 6.3: Editing a strategy in PEAL

6.4.2 Visual-Verbal Strategy

Another pedagogical strategy Professor Jones creates differentiates between students who are visual learners and those who prefer text.

He uses the wizard to define a variable representing if the user prefers visual or verbal content, named UM.GM.visverb, see Figure 6.4.

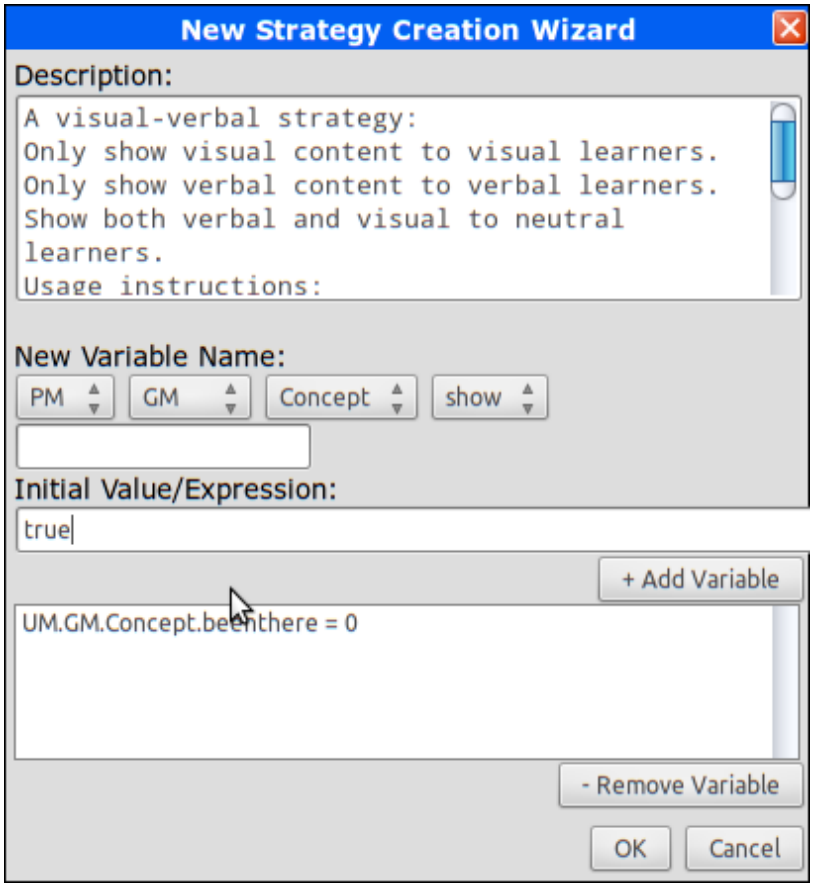

Figure 6.4: The strategy creation wizard in PEAL

This strategy requires pieces of content to be labelled 'visverb', and given a weight to indicate whether the content is visual or verbal. The user's preference variable is compared with the weight of the content, and if the result is above a predefined threshold, the content is shown.

When Professor Jones has completed his strategies, he publishes them on the university website. The descriptions of the adaptation strategies that he entered using the wizard (see Figure 6.4) contain usage instructions, informing any content creators of what labels need to be applied to the content to make the strategy work.

6.5 Combining and Enhancing Linear Input

6.5.1 Adding adaptive behaviour to linear content

After importing and enriching her existing lecture presentations to domain maps, as described in Section 6.3.4 , Professor Smith can now export her content into a pedagogical *goal map*, by clicking on an icon in MOT3.0. A goal map allows her to add pedagogical labels and weights (Figure 6.5, right side), according to the adaptation strategy that she will be employing.

She could convert the same domain map to many different goal maps and add different labels, thus reusing the same content for different pedagogical personalisation strategies. However, she decides to create only one lesson for now. The goal model environment also allows her to combine content from different domain maps. She uses this to add information from her Wikipedia domain map.

Professor Smith browses the variety of existing pedagogical strategies on the university website, and chooses the visual-verbal strategy that was created by Professor Jones (and described in Section 2.6.2.3). She then labels the image version of the slide as ‘visverb’, and gives it a weight of 30 (representing visual content) and the text version of the slide as ‘visverb’ with a weight of 70 (for verbal content). She knows the semantic and pedagogical meaning of these labels because she read the description of the pedagogical adaptation strategy (from the first few lines of the LAG file). These labels and weights correspond to the ones prescribed by the ‘Visual-Verbal’ strategy created by Professor Jones.

The MOT3.0 system allows her to apply the same label and weight to many goal model concepts at once, thus saving her time, as most of her material is either of a visual or of a verbal nature.

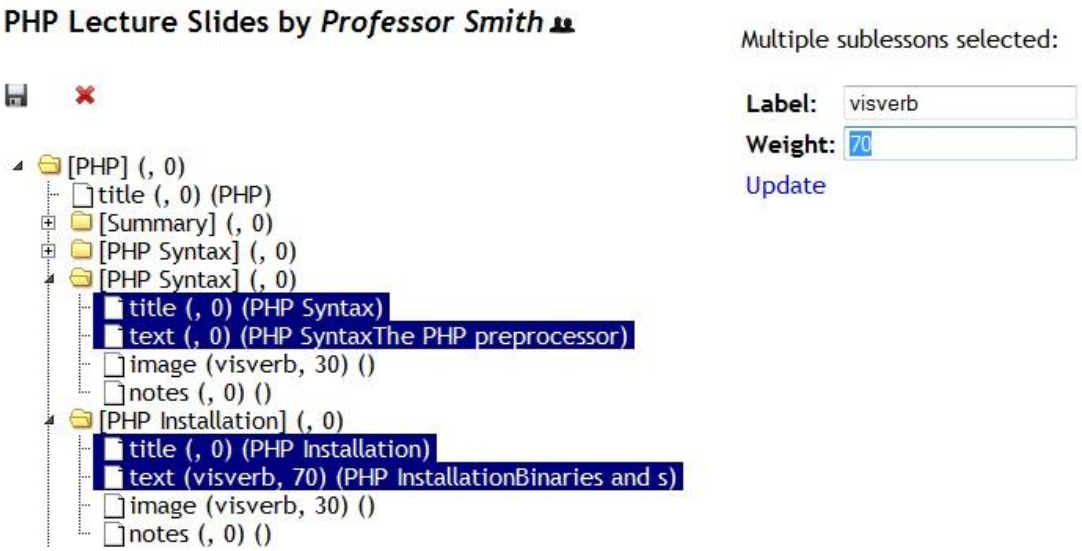

Figure 6.5: Goal model of the imported presentation

6.5.2 Delivering adaptive courses

Professor Smith can now export her pedagogical goal map (as a CAF file), and upload it to the delivery tool (such as ADE [50] or AHA![23]), together with one of Professor Jones's LAG pedagogical adaptation strategy files. The delivery tool then creates a course which combines the educational content with the adaptation strategy. Professor Smith's students can then use the adaptive course.

6.6 Evaluation and Discussion

Chapter 4 described an evaluation that was performed at the University of Warwick with six volunteer course authors and designers. Each participant was experienced in the process of both teaching and web-design, and had experience with adaptive hypermedia. This was a qualitative study, where the opinion of experts was elicited, thus the number of participants is not essential. They were asked to explore the

system, and answer 45 questions, which we then grouped into 10 categories of basic functions, as below:

1. *… browsing other author's materials*
2. *… editing with MOT3.0*
3. *… changing hierarchies of material via drag & drop*
4. *… copying and linking functionality*
5. *… editing HTML using the editor*
6. *… importing Wikipedia content*
7. *… importing Presentation content*
8. *… functionality of importing content*
9. *… authoring for adaptation support*
10. *… Semi-Automatically Creating and Linking Content for adaptation*

Figure 6.6 shows that the designers found most of the basic functions 'Easy' (or 'Very Easy') to use. To establish the statistical significance of these results, we have mapped the answers {'Very Easy', 'Easy', 'Difficult', 'Very Difficult'} onto the values {2, 1,-1, -2}. This assumes equidistance between these labelled values, as well as monotonicity, an assumption which is widely used in literature, and also conforms to the natural language use of these words.

We have then applied a one-sample T-test to compare the answers against the average of 0, corresponding to 'Neither Easy nor Difficult', to establish if the positive average is statistically significant.

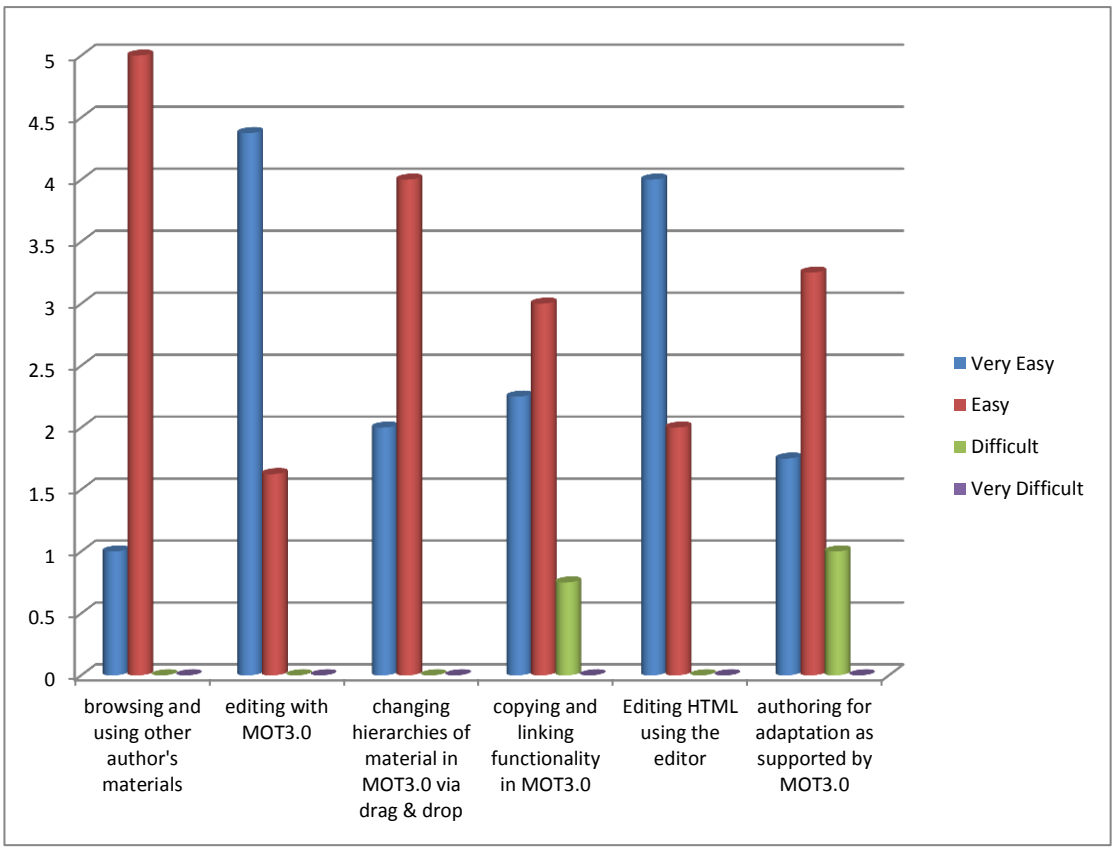

Figure 6.6: Histogram display of the responses to the MOT3.0 questionnaire

Table 6.1 shows that most answers are statistically significant with 95% confidence ($P<0.05$). Thus *browsing*, *editing, changing hierarchies* (Q1,2,3), and *editing HTML* (Q5) are significantly easy. Also (*semi-)automatically creating content and linking* are significantly useful.

Question	*Mean*	*StdDev*	*T*	*P*
1. browsing other author's materials	1.167	0.408	7	0.001
2. editing with MOT3.0	1.667	0.516	7.91	0.001
3. changing hierarchies of material via drag & drop	1.333	0.516	6.32	0.001
4. copying and linking functionality	1	1.095	2.24	0.076
5. editing HTML using the editor	1.667	0.516	7.91	0.001
6. importing Wikipedia content	1.833	0.408	11	0
7. importing Presentation content	1.833	0.408	11	0
8. functionality of importing content	1	1.095	2.24	0.076
9. authoring for adaptation support	1	1.095	2.24	0.076
10. Semi-Automatically Creating and Linking Content for adaptation	1.5	0.548	6.71	0.001

Table 6.1: The MOT3.0 questions and response frequencies

The *copying and linking functionality* (Q4), *importing content* (Q8) and *general authoring* (Q9) are only statistically significant with 90% confidence. We have thus further analysed the (sub-)questions that formed these categories, in order to establish the reasons for the lower confidence intervals. Below we show the one sample T-test values for question Q4, which was formed of three sub-questions:

Q4a: Dragging domain concepts between trees when copying/linking;

Q4b: Inserting goal map sublessons from domain concept attributes;

Q4c: Inserting other goal map lessons as sublessons.

Question	***Mean***	***StdDev***	***T***	***p***
Q4a	1.5	0.548	6.71	0.001
Q4b	0.5	1.225	1	**0.363**
Q4c	0.833	0.983	2.08	**0.093**

Table 6.2: One-Sample t-test Q4a, Q4b, Q4c (Test of mu = 0 vs not = 0)

Table 6.2 shows that whilst *dragging domain concepts* is significantly easy, *inserting goal maps from domain concept attributes* or *other sublessons* is not. Looking at the qualitative comments, the experts noted that: "*Inserting […] domain map attributes needs improvement […] partial goalmaps cannot be inserted*" and "*it is easy, but a bit inconsistent: for GM you have to click add, for DM you have to drag & drop. I would like it not to refresh back, as I may want to add more than 1 attribute*".

The questions including question 8 have already been presented and discussed in Chapter 6.

Question Q9 on *general authoring* was composed of questions on the issues of:

Q9a: Being able to create adaptive presentations with MOT3.0 (as compared with programming adaptation from scratch);

Q9b: Being able to (semi) automatically create content for adaptation;

Q9c: Being able to (semi) automatically link content for adaptation;

Q9d: Using graphical drag & drop interfaces in authoring for adaptation.

The one sample T-test results for Q9 are shown in Table 6.3. Thus, *creating adaptive presentations* with MOT3.0 is preferred (in a statistically significant way) to programming adaptation from scratch, and also *using graphical drag & drop interfaces* in authoring for adaptation is considered beneficial. Looking at why the experts are not convinced about (*semi)automatically creating and linking content*, the comments were as follows: "*The physical manipulation is easy, but you have to understand what you are doing*", "*Linking automatically is only possible in a hierarchical way. It would be interesting to see different types of automatic linking.*"

Question	***Mean***	***StdDev***	***T***	***p***
Q9a	1.167	0.408	7	0.001
Q9b	0.667	1.366	1.2	**0.286**
Q9c	0.5	1.225	1	**0.363**
Q9d	1.5	0.548	6.71	0.001

Table 6.3: Responses to question 9

Thus, whilst clearly some improvements can be done (and the experts have given us some very good pointers towards this), the overall evaluation shows that people like our imaginary Professors Smith and Jones can expect to be able to author with a reasonable degree of ease personalised courseware with a system such as MOT3.0.

6.7 Conclusions

This chapter has documented and evaluated the process that will allow educators to create adaptive courses from some of their existing resources. Specifically, we have introduced methods of generating domain models based on presentation slides and Wikipedia articles, and discussed how such a process could be used by real-world educators. It is hoped that authoring systems with import facilities such as those provided by MOT3.0 will encourage more educators – from a wide variety of subject areas – to author for adaptive hypermedia.

Future work could add other popular web-services, for instance the HTML editor could be extended to allow users to quickly find and add YouTube videos without the author needing to work with the HTML code directly (as described in Section 6.3.4). Moreover, this embedded HTML process could equally apply to many other web-services that also provide template fragments such as Twitter[32].

[32] https://dev.twitter.com/docs/twitter-for-websites

7 Increasing flexibility in authoring for adaptation: Iterative development of MOT3.1

7.1 Overview

This chapter describes the creation of MOT3.1, which builds on the feedback gathered from the previous chapters. The chapter also introduces two new tools, ADE and PEAL2, and discusses how they contribute towards an updated version of the MOT toolset. Finally, an evaluation of this new toolset is presented.

7.2 Adaptive Delivery Engine

To further explore the possibilities for providing adaptation, another PhD student at the University of Warwick's Computer Science Department, Joshua Scotton, started the development of ADE [50]. This meant that rather than relying on the opportunities provided by the MOT-to-AHA! converter [56], new constructs could be added to the LAG grammar [53], further enhancing the possibilities available for adaptation. Accordingly, the MOT toolset was now extended to include ADE.

7.3 Changes from MOT3.0 to MOT3.1

In response to feedback from the evaluations described in Chapter 5, MOT3.1 is a major rewrite of MOT3.0, featuring the following features.

7.3.1 MySQL Improvements

Whereas for compatibility purposes MOT3.0 used the same database schema as MOT1.0, MOT3.1 aimed to improve the efficiency and correctness of the database. The main change was the introduction of foreign keys (which were missing from previous versions of MOT), allowing the database to perform basic error checking

on all database updates. Specifically, when domain maps where deleted from the system, any goal map sublessons that relied on attributes from such a domain map had to be deleted by a PHP script that searched through every sublesson. Using foreign keys in MOT3.1 allows this process to be performed by the MySQL process[33], and is therefore more efficient whilst helping to ensure that the database is consistent.

7.3.2 jQuery Frontend

To implement the above changes, it was decided to use the jQuery[34] library as a framework for JavaScript. Specifically, the jQueryUI[35] and *jQueryUI Layout Plug-in*[36] were utilised. This allowed a major change to the layout of MOT3.1, providing a more coherent (and smooth) AJAX-based [82] interface, as will be explained in the following. The MOT3.1 layout divides the screen into 3 separate resizable areas (shown in Figure 7.1). It should be noted that although this functionality appears similar to the frame-based layout of MOT1.0, the utilization of AJAX within MOT3.1 ensures that content is loaded much more smoothly (and requires significantly less data to be transferred) than forcing the whole frame to be reloaded by every request (as in the frames used by MOT1.0).

[33] http://www.mysql.com

[34] http://jquery.com/

[35] http://jqueryui.com/

[36] http://layout.jquery-dev.net/documentation.cfm

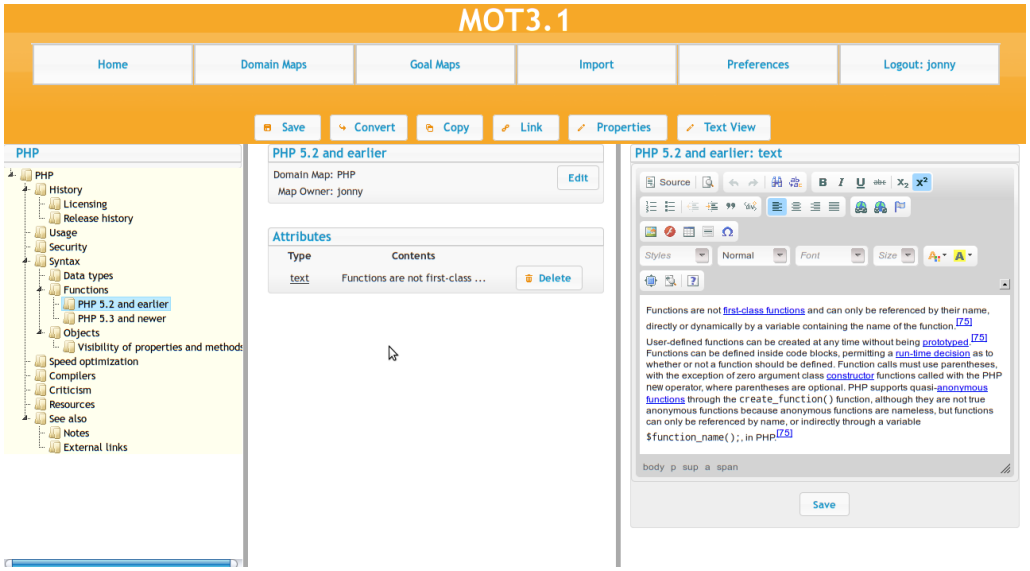

Figure 7.1: The layout of MOT3.1

The three frames allow the author to see the concept hierarchy, the attributes within the selected concept and the currently selected within the same screen.

MOT3.1 replaces the icons of MOT3.0 with a context-sensitive toolbar. The main toolbar is similar to the toolbar from MOT3.0, providing access to the main functions of MOT3.1 (*Home*, *Domain Maps*, *Goal Maps*, *Import*, *Preferences* and *Logout*). However, a secondary toolbar changes according to whether the user is currently viewing a Domain Map or a Goal Map. When editing a domain map, the toolbars in MOT3.1 adaptively change depending on whether the author is editing a single concept, or the whole domain map. This attempts to reduce the cognitive load by only providing options that are directly relevant to the current task.

MOT3.0 (and therefore MOT3.0.1) had exhibited a problem that is common with many AJAX applications whereby changes in state (such as loading a new domain map) do not register a new item on the web-browser's history [124]. This means

that the 'Back' button within the web-browser would navigate the user to the previous known page (which could be an entirely different website). This behaviour is counter-intuitive to many other web applications, and is unforgiving to the user since navigation mistakes are irreversible (this has parallels to the software design principles surrounding the undo feature [125]). This issue was solved by adding a plugin for jQuery that allowed the URLs to be manipulated by the JavaScript functions, allowing the web browser's back-button to be used. This is achieved by using the fragment identifier (or '*anchor*') element of a URL [126], with a link such as "*http://mot.dcs.warwick.ac.uk/mot31/#dmap6,concept80*", allowing exact links to be generated that navigate directly to the relevant concept.

7.3.3 Multiple labels/weights

Previous qualitative feedback had shown that users desired the ability to reference sublessons using more than one label, and thus create more complex strategies. Thus, multiple labels were introduced in MOT3.1. However, to ensure simple compatibility with existing tools (primarily ADE and AHA!), it was essential to make this change compatible with the CAF format.

The CAF format stores a *label* attribute and a *weight* attribute within the *link* element, for instance:

```
<link weight="5" label="beginner">PHP\PHP Syntax\Control
Structures\For Loop\image</link>
```

As described in Section 2.5.1.2 , weights and labels within LAOS have no intrinsic semantic meaning – the semantic meaning is defined by the adaptation strategy. In some strategies, the weight might be directly related to the label, or in other cases

they might be considered as separate pieces of metadata. This can be demonstrated by the following examples.

- **Label Scenario 1 - Weight related to label**: For example, in the case of a visual-verbal strategy, the label identifies the sublesson as a '*visverb*' concept, and the weight defines how visual or verbal the sublesson is.
- **Label Scenario 2 – Separate pieces of metadata:** For example, a *beg-int-adv* strategy, might be combined with *roll-out* strategy – where the weight of the sublesson might represent the number of times that the sublesson should be shown.

To ensure the maximum flexibility, after introducing multiple labels, it was decided to consider each label as having a separate accompanying weight, rather than allowing multiple labels to share a single weight.

In such a way, MOT3.1 allows authors to use multiple labels and weights. Internally, for the compatibility with the export format (CAF), this was achieved by concatenating the labels and weights together. For instance, if a user wishes to convey that a particular image is visual and suitable for beginners, they can use the MOT3.1 interface add a label named "visverb" with a weight of "80", and later add another label "beginner" with a weight of 5 – see Figure 7.2.

Figure 7.2: Adding multiple labels in MOT3.1

When the goal map is exported to a CAF file, the XML representation of the sublesson will be:

```
<link weight="0" label="visverb:80;beginner:5">PHP\PHP Syntax\Control Structures\For Loop\image</link>
```

However, this method of concatenating labels prevents many existing strategies from working with sublessons that do have multiple labels. For instance, the expression "`GM.Concept.label == visverb`" will not evaluate to true for the *PHP Control Structures* example shown above because the full label is now effectively "`visverb:80;beginner:5`".

To cater for such a scenario, ADE introduced the LIKE syntax into the LAG grammar. The LIKE syntax in LAG allows strategies to search for labels that match a particular pattern, so the expression "`GM.Concept.label == visverb`" can now be replaced with "`GM.Concept.label LIKE *visverb*`", or – if a particular weight is being searched for – "`GM.Concept.label LIKE *visverb:80*`".

Although this extension allowed working with multiple weights and labels, there were some limitations. The main issue is that the weights within the label string cannot be separately addressed numerically (i.e., no mathematical comparisons can be performed with the weight). For instance, it would be impossible to implement the *roll-out* strategy using multiple labels in this way, because the weight value of the `showatmost` label could not be separated from the rest of the label. It would therefore be impossible to compare the visited counter with the label's weight. To overcome this limitation, a '*primary*' reserved keyword was introduced. When the user adds a label named 'primary', the associated weight gets assigned to the sublesson – thus promoting the weight to the weight attribute of the *link* tag. This was implemented as a temporary solution to the label/weight pair limitation, a more permanent solution is documented in Section 10.4.2.3 .

7.3.4 Improved Wiki Importer

As described in Section 4.7.1.9 , the implementation of MOT3.1 introduced a new Wikipedia[37] import script which was redesigned to improve formatting, and – in response to user feedback – additionally process the images associated with each article.

7.3.5 Relatedness Search

The evaluation described in Chapter 5 showed that only one group created any relatedness links in MOT3.0. This suggested that the interface for allowing such links to be created was insufficient. Additionally, feedback had shown that the algorithm used to calculate the weights of relations between domain concepts was

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

not scalable, as every concept in the system had to be compared with every other concept, which could become quite time consuming for large domain maps.

As described in Chapter 5, the original algorithm used within MOT1.0 and MOT3.0 involves splitting every attribute in the system into individual words every time the user searches for related concepts via the 'Concept-oriented, relevance ranking method' [80]. MOT3.1 introduced a new algorithm based on TF-IDF [127], which provides quicker, more accurate results. To achieve this, a script periodically indexes the contents of all attributes within the system.

The first stage of indexing is to remove the HTML formatting of each attribute, and divide it into individual words. For each attribute, the indexing attribute algorithm then calculates the term frequency (*tf*) of each word, based on the formula: $tf = \frac{f_w}{|K_S|}$, where *tf* is the term frequency, f_w is the number of occurrences of the current word, and K_S is the set of words within the current attribute.

The *tf* is stored in a *wordfrequencies* table in the database that contains the columns *wordid*, *attributeid*, *frequency* and *tf*. Note that the *wordid* column is a foreign key to another table called *words*, which stores the actual word string.

The old (MOT1.0) style of calculating relatedness involved the server reading every attribute and dividing it into separate keywords every time a user performed a search. The new method described here is much faster since the attributes are indexed and thus the search only reads pre-cached numerical values. When the author wishes to discover concepts that are related to the current concept, the following query is performed:

```
SELECT DISTINCT(wf2.attributeid) FROM wordfrequencies as
wf1, wordfrequencies as wf2 WHERE wf1.wordid =
wf2.wordid AND wf1.attributeid = srcAttribute AND
wf2.attributeid <> wf1.attributeid
```

This statement looks up all attributes that have any words in common (wf1.wordid = wf2.wordid) with the source attribute. The attributes that have any words in common with the source attribute are known as target attributes. The search then creates a vector of *wordid*s and frequencies for the current attribute, and then creates vectors for all target attributes. A score for each target attribute is then calculated by comparing each target vector with the source vector using cosine similarity. An example of this is shown in Figure 7.3.

As with the MOT1.0 (and MOT3.0) relation search, the author can choose whether to restrict the search to the *keywords* attribute[38], or compare contents from other attributes.

Figure 7.3. Related Concept Search using TF-IDF

[38] This is achieved by adding an extra *WHERE* clause to the above SQL query.

7.4 PEAL2

Chapter 5 described the introduction of PEAL [53], which aims to simplify the specification of LAG strategies by providing adaptation authors with IDE-style features. Following on from PEAL, Jan Bothma (a 3rd-year BSc Warwick Computer Science student) created PEAL2 as part of his 3rd-year project in the academic year 2009-2010. PEAL2 focuses on providing a way of visualizing the flow of a LAG strategy through the use of flowchart style [128] icons.

PEAL2[39] introduces a visual programming feature, allowing authors to view the logic of their adaptation strategy using graphical flowchart elements (seeFigure 7.4). As with the original version of PEAL, PEAL2 allows the adaptation specification to be exported into a LAG file.

The interface for PEAL2 is divided into four main tabs.

- **Description**: A simple text box aims to encourage authors to provide a natural language description of what the strategy does, and what is required of the content author to use this strategy. The description tab appears red if it is empty, to encourage authors to add this important information. The description is important since it describes how the strategy works to non-programmer authors.
- **Initialization**: A visual programming section which allows the author to specify the *initialization* part of the strategy – see Figure 7.4. As described in Chapter 2, the initialization is only run once, when the user initially registers

[39] http://mot.dcs.warwick.ac.uk/peal2/

on the course. Hence, the visual program lets this be created in a different tab.

- **Implementation:** A visual programming section allows the author to specify the *implementation* part of the strategy – see Figure 7.5. Within the implementation tab there is a visual flowchart style representation to remind the user that their code will be run for each concept every time a user accesses a concept (as described in Chapter 2).
- **Text Editor:** The final tab retains all the code editing features of the original version of PEAL. Moreover, PEAL2 allows the user to switch between the visual authoring views and the code editing view, with the code being updated synchronously with the flowchart view.

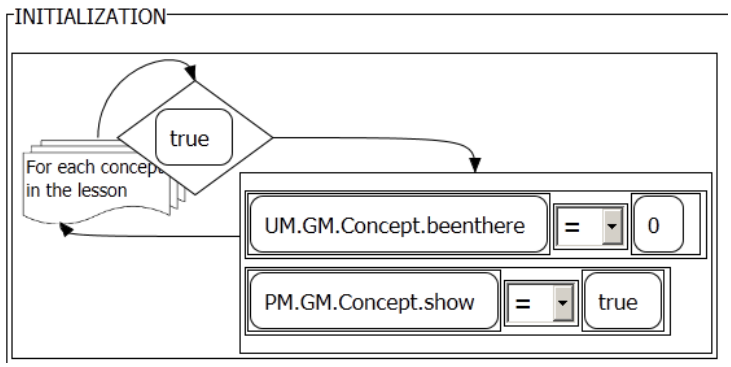

Figure 7.4: Part of an initialization loop in PEAL2

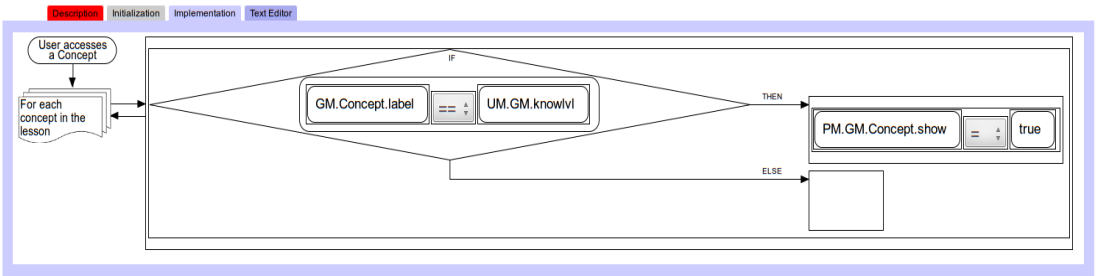

Figure 7.5 An implementation loop in PEAL2

7.4.1 Visual Programming User Interface in PEAL2

Casual verbal feedback had shown that it can be difficult for users to remember the precise meanings of the initialization and implementation sections. Specifically, the user needs to remember that the code within the implementation section is run every time the user clicks on a concept and for each concept in the lesson, which is dissimilar to the procedural programming style they are accustomed to. For this reason, the implementation section automatically contains a visual element, to remind the user when the section will be run (see Figure 7.6)

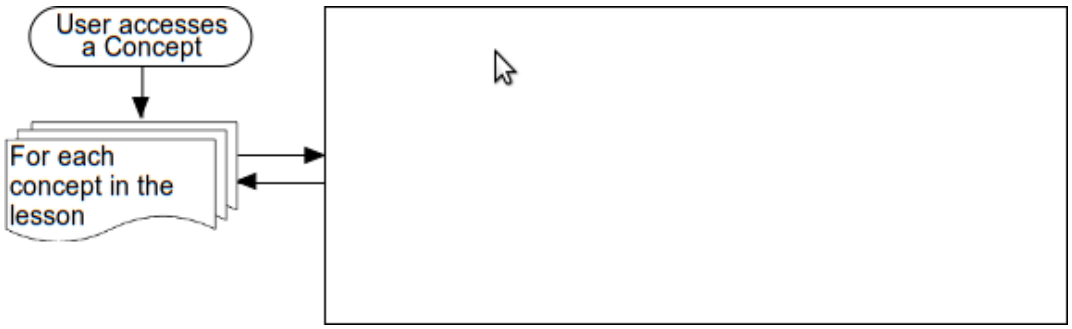

Figure 7.6: The default implementation section in PEAL2

PEAL2 uses other flowchart conventions [128] (such as using a diamond shape for decisions) to show other constructs of the LAG language. For instance, the diagram shown in Figure 7.7, represents the code shown in Figure 7.8.

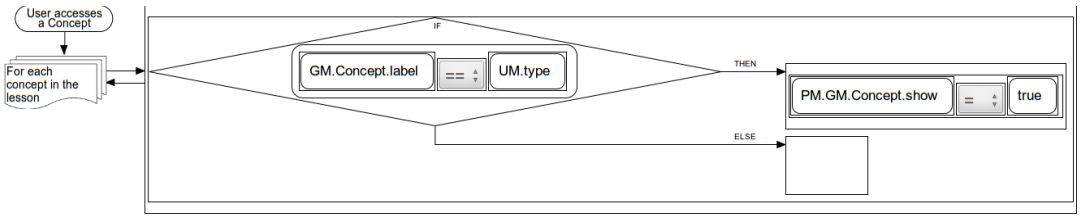

Figure 7.7: Simple code example in PEAL2

```
implementation (
  if GM.Concept.label == UM.type then (
    PM.GM.Concept.show = true
    )
)
```

Figure 7.8: Extract of LAG code in text mode of PEAL2

7.5 Delivering a course

The output of the MOT tool is stored in lightweight, exportable formats: the CAF format, using standard XML web technology, and the LAG format, using an adaptation language. These formats describe the content and adaptation strategy, respectively, that are needed to define the adaptive course. These files can be delivered via an adaptive delivery engine, such as AHA![56], ADE[50], WHURLE[54], etc. However – as described above – the newer features of multiple labelling are currently only supported by ADE.

7.6 Evaluation

To evaluate the effectiveness of the above improvements, the MOT toolset was used with the students from the 'Dynamic Web-based Systems' module in the winter term of 2010. This was performed in a very similar manner to the experiment described in Section 5.5 . As before, the students were given an introduction to the system, and encouraged to complete some smaller, non-assessed pieces of coursework during the term so that they could become familiar with the system. During this time, the module lecturers taught them the principles of adaptive hypermedia.

The structure of the coursework was very similar to the 2009 coursework (described in Section 5.5), with the students selecting two 'Dynamic Web' related

topics, and then creating two adaptive courses for each topic. The only major change to the structure of the coursework is that some groups were offered the chance to implement some adaptive software (such as an authoring tool or a delivery engine) instead of completing the normal coursework.

7.6.1 Analysis of Created Content

As part of the assessed coursework, the students were asked to create four adaptive courses, based on their allocated subjects (two courses per subject). Concretely, this means creating at least two Domain maps, and between two to four Goal maps (as described in other chapters, reusability allows more than one course to be generated from a single Goal map).

There are a number of ways that could be used to judge how appropriate a course actually is. One such method would be to subject the created courses to real-world students, who could be given a pre-test and a post-test – the scores of which could be used to assess how much each student had learnt through the system. Methods such as these have previously been used by Ghali [58] to discover the utility of various social interaction features.

However, the utility of the content created by the students is likely to vary dramatically between groups, because they are not pedagogical experts. Moreover, it is beyond the scope of this thesis to investigate exactly *how adaptation should be applied* to maximise the teaching value of the course. Some sources suggest that rather than making real-world teaching materials a showcase for the opportunities provided by adaptive hypermedia, the adaptation strategy should be subtle enough for the user to not be distracted by it [4]. The purpose of the toolset is, however, to

allow authors to express their content using whatever adaptation techniques they feel are appropriate. For this reason, it is essential that the adaptive toolset offers as much flexibility for specifying adaptation as possible, and encourages authors to add semantic metadata to allow complex adaptation strategies to be expressed.

For this reason, the evaluations investigate the usage of the tools by the students, and particularly focus on the *complexity* of the domain maps, goal maps and strategies that are created. If the students are able to define large, complex and well-structured domain and goal maps, it is likely that expert pedagogical authors will be able to use the tool to create sufficiently detailed (and pedagogically useful) courses.

7.6.1.1 Domain Content

As with the analysis performed in Chapter 5 , this section compares the Domain content from three different perspectives: *All Domain Maps, Best Domain Map per group* and *Average Domain Content per Group*. For the definitions of these perspectives see Section 5.5.2.2 .

	Number of Cases	*Concepts*	*Depth of Average Concept*	*Number of Attributes*	*Attributes per Concept*	*Number of Custom Attribute Types*	*Percentage Attributes (excluding title) containing more than 2 unique HTML tags*
MOT1.0: Average for All DMs	4	22 StdDev: 12.44	2.75 StdDev: 0.31	63 StdDev: 37.28	2.8 StdDev: 0.56	1.25 StdDev: 1.89	2.43 StdDev: 4.86
MOT3.0: Average for All DMs	29	22.24 StdDev: 12.22	2.93 StdDev: 0.7	54.62 StdDev: 30.18	2.52 StdDev: 0.6	1.75 StdDev: 2.79	19.4 StdDev: 21.93
MOT3.1: Average for All DMs	21	26.38 StdDev: 11.71	2.77 StdDev: 0.25	75.43 StdDev: 40.25	2.81 StdDev: 0.59	1.71 StdDev: 1.9	15.98 StdDev: 17.13
MOT3.0: Best DM Per Group	10	27.1 StdDev: 12.3	3.06 StdDev: 0.66	65 StdDev: 32.62	2.45 StdDev: 0.58	1.6 StdDev: 2.17	24.2 StdDev: 25.97
MOT3.1: Best DM Per Group	8	32.75 StdDev: 11.51	2.86 StdDev: 0.19	90.88 StdDev: 45.68	2.66 StdDev: 0.58	1.5 StdDev: 2.07	19.1 StdDev: 20.73
MOT3.0: Average DM per Group	10	22.2 StdDev: 10.12	2.85 StdDev: 0.61	54.34 StdDev: 21.5	2.53 StdDev: 0.42	1.64 StdDev: 1.59	20.5 StdDev: 25
MOT3.1: Average DM per Group	8	27.19 StdDev: 7.37	2.77 StdDev: 0.13	77.9 StdDev: 30.89	2.81 StdDev: 0.55	1.65 StdDev: 1.8	17.7 StdDev: 18.08

Table 7.1: Domain Production with MOT3.1 (2010) compared with MOT3.0 (2009) and MOT1.0 (2008)

Table 7.1 shows that for each category of measurement, the number of concepts has increased through the usage of MOT3.1 above that of MOT3.0 and MOT1.0. However, this difference is not statistically significant.

When considering all domains and the average domain per group, the increase in the number of attributes is statistically significant. Moreover, when considering all submitted domains, the number of attributes per concept also shows a statistically significant increase over MOT3.0. Table 7.2 shows the effect size for these improvements when using an independent t-test at a 95% confidence interval, and using Pearson's correlation coefficient (*r*), as described in Chapter 5 .

	Degrees of Freedom	***t***	***r (Effect Size)***
Average DM per Group: Attributes	16	-1.908	0.43 (medium)
All DMs: Attributes	48	-2.091	0.29 (small)
All DMs: Attributes per Concept	48	-1.7	0.24 (small)

Table 7.2: Effect sizes for attribute creation in 2010

The depth of the average concept was actually slightly smaller in 2010, yet still larger than the depth in 2008. However, the change in depth is not statistically significant.

Similarly, the percentage of attributes containing more than two HTML tags, and the number of custom attributes defined were also slightly lower in 2010, but these changes were also not statistically significant.

Interestingly, in 2010 (using MOT3.1) no groups used the relatedness relations features at all. As suggested in Chapter 5 (which discussed how there was only one such relation used in 2009), this lack of use could be directly related to the fact that the LAG language only provided limited support for such relations.

7.6.1.2 Goal Map Content

A similar analysis can be applied to the Goal Map content in the submitted CAF files. As with the analysis in Chapter 5 , the following criteria are considered: *All GMs*, *Best GM per Group* and *Average GM per Group* – for details about these categories, see Section 5.5.2.3 .

	Number of Cases	*Average Number of Distinct Labels*	*Average Number of Sublessons*	*Percentage of Labelled Sublessons*	*Percentage of Weighted Sublessons (i.e., labels with weight other than 0)*
2009 (MOT3 0): All GMs	40	3.83 (SD: 3.09)	62.38 (SD: 35.38)	53.25 (SD: 34.73)	37.29 (SD: 34.73)
2010 (MOT3.1): All GMs	24	5.88 (SD: 10.78)	83.17 (SD: 48.56)	68.26 (SD: 37.27)	32.26 (SD: 41.17)
2009 (MOT3.0): Best GM per group	10	4.7 (SD: 3.9)	96.4 (SD: 41.31)	54.89 (SD: 33.16)	42.3 (SD: 31.91)
2010 (MOT3.1): Best GM per group	8	11.25 (SD: 17.96)	106 (SD: 63.05)	84.44 (SD: 31.44)	67.27 (SD: 40.75)
2009 (MOT3.0): Average GM per group	10	3.83 (SD: 2.26)	62.38 (SD: 21.03)	52.29 (SD: 21.39)	38.94 (SD: 24.02)
2010 (MOT3.0): Average GM per group	8	5.3 (SD: 4.87)	86.96 (SD: 42.14)	69.55 (SD: 28.25)	39.53 (SD: 25.41)

Table 7.3: Goal Map Analysis in MOT3.1 versus that of MOT3.0

Table 7.3 shows an overall increase in the *number of distinct labels*, the *number of sublessons* and the *percentage of labelled sublessons*. When looking at all Goal Models, there is a decrease in the *percentage of weighted sublessons*. However, when looking at the *Best GM per group* and the *Average GM per group*, there is a general increase. Nevertheless, the only statistically significant results shown by these results are that for the *All Goal Models* category, the number of sublessons was higher ($t(62) = -1.976$, $p < 0.05$, $r = 0.24$) and that when considering the *Best GM per group*, the percentage of labelled sublessons was significantly higher ($t(16) = -1.922$, $p < 0.05$, $r = 0.43$).

The increase in the average number of distinct labels is a direct result of introducing multiple labelling. For instance, the statistics presented in Table 7.3 would consider the labels “image:0;beginner:0” and “image:0;advanced:0” as two distinct labels, even though the strategy may check for three separate sub-labels (*image*, *beginner* and *advanced*). When considering only the individual unique sub-labels, there was an average of only 3.92 (SD: 2.64) when considering the average goal map per group. This rises to 4.33 (SD: 4.68) when considering all 2010 goal maps or 6.5 (SD: 7.33) when considering the Best Goal map per group.

The fact that the percentage of weighted sublessons is lower for 2010 than for 2009 could be because of the increase in usage of multiple labelling. Due to the limitation in addressing a specific weight in LAG when using multiple labels, some groups opted for strategies that did not utilise the weights. Interestingly, one of the groups that used multiple labels appeared to be combining the labels with a number, such as “textL:0;1L:0;”, “textL:0;4L:0;” and “sumL:0;16L:0”. However, the strategy they

created to use the goal map did not actually utilise such labels. This implies that the students desired a way of adding complex multiple labels to the content, yet the implementation of LAG (and therefore MOT) was insufficient. This multiple labelling problem is addressed in Chapter 10.

7.6.1.3 LAG Strategy Content

The LAG code created by students was also statistically examined as follows. We were analysing if the improved version of PEAL had any impact on the type of code the students wrote. Here, the quantitative analysis is presented, in terms of quantity of code, description and comments. For each category, we count both the number of lines and the number of characters.

As with other analysis (and as explained in Section 5.5.4.2) each group is considered from three separate perspectives: *All Strategies*, *Best Strategy per Group* and *Average Strategy per Group*.

	N	*Lines of Code*	*Lines of Description*	*Lines of Comments*	*Characters of Code*	*Characters of Description*	*Characters of Comments*
PEAL (2009) All Strategies	38	46.18 SD: 25.65	8.66 SD: 6.78	6.53 SD: 9.92	957.79 SD: 613.274	493.45 SD: 599.37	391.61 SD: 680.928
PEAL2 (2010) All Strategies	31	48.94 SD: 37.18	10.94 SD: 6.78	10.19 SD: 9.59	964.32 SD: 614.27	711.1 SD: 573.61	574.52 SD: 448.44
PEAL (2009) Best Strategy per Group	10	66.4 SD: 22.76	10 SD: 7.5	12.1 SD: 17.15	1457.9 SD: 685.54	680.8 SD: 788.88	757.5 SD: 1182.25
PEAL2 (2010) Best Strategy per Group	9	76.56 SD: 51.79	11.67 SD: 11.68	16 SD: 14.49	1560.33 SD: 1211.04	650.33 SD: 682.43	818.78 SD: 206.66
PEAL (2009) Average Strategy per Group	10	45.7 SD: 17.02	8.53 SD: 5.51	6.2 SD: 7.04	950.55 SD: 456.59	481.83 SD: 521.36	372.03 SD: 517.02
PEAL2 (2010) Average Strategy per Group	9	47.75 SD: 22.22	10.06 SD: 7.57	9.03 SD: 6.65	932.94 SD: 503.48	639.64 SD: 475.7	510.61 SD: 349.68

Table 7.4: LAG Content when using PEAL2 in 2010 versus PEAL in 2009

Table 7.4 shows a small increase in the amount of code in the submitted strategies for the group of 2010 when compared to the 2009 one. However, this difference is not statistically significant. Similarly, there is an increase in the amount of comments and description, but this also is not statistically significant.

The introduction of the original version of PEAL (described in Chapter 5) had shown a significant improvement in the amount of code, description and comments over the 2008 cohort that could only use a text-editor to create LAG files. However, the difference between the strategies created using PEAL2 and those created using PEAL is not as significant.

Since the visual way of writing LAG strategies was implemented as an extension of the original PEAL, it is likely that many students decided not to use the visual features of PEAL2, and simply used the original features of PEAL. This would explain why there doesn't appear to be much difference between the usage of PEAL and PEAL2.

The icons used in PEAL2 are very large, and complex strategies are thus quite difficult to display within the web-browser's window. Figure 7.9 shows a screenshot of an (incomplete) Beginner-Intermediate-Advanced strategy. The font size in the browser had to be reduced several times in order to see an overall view of the strategy. It is, therefore, quite difficult to use PEAL2 with large strategies, and scalability is certainly an issue. This could be one of the reasons why there appears to be no significant difference between the strategies created in PEAL and those created in PEAL2.

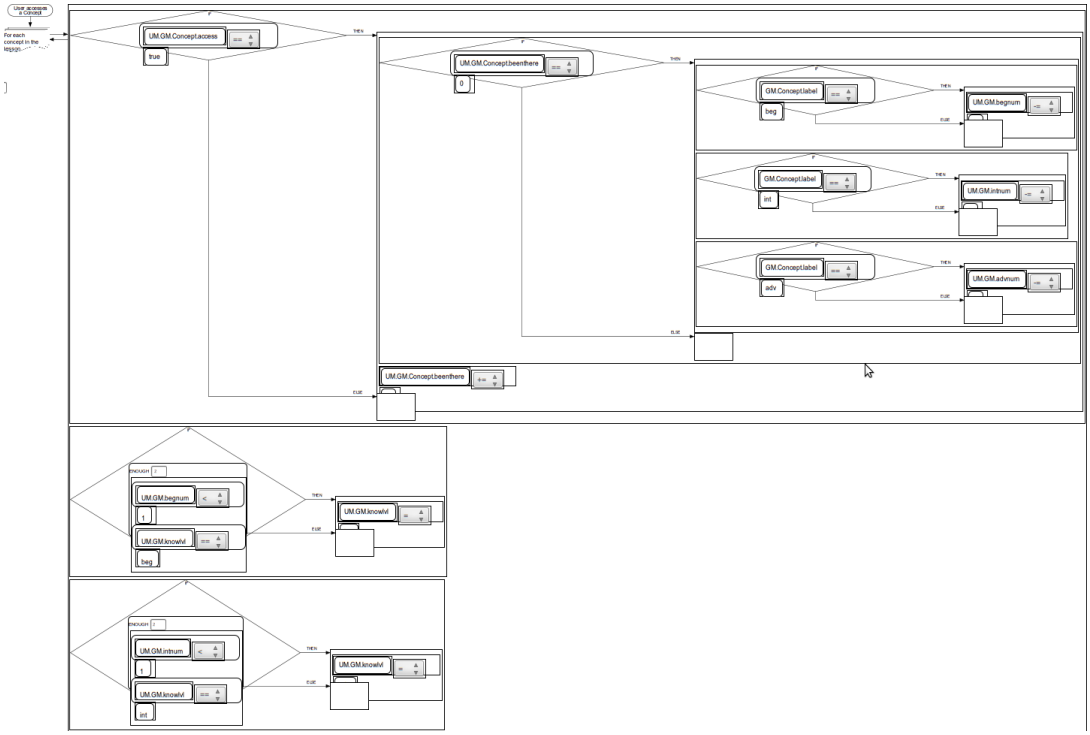

Figure 7.9: Extract of a Beginner-Intermediate-Advanced Strategy in PEAL2

7.7 Conclusion

This chapter has described a further iteration in the development of the MOT toolset, MOT3.1, which was based directly on the feedback gathered from MOT3.0. The chapter has also presented an evaluation of this version of the MOT toolset. Specifically, this chapter has introduced the idea of specifying adaptation using graphical programming techniques.

Evaluations of the submitted content have indicated that multiple labelling is a useful feature. However, the support for multiple labelling was not found to be sufficient enough to encourage authors to create strategies that fully utilize such labels. Further work on this area is presented in Chapter 10.

The evaluation described in this chapter has found no direct evidence that the currently used visual programming tool (PEAL2) assisted the authors with the

process of specifying an adaptation strategy. There is no statistical evidence to show that the quality of the strategies got worse (indeed, there is a very small general improvement in the quantity of the submitted content), this could be because the students simply used the text-based features of PEAL2.

PEAL2 has not demonstrated that authors have a clear preference for such visual authoring of adaptation. However, the next chapter discusses an alternative method for visually specifying adaptation.

8 Visual graph-based authoring: GAT

8.1 Overview

This chapter expands on the description of the GRAPPLE project provided in Chapter 2, and documents the changes that have been made as part of the work on this thesis, to improve the usability of the GRAPPLE Authoring Toolset.

8.2 Related Research: The GRAPPLE Project

The GRAPPLE EU FP7 STREP project [4] sets out to "enable adaptive life-long learning for learners in higher education and industry" [129]. One of the primary aims of the project was to integrate adaptive learning environments with existing LMSs – as described in Chapter 2. This chapter uses the '*Milky Way'* example, which was created by Ploum [130], and is widely used as an example when describing GRAPPLE [4], [26].

The adaptive authoring and delivery aspect of the project focussed on two main tools – GALE and GAT – as briefly revisited below.

8.2.1 The Generic Adaptive Learning Environment (GALE)

The Generic Adaptive Learning Environment (GALE) [26] is a delivery engine for adaptive hypermedia. It is based on the previous research carried out at Technische Universiteit Eindhoven (TU/e), and is a direct successor to the AHA! delivery engine [23]. In terms of authoring, the primary input format for GALE is the CAM XML language [37], which is authored using GAT.

8.2.2 The GRAPPLE Authoring Toolset (GAT)

The GRAPPLE Authoring Toolset (GAT) was designed to make the process of authoring for adaptive hypermedia easier. The toolset was designed using Adobe Flash Builder[40], allowing it to be run in any browser that supports Adobe Flash Player[41]. The design and implementation of the toolset was carried out as part of Work Package 3 of the GRAPPLE project. The authoring framework for GRAPPLE uses features from both the AHAM model [32] and the LAOS model [35], and is based on research on most existing authoring frameworks [46].

The design and implementation of the authoring tool was divided into four separate areas: the *Domain Model* (DM) tool, the *Concept Relationship Type* (CRT) tool, the *Concept Adaptation Model* (CAM) tool and the *Shared* interface. The researchers at the University of Warwick were directly responsible for the CAM and the Shared components of the authoring tool, but also led the Work Package that oversaw the development of the overall authoring tool.

This section provides a basic overview of the features provided by the authoring tool (and is thus presented as '*Related Research*'). Such an explanation is necessary in order to describe the work presented in Section 8.3 which details the usability and functionality changes that were made to the CAM and shared tools based on previous research described in this thesis.

[40] http://www.adobe.com/products/flash-builder.html

[41] http://www.adobe.com/uk/products/flashplayer.html

8.2.2.1 Domain Model Tool

Led by eXact Learning Solutions, Italy.

The Domain Model [44] tool allows authors to divide their content into *Concepts* which are linked by *relationships* to form a graph structure.

8.2.2.1.1 Concepts

Concepts are represented as nodes in the Domain graph. Each concept can have a number of resources and properties.

8.2.2.1.2 Resources

Resources refer to the URL of the document containing learning content. Any document that can be displayed in the user's web browser can be used as a resource – although HTML pages are typically used[42]. Specifically, XHTML pages can be used to provide more detailed adaptation (as described below).

8.2.2.1.3 Properties

Each concept has a number of properties that are used as semantic metadata. The properties can be addressed as part of the adaptation strategy using GALE code, either as part of an XHTML resource, or within a pedagogical rule (see Section 8.2.2.3).

- *Name*: The name can be used as the name of the concept when generating the hierarchical menu, and is also used to identify concepts within the

[42] All HTML pages are automatically converted to XHTML by GALE at time of delivery.

authoring tool[43]. It is also used by the delivery engine to generate the URI of the concept, such as *gale://gale.tue.nl/cam/Milkyway/Star* - where *Milkyway* is the name of the CAM (course) and *Star* is the name of the concept.

- *Description*: The description is used primarily within the authoring tool rather than being displayed to the learner.
- *Order*: The order property is numeric, and is used to decide the concept's position in the hierarchical menu.
- *Title*: If the title property is specified, it will be the title of the concept shown in the hierarchical menu (instead of the name).
- *Type*: GALE automatically creates a header that says "*has read X pages and still has Y to read*". When calculating the values of *X* and *Y*, GALE only considers concepts that have their type set to *page*.

Figure 8.1 shows the list of properties in the editing pane for the *Star* concept. As well as the above defined properties, the creator of the *Star* concept has added some custom properties; *image*, *info* and *info1*.

[43] Please note that internally the tool uses GUIDs for identifying concepts. However, the name is used as a human-readable unique identifier that is shown to the author.

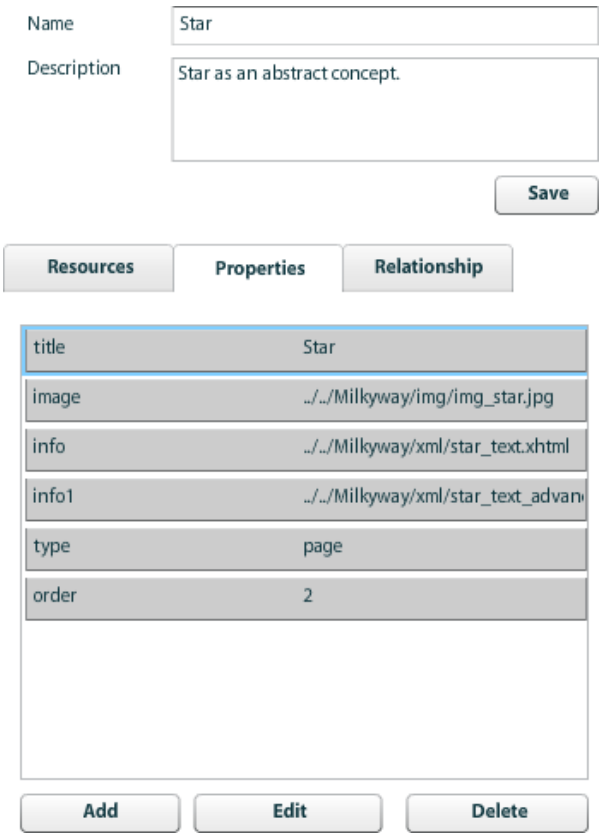

Figure 8.1: The Properties of the Star Concept

In the case of the *Planet* concept, the associated XHTML resource contains the following GALE code:

```
<h1><gale:variable expr="${?title}"/></h1>

<h4>Is <gale:variable expr="${->(parent)?title}"/>

of: <a><gale:variable expr="${->(isPlanetOf)?title}"/>

<gale:attr-variable name ="href"

expr="${->(isPlanetOf)?title}"/>

<gale:adapt-link/></a></h4>
```

```
<h2>Image of <u><gale:variable
expr="${?title}"/></u></h2>

<img><gale:attr-variable name="src"
expr="${?image}"/><gale:attr-variable name="width"
expr="300"/></img>
```

The GALE-specific XML instructs the parser to substitute the values from the properties – the resulting output is shown in Figure 8.2. This means that the author can create a *template 'planet'* concept and reuse it to generate each of the planets, by simply modifying the properties for each concept rather than the resource.

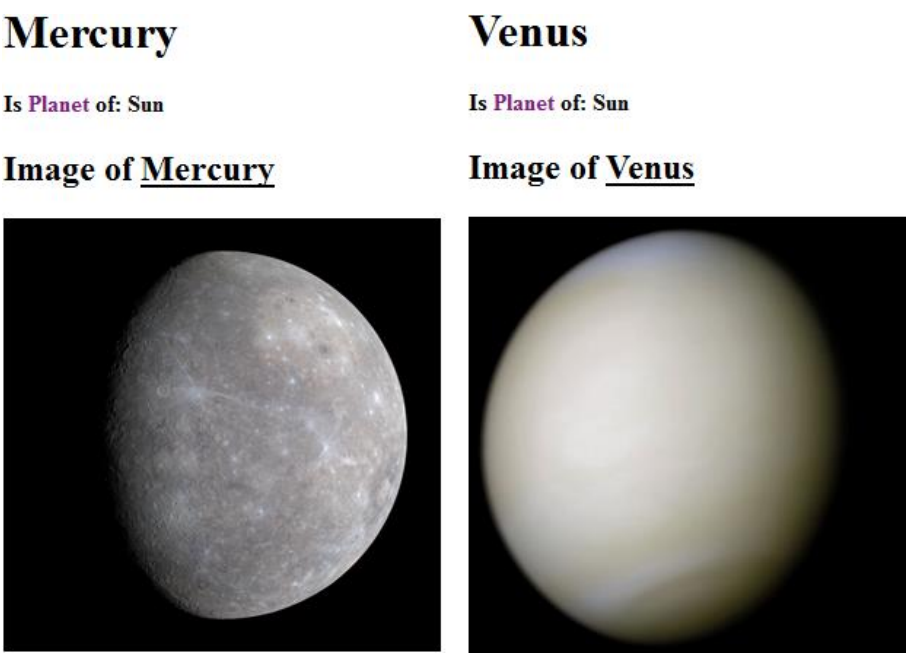

Figure 8.2: Part of the generated pages for the Mercury and Venus concepts

However, using template concepts in this way means that the domain author would require knowledge of the XHTML and GALE XML tags, since GAT does not provide

any support for the creation of the XHTML resource. It is therefore expected that many domain authors will use larger static resources (as described in Chapter 9).

8.2.2.1.4 Relationships

Each concept can take part in relationships. Relationships form the *directed* edges of a graph structure (see Figure 8.3), and represent a semantic relationship between exactly two concepts such as '*is-a'* or '*belongs-to'*. The relationships used can be defined by the user, and can therefore be specific to the content. The semantic information from the relationship definition can be used by the adaptation. For instance, the '*parent*' relationship, is used to generate the hierarchical navigation menu. The reuse of domain relationships in pedagogical settings should normally be done within GALE code (specified either as part of the resource as described above, or in the CRT tool – see Section 8.2.2.3). However, for the '*parent*' relationship, a shortcut was used such that it is always interpreted as the relationship to inform the navigation menu. This is a decision that moves away from the separation of concerns between content and pedagogy, but was made in the project for expediency sake.

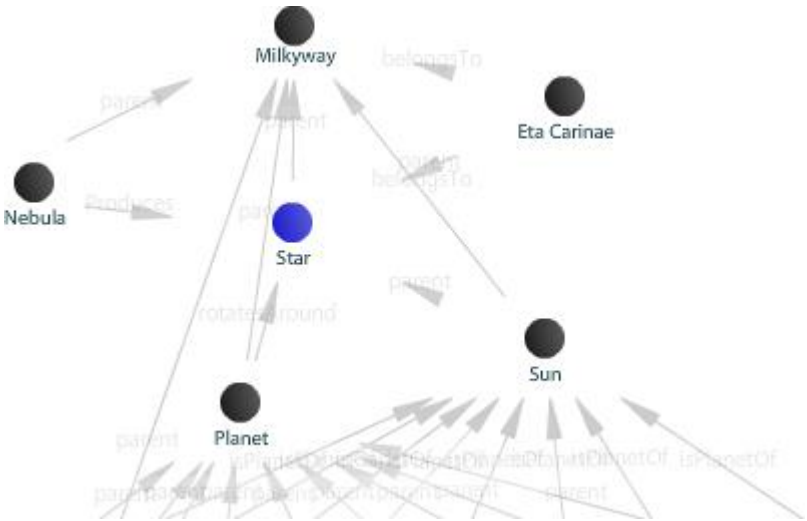

Figure 8.3: Part of the *Milky Way* example domain (the *Star* concept is currently selected)

8.2.2.2 Concept Adaptation Model Tool

Led by the University of Warwick, UK – first version developed by Maurice Hendrix[46]**, updated versions developed by Jonathan Foss**

The Concept Adaptation Model (CAM) tool [37] (later also called 'Course Tool', for simplicity) allows authors to specify how the Domain concepts should be presented by the adaptive course. This is achieved by enabling the user to choose from a variety of pre-created pedagogical rules. In this way, the Course Tool combines input from the other two tools, Domain Tool and Pedagogical Rules Tool, in order to create a real course instance.

Each pedagogical rule contains one or more sockets, which are placeholders for the (one or more) concepts that will be involved in that rule. The rule then defines how the adaptation that is specified in the rule should be applied to the concepts in the sockets. Sockets have names that allow the adaptation code to address them. Typically, rules that require two sockets name them *source* and *target* – denoting

that the rule reads values from the concepts in the source and consequently manipulates a value on the target concepts**.**

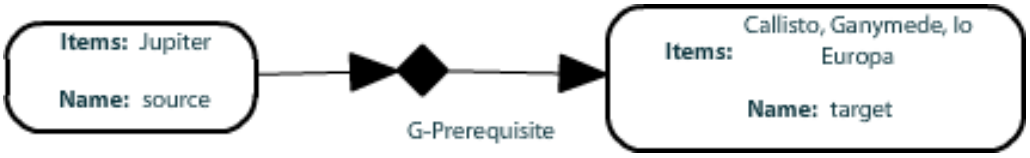

Figure 8.4: A *G-Prerequisite* CRT rule as displayed in the CAM tool

There are 13 different rules that are defined by default. Authors can create a course by using the CAM tool to insert any combination of these rules into their course, and then populating it with concepts from the Domain model. The GRAPPLE authoring team has created a number of pre-defined pedagogical rules, which are to represent the initial stage of the rule library, as well as highlight how the pedagogical rules can be used and show the range of possibilities available. Thus these rules are considered essential for a non-programmer to use, as main building blocks for any adaptive course, as follows.

- *G-Start***:** Defines which concept should be shown when a user first registers on the course.
- *G-Layout*: Generates the basic page layout (including the navigation menu, header and footer) when any of the concepts in its socket are accessed. The hierarchical navigation menu is created by the *static-tree-view* component in GALE, which uses the parent relationship (and the order property) from the domain – as described above.
- *G-Prerequisite:* States that a concept (or a group of concepts) is a prerequisite of another concept. Concretely, GALE will not set the suitability value (of any of the concepts in the target socket) to true until the user's

knowledge of the concept(s) in the source socket has exceeded a threshold value (by default, 80%). As described in Chapter 2, from the point of view of the learner, the navigation menu will show the target concept(s) as black text (with a red icon) until all prerequisites have been satisfied, and the navigation menu link will turn blue (with a green icon).

A more detailed explanation of all pedagogical rules can be found in the online GRAPPLE Tutorial [131]. The set of pre-created pedagogical rules are designed to allow beginner authors to create a wide variety of different adaptation strategies. However, expert authors may wish to create their own pedagogical relationship types using the Concept Relationship Type tool.

8.2.2.3 Concept Relationship Type Tool

Led by University of Graz, Austria

Concept Relationship Types (CRTs) [132] (also called 'Pedagogical Relationship Types' as described in Section 8.3.1.1) are the pedagogical rules that describe how adaptation should be applied to certain (groups-of) concepts. The idea of Concept Relationship Types was originally used by AHA! [23]. The CRT tool allows expert authors (i.e. programmers and pedagogical experts) to write GALE code that describes the adaptation that should be applied to the concept(s) in the rule's sockets. The tool is not intended for the non-programmer author, who can use pre-existent pedagogical rules as described above. Unlike the other tools in GAT, the CRT tool is entirely dialog-based, rather than graphical. The interface allows the author to define the following types of information about the pedagogical relation.

- *General:* Allows the author to specify basic information about the CRT, such as name, description and how many sockets will be available in the CRT.
- *Meta Info:* Allows the user to specify a comment (pedagogical description), and define various parameters. This description is important for non-programming authors, and thus should be written in laymen's terms, but be clear enough, so that the pedagogical rule can be used by such authors. The interface allows the CRT creator to define default values for these parameters. However, the CAM interface allows the author to further adjust these variables for individual instances of the rule.
- *Code:* A simple text box that allows the author to specify the GALE code that will be used by the pedagogical rule.
- *User Model:* Allows the author to define user model variables that will be used by the rule.

8.2.2.4 Shared Interface

Led by the University of Warwick, UK – first version developed by Maurice Hendrix[46]**, updated versions developed by Jonathan Foss**

To provide a coherent look-and-feel between the three separate tools of the toolset, a shared component in the project was developed. This component is also responsible for functions that apply to all tools, such as file management and communication with other tools such as GUMF [133] and GALE [26].

8.3 Enhancing the Usability and Functionality of the GRAPPLE Authoring Toolset

The GRAPPLE Authoring Toolset aims to simplify the process of authoring for adaptive hypermedia, and as such should be usable by a wide variety of users (of varying technical abilities). The content authors may be educators (i.e. teachers and lecturers) from a variety of different disciplines to a variety of different types of students (of any age). Concretely, the toolset was designed to allow all content authors to be able to easily create domain maps based on their own content. Moreover, through the use of modular 'building brick'-style pedagogical rules, the GRAPPLE project aimed to allow non-technical authors to specify the adaptation that should be applied within their own courses. However, as part of the project, various issues with the implementation of the GRAPPLE Authoring Toolset were identified.

As described above, the University of Warwick was primarily concerned with the design and implementation of the *shared* and *CAM* components. This section describes the improvements that were made in response to user feedback. Moreover, many of these improvements are directly in alignment with some of the principles that have been described in previous chapters when discussing the MOT toolset.

8.3.1 Interface Changes

8.3.1.1 Name Changes

Some of the qualitative feedback provided at the ECTEL2009 and ICALT 2009 conferences [46] had shown that the terms *Domain Model (DM)*, *Concept*

Relationship Type (CRT) and *Concept Adaptation Model (CAM)* were confusing. These terms were renamed to *Domain*, *Pedagogical Relationship Type (PRT)* and *Course* respectively, in the new version created during this thesis. Moreover, whereas individual CRT instances in the CAM had been referred to as *CRT instances*, dialog boxes in the Course tool now refer to specific instances as *pedagogical rules*.

Similarly, the terminology used for model sharing permissions was changed. Previously, the tool used the terms *readwrite*, *read* and *nopermissions* – now *Public* (can be read and modified by anybody)*, Read Only* (can be read by anybody but can only be modified by the original author) and *Private* (can only be read and modified by the original author) are used, again, for more clarity.

8.3.1.2 Welcome Screen

Previous versions of GAT had presented the user with an empty screen when the tool was first opened. The user then had to go to the File menu and choose to either create a new model or open an existing model. A welcome screen was designed to immediately present the user with some suggestions of frequently used tasks (see Figure 8.5). It was designed to be colourful to attract the user's attention. Each button contains a main heading and a sub-heading that describes the functionality, thus introducing the user to the terminology used by the tool. This is similar to the type of welcome screens that are commonly used in software such as SPSS [6] and Microsoft Access[134] which (unlike other software such as Microsoft Word[39]) offer the user a choice of activities when the program is launched. The fonts and colours used were selected in such a way that they are not obtrusive, but

at the same time convey the feeling that it's 'child's play' to use the system; similar to the use of vivid use of colour in Scratch[69].

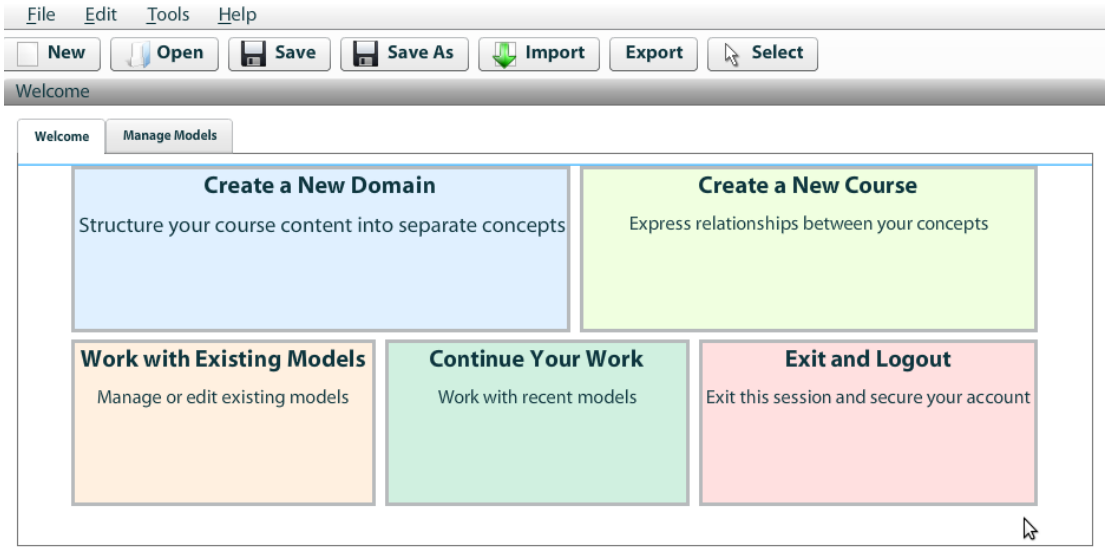

Figure 8.5: The Welcome screen in GAT

8.3.1.3 Toolbar

In line with many other common applications, GAT has a button toolbar (shown in Figure 8.5) that allows simple access to some of the most frequently used functions. Where possible, commonly recognised icons have been chosen to accompany the text on the buttons. The icons are based on the accepted usability conventions for icon sets, and look similar to those used in widely-used software such as Microsoft Word[39], Microsoft PowerPoint[92] and Apache OpenOffice[93].

8.3.1.4 Advanced Mode

As described above, non-technical authors should be able to design adaptive courses by only using the Domain and the Course tool. Moreover, beginner users need not concern themselves with the creation of new PRTs. This separation of advanced and beginner features was made much clearer in the version designed as part of this thesis work – when GAT launches, initially only the options for creating

and modifying Domains and Courses are available. The user can choose to work with the pedagogical relationship type tool by selecting '*Advanced Mode*' from the *Tools* menu.

8.3.1.5 Prevention of accidentally closing the window

Since GAT is an Adobe Flash application, it can easily be terminated by closing the web-browser window. To prevent the user from accidentally closing the window and losing their unsaved work, a piece of JavaScript code was introduced to warn the user before the web-page unloads.

8.3.1.6 Export and Import functionality

Export and Import features were created to allow models to be saved as XML files to the user's local file-system. Users would not normally need to work with XML files, since the models are stored within the GAT server's database. However, import and export functionality is useful when users wish to transfer files between different GAT installations. Previously, this functionality was only available (via a hack) by copying/pasting the XML in the '*Debug*' tab (which is present in each tool).

8.3.1.7 'Manage Models' Screen

In conjunction with the welcome screen, a new screen was created to allow simple access to each separate type of model. The '*Manage Models*' screen (see Figure 8.6) allows the user to perform various tasks with the selected model.

- *Export Model*: Allows the user to save the selected model to an XML file on their local file-system. This is not usually necessary, since all models are saved in the GAT online database, however it is designed to allow models to be shared between two GAT installations, or any systems which can process

the GAT output files (in conjunction with an import function that reads XML files).

- *Clone Model:* Allows users to create a clone of the model. This is particularly useful if a user wishes to create a model that is based on another user's model.
- *Delete Model:* Removes the model from the GAT database.
- *Properties:* Launches a dialog that allows the user to change the model's name, description or sharing permissions.

Each model type has two separate tabs ('My [modeltype]' and 'Public [modeltype]'), one shows all the models that have been created by the current user, and another shows all the models that can be read by the current user (all content that the user has permission to read).

All this represents a reorganisation of previous functionality. In the previous version of GAT, all models, domain models, courses, even advanced pedagogical relations were called DM, CAM, CRT (or PRT) and listed all together in one big list. User studies [46] showed that it was difficult for users to find the ones they made, to differentiate between the domain models and the courses (as often they would have the same or a similar name), to find recent ones, etc., although filtering functionality had been provided (via a search box similar to that shown in Figure 8.6). Dividing these model types into sections, together with the simpler naming

convention, was aimed at being much more user-friendly.

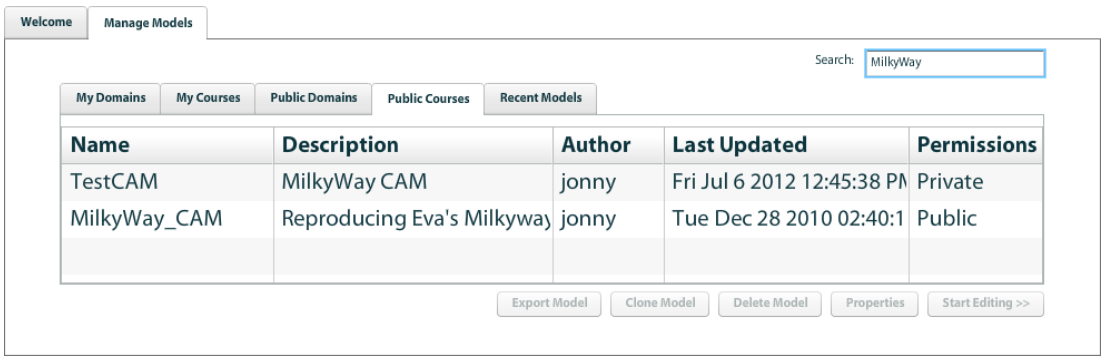

Figure 8.6: The 'Manage Models' screen

8.3.1.8 Multiple Selection Rectangle

Another issue in the previous implementation of GAT was that, whilst the domain and course tools were graphical, actual graphical manipulation of the objects on the screen was reduced. Specifically, concepts had to be selected individually, and this was alleviated in the new version, to bring the toolset in line with modern commercial software principles. In common with many graphical drawing programs, GAT now allows users to select multiple items at once by drawing a '*rubberband*' selection rectangle [135]. This was implemented in the *Shared* component, thus allowing the feature to be used by both the Domain tool (for selecting concepts) and the Course tool (for selecting rules), to give a more common 'look and feel'.

8.3.1.9 Drag & Drop Concepts from Domain to Course

Amongst the limited graphical manipulation facilities of the previous implementation was that of the copying functionality. Previous versions of GAT allowed the user to double-click the socket (to open the '*Edit socket*' dialog) then click *Insert Concepts,* choose a Domain from the file manager and then select from a list of concepts. Alternatively, the user could select concepts from the Domain tool, and select *copy* from the menu. The concepts could then be pasted into the '*Edit*

socket' dialog. Both of these methods performed the required functionality, but were considered problematic from a usability point of view. This was mainly because each of these methods required several steps (involving a number of dialogs). Moreover, the process of inserting concepts into sockets is a vital part of the course creation process and must be performed many times. Similarly, there was an expectation from the user that since the tools are presented in the form of a graph, they would expect to be able to select multiple items from it in the graph, rather than having to carry out such a laborious process.

The updated version of GAT allows authors to dock a domain window next to a course window. The user can then drag a concept from the domain, and drop it directly into a socket in the course.

8.3.2 Course Tool Changes

This section describes specific changes that were made to the Course (previously, CAM) tool.

8.3.2.1 Rectangular Socket Shape

Previous versions of the course tool had used ovals for displaying the sockets of a rule. This meant that concept names were often truncated. The updated version uses rounded rectangles that are the same width and height as the ovals, but allow more concept names to be displayed. Moreover, if the list of concepts is too long to be displayed, the socket is expanded upon hovering, allowing the full list of concepts to be displayed when users move their mouse over the socket (see Figure 8.7).

Figure 8.7: A truncated list of concepts and an expanded socket

8.3.2.2 Pedagogical Rule Toolbox

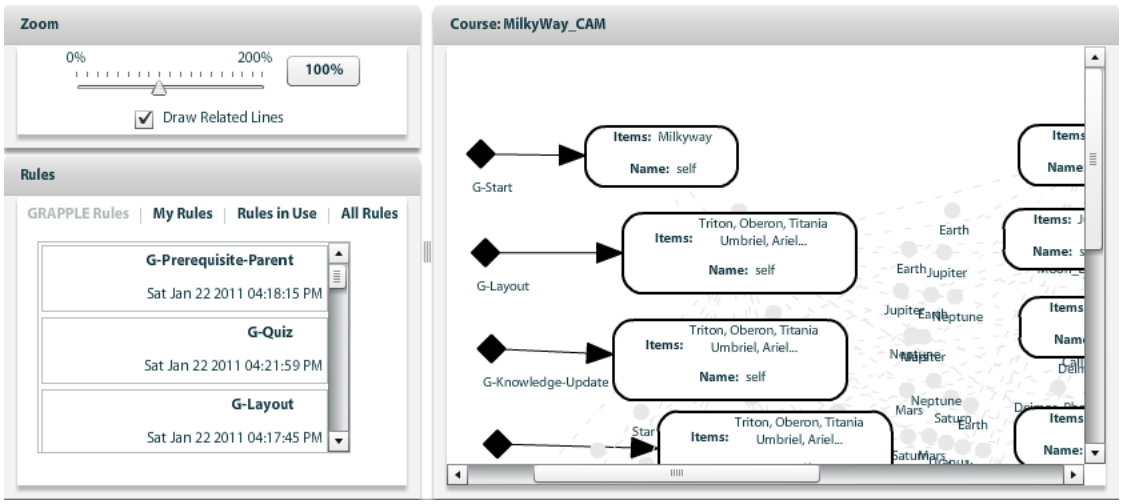

Figure 8.8: The course tool with Rule toolbox

Previous versions of the course tool had simply provided the user with a blank canvas and allowed them to right-click on the canvas to access a menu item to insert a pedagogical relation. This menu would then present the user with a dialog, from which they could choose a pedagogical relation to insert. That meant that at least two or more clicks were necessary for each pedagogical relation. This was considered inefficient, as inserting pedagogical relations is a repetitive action that a

user (author) would have to do, often having to insert the same pedagogical relations repetitively.

Instead, the updated version of the course tool provides a toolbox on the left side of the screen. This allows the user to select the appropriate rule, and drag-and-drop it into the course canvas, in one step. Moreover, instead of having a long list of a variety of rules (some of which might not necessarily be finished, and still being slowly updated by other authors) the rules are divided into relevant categories. The toolbox has the following four categories.

- GRAPPLE Rules: The basic rules that were originally created as part of the GRAPPLE project. These were selected to be the most frequently used adaptation rules (including start, stop and prerequisite rules), which are indispensable for non-programmer users to be able to add any adaptive features to a course.
- My Rules: All rules created by the current user.
- Rules in Use: All rules that are currently used in the current course. Thus, when opening a course by another author, a user can see at a glance what rules are used in it.
- All Rules: All rules in the GAT database.

8.3.2.3 Selecting rules according to the type

Hendrix [46] had suggested that the course tool could be improved by allowing the user a finer control over which rules should be shown. Figure 8.9 shows that the *G-Start* and *G-Layout* rules are selected in the toolbox, and hence the *G-Start* and *G-Layout* instances in the course are highlighted.

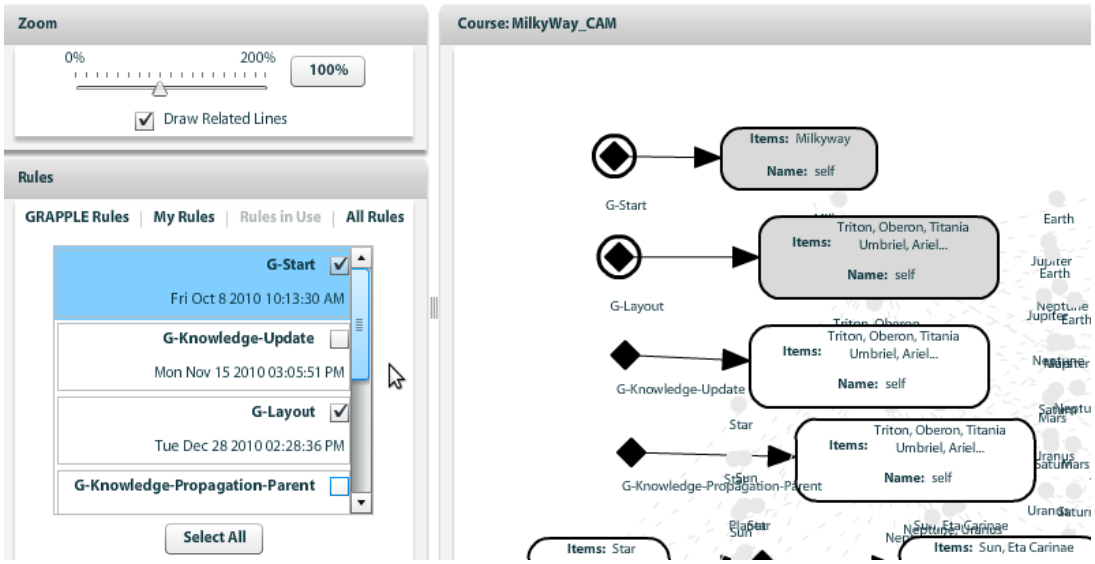

Figure 8.9: Selecting rules by type in the Course tool

8.3.2.4 Performance Improvements

The course tool was demonstrably slower when many pedagogical rules were inserted into a course. It was found that delays happened because the graphics library that was being used recalculated the positions of each of the graphic elements (i.e. the diamond shape, the arrow and the sockets) every time the user made any change (such as adding/deleting/moving rules). To minimize the amount of calculations that needed to be done, the positions of the rules were stored, therefore improving the responsiveness of the tool.

8.3.2.4.1 Related Lines

As with previous versions of GAT, related lines are drawn between sockets that contain concepts in common. For example, Figure 8.10 shows a simple example where *Mercury* is a pre-requisite of *Venus*, and *Venus* is a prerequisite of *Earth*. There is therefore a related line (with a grey concept icon showing 'Venus') between the target socket of the first prerequisite and the source socket of the

second prerequisite. In this case, the related line is designed to assist the author in visualizing the flow of the course.

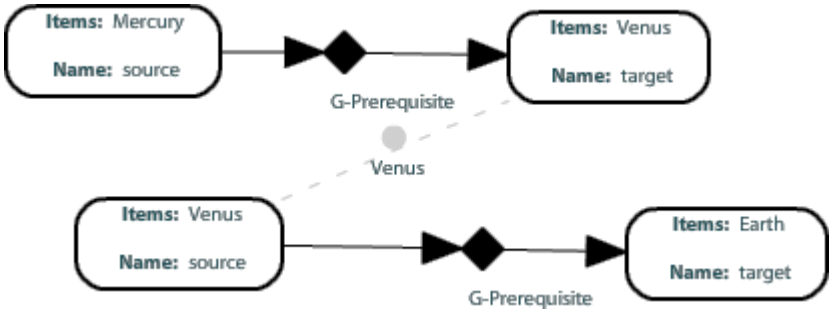

Figure 8.10: A chain of prerequisites with a related line

However, if a *G-Layout* rule (containing all three concepts) was added to the course shown in Figure 8.10, there would be a related line drawn between the *G-Layout* rule's socket and the other four sockets. Since the G-Layout rule merely affects the presentation of the concept and does not have any effect on the sequence of the course, the related line does not provide much information to the author. Furthermore, when there are many pedagogical rules within a course, the course becomes densely populated with related lines. This means it is often difficult to identify which sockets are being referenced by the related lines. For example, Figure 8.11 shows part of the *Milky Way* [4] example with many related lines in the middle that are difficult to understand.

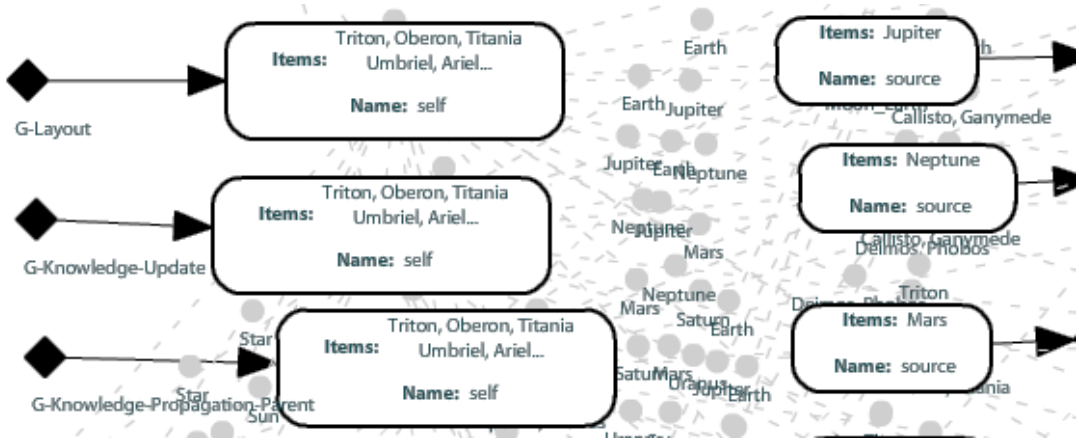

Figure 8.11: Part of a course with too many related lines

In addition to the cognitive overhead, the positions of each related line needed to be recalculated every time the screen needed to be redrawn (as described in Section 8.3.2.4). For this reason, the updated version of GAT adds a checkbox that allows the user to hide the related lines, thus increasing the responsiveness of the tool, whilst removing unnecessary information from the interface.

8.4 Conclusion

This chapter has described the GRAPPLE Authoring Tool, which allows authors to specify adaptation using graphical methods. The chapter has focused on the recent updates to GAT, specifically looking at methods of improving the usability.

Future work could investigate ways of providing adaptive functionality within the authoring tool. A simple way of doing this would be to remember the level of the author and automatically ensure that when a user selects advanced mode, the system remembers the user's preference for advanced mode. This could be made more adaptive by allowing the tool to monitor the types of resource that the user usually uses. For instance, if a user frequently works with static document resources, the system could assume the user is a beginner author with little

experience in adaptation. However, if the user frequently uses XHTML content containing GALE code, the system could assume that the user has more experience in the authoring of adaptation, and could therefore recommend usage of the Pedagogical Relationship Type tool.

Similarly, the experience of using resource templates (as described in Section 8.2.2.1.2) could be enhanced by adding extra features to the domain tool. The domain tool could automatically read the XHTML resource to identify which properties are used by the XHTML resource template. The tool could then prompt the user to ensure that they set these properties to appropriate values. Moreover, a preview function could be added to the tool so the user can instantly see what effect the value of the properties has on the template.

Many of the usability changes described in this chapter are in direct response to the set of imperatives that have been derived during the continual development of the MOT toolset, as documented in previous chapters of this thesis.

There are however substantial differences between the authoring approaches taken by both GAT and MOT. This means that although each tool respects many of the usability principles that have been described in this thesis, they implement the imperatives in different ways. The next chapter compares the two approaches according to the authoring imperatives, and presents the results of an evaluation of both systems, thus allowing the imperatives to be refined further.

9 Different authoring paradigms: Comparison between MOT and GAT

9.1 Overview

This chapter expands on the authoring principles and recommendations that have gradually been introduced over the previous chapters. Specifically, this chapter describes how the two separate toolsets respect the authoring imperatives. The chapter also presents an evaluation between the two systems.

9.2 Revisiting the Authoring Imperatives

Based on the results of previous research, we have gradually refined a set of main authoring principles, as well as a set of recommendations for usability. In this section, we revisit, update and extend the authoring imperatives defined in previous chapters. This chapter specifically looks at how the frameworks (and therefore authoring principles) have been implemented within the authoring tools.

9.2.1 Functionality Imperatives

The following imperatives detail the basic functionality required for all adaptive hypermedia authoring systems.

1) A way of *describing content and annotating content with relevant semantic metadata*: As described in Chapter 2, any system needs to be able to represent the content (resources) to be used in the adaptation process [29], [35], [37]. Similarly, sufficient semantic metadata must be associated with the content.

There is, however, a balance to be sought between having a high level of reusability (by using a high number of semantic metadata labels) or lowering the authoring

effort and thus either generating these labels (semi-) automatically, or allowing for a manageable, low number of labels to be added manually by the author. Please note that the way in which the metadata should be represented or processed is not imposed by this requirement. However the complexity imperatives (below), for instance, would recommend the use of standards where possible.

Furthermore, to ensure the reusability of content (and in order for content to be conditionally presented) the content needs to be divided into smaller pieces. Here, the author must decide what the granularity of each piece should be [51]. Should each piece contain full documents, or parts of documents, or even smaller pieces?

By definition, static content cannot be adapted. Therefore, fine-grained static content can be more flexibly personalised to the needs of the learner than coarse-grained content. Moreover, the user model of the system (which is typically implemented as an overlay of the content), needs to be able to store information about the user's knowledge of a content fragment (since the user model is usually implemented as an overlay model of the concepts [12] – as described in Chapter 2). Therefore, the system will be able to produce a more detailed model of the user's knowledge if content fragments are authored with a fine-grained detail.

However, if the pieces are too small (fine granularity), the authoring process will be time-consuming and complex. Whereas if the pieces are too large (coarse granularity), the adaptation will be limited. Thus, a balance between authoring flexibility and authoring ease has to be found.

2) A way of *describing adaptation*: The adaptive behaviour determines how the static components (such as the content) can be reused and presented to a user (or a group of users) in a variety of different ways, depending on user-related, environment-related, topic-related, pedagogy-related, etc., information.

As with content, it is important to consider the granularity of the adaptation description, in terms of reusing parts of already created adaptation. Here, again, the semantics limit the size of the smallest possible 'piece of adaptivity'. However, common sense (and the limited time of an author) imposes the need to balance between reuse flexibility and ease of authoring. Such a division would also need annotation of the separate pieces with appropriate metadata. Note that we do not impose here any specific way of representing the adaptation. For generic adaptive hypermedia, no mention of pedagogy is made, although clearly an appropriate pedagogical model is a vital goal for adaptive educational hypermedia.

3) A way of *addressing many different adaptation techniques*: Ideally, the tool should be as flexible as possible and allow all users to be able to use a large variety of possible adaptation techniques. Brusilovsky [13] defined a taxonomy of techniques that can be used to provide adaptation. More recently, Knutov [20] refined Brusilovsky's taxonomy based on techniques exhibited by more recent adaptation systems, particularly redefining some of the presentation techniques. In practice, some of the techniques will be more difficult for users to author than others.

These functionality requirements are minimalistic, representing the basics on which all adaptive authoring systems are built. As long as there is a way of expressing

adaptation, and managing and labelling content, adaptive hypermedia can be produced. More advanced authoring systems can also add other layers to separately express other models (such as the user-, learner-, pedagogical-, presentation- models, etc.).

9.2.2 Complexity Imperatives

The following imperatives are designed to simplify the process of authoring courses for adaptive hypermedia. They are not completely independent, in fact, they build on each other.

1) *Separation of concerns*: To ensure reusability and flexibility, the static content (e.g., domain model or content) should be kept as a separate challenge to the authoring of adaptive content (e.g., the adaptation strategy) – as explained in Section 2.9.1 This imperative was also one of the original imperatives from Section 5.2.1 CI1.

2) *Reuse*: To lower the complexity of the authoring process, it is essential to allow authors to reuse existing work (created by themselves and by others). Ideally, the system should allow each element of the authoring process to be reused by other courses. This includes content, links, adaptation descriptions, and pedagogies.

Granularity (as recommended by the functionality imperatives) also has a major role to play in how reusable the content will be.

3) *Clear definition of roles*: The separation of concerns principle allows users to specialize on particular parts of the authoring process. To fully demonstrate this to

the user, the system should provide separate operating modes, for example '*Beginner*' and '*Advanced*' or '*Author*' and '*Programmer*'.

4) *Use of frameworks*: As described in Section 5.2.1 CI2.

5) *Use of standards*: As described in Section 5.2.1 CI3.

9.2.3 Support Imperatives

The following imperatives describe how an authoring system can support the author through more intuitive techniques.

1) *Simple access* to pieces that need to be reassembled for different types of adaptation: The system should allow the author easy access to functionality such as reordering, labelling, copying & pasting, linking and editing (e.g., of content, strategies, etc.). This was described in Section 5.2.2 SI1.

2) *Shallow learning curve*: As described in Section 5.2.2 SI2, the system should be easy to learn, and should allow beginner authors to use a wide variety of features in simple ways.

3) *Familiarity*: As described in Section 5.2.2 , the system should be familiar to the author, thus use as many conventions as possible (functionality, etc.) which are familiar to the author. In particular, the system should provide:

a) *Consistency with existing applications and systems*

b) *Interoperability with familiar course creation formats*

c) *Interoperability with familiar course creation standards*

d) *Interoperability with familiar course creation systems*

4) *Coherence*: The authoring tools should provide a coherent toolset. So rather than requiring the user to use a variety of separate tools, the toolset should maintain a consistent '*look & feel*'. This provides a greater integration between components of the tool.

5) *Visualization*: The authoring system should provide a clear and consistent way of displaying course content and adaptation strategies, via simple and familiar paradigms. Moreover, the system should clearly show a representation of the overall course. Additionally, an authoring system could provide various views for the different parts of the authoring process, for the different levels of detail, etc.

9.3 Applying the Authoring Imperatives

This thesis has described two toolsets that allow the user to author adaptive hypermedia courses; the GRAPPLE Authoring Tool (GAT), and the MOT toolset. Although both methods strive to allow non-technical authors to achieve the same goal (i.e., create an adaptive course), the tools are implemented in slightly different ways. The fundamental difference between the two tools is that GAT intentionally tries to make the authoring process a visual experience, whereas MOT attempts to keep the authoring process simple by using more familiar hierarchical paradigms, such as that used in a file manager (as described in Chapter 3). This section defines the main differences between the two tools with respect to the authoring imperatives described in the previous section.

9.3.1 Functionality Imperatives

9.3.1.1 Describing Static Content

In MOT3.1, each concept can have a number of attributes that describe the concept in a different way (e.g. using a different medium). Each attribute has a type name (such as *text*, *image* or *video*). Attributes are written within MOT3.1 using the HTML editor, and are stored in the MOT3.1 database before being exported into the CAF XML format.

In GAT, each concept has at least one resource associated with it. Each resource is the URL of a piece of learning material. This means that the URL (rather than the content) is exported as part of the CAM model. Instead of type names, it is possible for the user to specify their own metadata properties for each resource. However, only one learning resource for each concept can be delivered to the user at a time, and the metadata can only be addressed by using GALE code.

GAT resources only allow users to provide the URLs of web documents, rather than editing content directly within the tool. This means that if an author wishes to create their own material, they must create it in an external editor, and make their own arrangements for uploading it to a separate web host.

If GAT users were to create Domains that were at least as complex as the average domain map reported by users of MOT3.1 in Chapter 7, this means that GAT Domains will contain at least 27 concepts, and would therefore require the author to upload over 27 documents. Of course, this would assume that each concept only had a single resource, although many adaptive strategies (such as visual-verbal) require each concept to have different content alternatives. If a visual-verbal style

course was to be created using GAT based on a domain with 27 concepts, the author would have to upload 54 separate pieces of content. It is therefore likely that (on average) each GAT resource will have much more coarse-granularity static content than that of a single MOT attribute[44].

9.3.1.1.1 Annotating Static Content with Relevant Semantic Metadata

In MOT, labels and weights are added to sublessons using the *Goal Model* tool. When the adaptation strategy runs for each concept, the strategy checks the value of the label/weight of the concept and then chooses the appropriate action. This allows the entire strategy to address concepts that have been labelled in a particular way.

In GAT, labels are applied by populating the sockets of pedagogical relations. Rather than applying a label that can be accessed globally by the strategy, the sockets in GAT describe how a particular pedagogical relation should treat the concepts.

As explained in Chapter 8 GAT also allows '*properties*' to be specified for concepts (at the Domain model level). Properties are distinct from resources (in GAT) or attributes (in MOT) in that they allow a metadata type to be specified. Such metadata can be queried by the adaptation strategy, allowing different rules to be triggered depending on the metadata of the chosen concept. However, such

[44] XHTML based resources can contain GALE code, which allows fragments of content to be shown/hidden depending on various conditions (such as Domain properties, and User Model variables). However, this would require a more technical knowledge of GALE code and is not directly supported by GAT.

behaviour needs to be specified using GALE code and is therefore not supported by GAT.

9.3.1.2 Describing Adaptation

In the MOT toolset, the adaptation is completely specified within the LAG file. The author must create a LAG file (optionally using PEAL/PEAL2) to describe the adaptation strategy.

In GAT, the author must populate the course with instances of PRTs, and then populate the sockets of each PRT with concepts. The adaptation code is written within the PRT, allowing multiple pedagogical relations (of a variety of types) to be combined.

One of the main differences between the two authoring paradigms concerns the granularity of the adaptation. In GAT, adaptation specialists write small fragments of GALE code within the definition of a PRT. However, in MOT the adaptation is written in a larger LAG file – leading to a much more 'coarse-grained granularity' of adaptation code.

9.3.1.3 A way of addressing a variety of adaptation techniques

The final functionality imperative aims to ensure that the adaptive hypermedia authoring system can allow the author to specify a variety of different techniques. The available adaptation is determined by the functionality of the delivery engine and the authoring tools. An analysis of the adaptation available for each paradigm is described in Section 9.5 .

9.3.2 Complexity Imperatives

9.3.2.1 Separation of Concerns

One of the main benefits of the separation of concerns principle is that it allows different authors to concentrate on the area of the authoring process that they specialize in. In terms of both MOT and GAT, the separation of concerns principle provides three rules:

- Domain content should only contain information about the static content – i.e., the factual teaching material;
- Pedagogical metadata should only contain information about the structure of the course – providing an instantiation of the domain concepts by describing how they should be taught;
- Adaptation should be expressed in a way that does not inherently bind itself to a particular course or domain.

An application of the separation of concerns principle would recommend that each of these three tasks should be achievable by separate authors, who can specialise in their own authoring field (Domain, Metadata or Adaptation) in order to collaborate on the creation of an adaptive course. This principle also promotes reusability as described in the following.

9.3.2.1.1 Separating the Domain from the Course

In the MOT toolset, Domain content only contains the factual attributes that describe each concept, and the relationships between those concepts. However, in GAT, Domain concepts contain properties such as '*order*', which are used by the G-Layout PRT to generate the order of the navigation menu for the course. This

means that the author would need to explicitly change the properties in the domain to change the structure of the course.

Similarly, in GAT the structure of the navigation menu is directly inherited from the 'parent' relationships in the Domain[45]. A similar situation occurs in MOT in that when the author converts a Domain Map into a Goal Map, the conversion process automatically preserves the hierarchical structure of the Domain Map into the initial structure of the Goal Map. However, the author of the Goal Map is able to rearrange the structure of the Goal Map hierarchy (and therefore the structure of the navigation menu), without affecting the original Domain Map. This extra stage gives the course author a chance to modify the layout of the navigation menu, however GAT does not provide an option for this intermediate step.

In some ways, the GAT method of linking the Domain with the final structure of the course simplifies the authoring process, in that the design process happens entirely within the Domain tool. However, if an author wishes to reuse a concept from another author's domain, they would have to copy the domain and manually edit the parent relationship and the order property in order to choose the concept's position in the new hierarchy.

[45] In previous publications (before the G-Layout PRT was finalized), the 'is_part_of' relationship is instead of 'parent'.

9.3.2.1.2 Separating the Adaptation Strategy from the Domain and the Course

As described above, to fully implement the separation of concerns principle, the adaptation strategy also needs to be considered separately from the Domain content and the course.

The MOT toolset achieves this by separating the LAG strategy from the CAF content. The CAF file contains separate elements for the Domain model(s) and the Goal model, whilst the LAG file contains only the adaptation description. This prevents the adaptation author from needing to be concerned about the Domain content or the structure of the Goal model. Of course, in order to create a working strategy, the Goal model creator must communicate with the adaptation author to ensure that the correct pedagogical labels are used.

By contrast, there are two stages in GAT where adaptation can be authored.

9.3.2.1.2.1 GALE code within PRTs

The PRT tool allows adaptation authors to define adaptation in terms of how the course should treat the contents of the sockets. Usually, the adaptation author is able to write a rule that is generic, and does not need to refer to any particular domain concepts, thus showing a clear separation of concerns. If necessary, the author could write GALE code that assumes particular properties or concepts exist within the course, for instance, if the author was writing a PRT that chose a particular property (from a concept in one of the rule's sockets) to display based on the value of a user model variable. Such a rule would have to assume that the properties existed within the concept.

9.3.2.1.2.2 GALE code within XHTML resources

As described in the previous chapter, XHTML resources (specified at the domain model level) can contain GALE tags. This is directly against the separation of concerns principal, since it binds the adaptation code into the static content.

9.3.2.2 Reusability

Granularity also has a role to play in reusability. As described by the support imperatives, there are two types of content authoring where granularity is important. If content is written with granularity that is too coarse, this limits the possibilities for reuse.

Reusability is especially important when authors specialise and collaborate on the creation of an adaptive course. There are two primary uses for reuse.

9.3.2.2.1 Reuse within the same layer

Some authors may wish to copy static content for use within more than one course. MOT and GAT both directly support the copying and pasting of concepts between two different domains – thus promoting the reuse of static content. Moreover, both tools allow such content to be reused by users other than the original author – thus promoting collaborative authoring.

GAT's use of external references provides a very simple approach for the reuse of static content in that a domain author can (manually) copy the URLs used by another domain map. MOT also allows users to '*link*' to concepts from other domain maps, rather than directly copying the concepts. This means that if the source concept (the concept being linked to) changes, the update will be instantly reflected in the target domain map.

In the case of MOT, the interpretation of the link is performed at design time (when the CAF file is exported). This allows the author to check that the source concept still contains suitable information before deploying the finished course. However, using GAT's approach the domain author must trust that the web resource remains suitable (and doesn't get changed by the resource's author), throughout the entire period that the course will be used. In some ways, this may allow the GAT approach to provide more up-to-date material (the web resource's owner can update the content, without the course author needing to perform any updates) however, reuse of resources in such a context would remove a lot of control from the domain/course author.

In the MOT toolset, PEAL provides a code library to allow adaptation authors to reuse fragments (or whole strategies). Adaptation authors in GAT can clone an entire PRT (or copy parts of the GALE code) and edit the behaviour accordingly.

9.3.2.2.2 Mixing and matching between different layers

In MOT, authors can create a new adaptive course based on an existing Domain by exporting the Domain to a new Goal model. The course author can then choose an appropriate strategy, and start adding metadata. The course author can then rearrange the Goal model; add concepts from other Domains or sublessons from pre-existing Goal models, or delete any unnecessary sublessons.

By contrast, an author in GAT can create a new course and choose which concepts (from various pre-existing domains) should be included in the course. However, because of the lack of separation of concerns between the domain and the course

layers, the course author will be unable to change some features that are determined by the Domain's metadata, such as the layout of the course.

Similarly, the granularity of the adaptation affects reusability. MOT allows for reuse of code snippets, using the PEAL editor, but mostly encourages reuse of whole strategies, whereas GAT divides the adaptation into reusable pedagogical relationship types (PRTs), of low granularity.

PRTs allow code to be written in smaller, reusable chunks that are later instantiated with concepts and linked together in the course tool. GAT aims to allow non-technical authors to create a whole adaptive course using these 'building blocks'. Although the small adaptation rule types (PRTs) are designed to be reused within many different courses, the combination of these adaptation rules is specific to the chosen domain. This means that if an author creates a large pedagogy (using many different PRTs) based on a particular topic, the author cannot use the same pedagogy to teach a different topic without starting again.

The opposite situation occurs in MOT, where the adaptation is specified in a whole LAG file, allowing the entire pedagogical strategy to be reused by different goal models (and therefore domain models). However, since the granularity of the adaptation strategy is so large, non-programmers can only reuse the entire pedagogical strategy, rather than having to build the strategy from its basic elements.

To enable content to be reused, PEAL2 supports both a private and a shared space for storing adaptation programs or snippets. GAT allows content to be public,

private or read-only to other authors. In MOT, all content is readable by all authors, but only writable by the original author.

9.3.2.3 Clear Definition of Roles

In order to ensure that as many teachers as possible can contribute to the authoring process for adaptive courses, the authoring system needs to make it clear how the authoring process can be separated into different roles.

9.3.2.3.1 MOT

The various roles that need to be performed as part of the authoring process are detailed in Chapter 6. Concretely there are three main roles in the authoring process using the MOT toolset:

1. A subject specialist authors content into a domain map (using MOT)
2. A pedagogical expert adds pedagogical metadata to the content by exporting the domain map into a goal map (using MOT)
3. A programmer creates a LAG strategy (using PEAL2)

If necessary, these three stages can be carried out by three different people. These people can either collaborate with each other on the authoring of a course, or reuse pre-existing content that was created for other courses.

In practice, it is likely that the three stages will be performed by fewer than three people. For instance, the pedagogical expert might be capable of writing a strategy, or the domain author might prefer to specify their own goal metadata.

The idea of sharing strategies via a university web page[46] is designed to encourage this separation of concerns (and therefore roles), allowing MOT users to choose a pre-existing pedagogical strategy without the need to learn LAG code. Similarly, CAF files can be published online, allowing for the static content to also be swapped between educators. Moreover, all domain and goal model content is readable by all other MOT users, allowing the possibility for authors to base courses on pre-existing material.

9.3.2.3.2 GAT

The same three stages (Domain, Metadata, Adaptation strategy) can also be applied to GAT. As described in the previous chapter, GAT attempts to provide a clear definition of roles by providing an '*Advanced Mode*'. This prevents inexperienced users from accessing the Pedagogical Relationship Type tool, yet still allows them access to the creation of Domains and the creation of Courses.

In GAT, domain content can contain both course metadata (properties) and content adaptation (XHTML GALE code[4], [26]). Due to the difficulties with reuse explained above, it is likely that the course author will need to (at least) modify the domain model, to ensure that the domain properties work with the pedagogical rules within the course.

9.3.2.4 Use of Frameworks

The MOT toolset is based directly on the LAOS [35] framework. The adaptation model for MOT is specified using the LAG [119] framework. Currently, this

[46] Such as http://prolearn.dcs.warwick.ac.uk/strategies.html

framework is implemented by the LAG language, but (depending on the capabilities of the adaptive delivery engine) the language could be changed to anything that was capable of following the LAG framework such as LAG-XLS [19].

The GRAPPLE project inherits many of its features from the AHAM [29] framework, but also inherits features from the LAOS framework [37]. Due to GALE's architecture it is also theoretically possible to replace the GALE code interpreter with an interpreter for another adaptation language [26].

9.3.2.5 Use of Standards

Although there are no defined adaptation standards, both MOT and GAT use more generic web-based standards. In the case of MOT, hypermedia is usually stored in HTML. However, this is merely to simplify the process for the user. Content attributes could theoretically be written in any language (such as plain-text or XML). Similarly, ADE [50] could be easily modified to ensure that the correct MIME-type is delivered to the end user. The main difficulty with using a non-HTML standard is that the content fragments would have to be reassembled in such a way that the output file was coherent. For HTML, the coherency of the output file is a relatively trivial thing to ensure, however for more complex formats such as image, audio or video files there might be more complicated headers (such as transport-streams or checksums) to consider.

GAT allows users to refer to learning materials using any valid URL. GAT's Domain representations are also stored in IMS-VDEX[47] format.

[47] http://www.imsglobal.org/vdex/

Additionally, MOT uses e-learning standards, such as import from IMS-CP, IMS-QTI and SCORM [87]. Furthermore, MOT uses 'de facto' standards, such as PowerPoint [92] and WikiText. Thus, the MOT approach is currently more advanced in the use of standards. However, both approaches have proposed portable export languages which could lead to future adaptation standards, such as the CAM XML language [43] in the case of GAT, and the CAF XML [3] and LAG [53] language in the case of MOT.

9.3.3 Support Imperatives

9.3.3.1 Adaptive Functionality

Basic adaptive functionality exists in MOT3.1 by providing context-sensitive menu options. For instance, when selecting a domain, the submenu automatically changes to domain-specific options, and when the user edits a concept the submenu changes to concept-specific options. Similarly, when a sublesson is selected in MOT3.1, the centre section of the layout (see Chapter 7) changes to allow labels to be added to the sublesson. Such changes provide a visual cue to the user about what actions can be performed with the selected item(s). Furthermore, PEAL2 has features such as automatic completion of code, allowing for support for the more advanced authors.

By contrast, GAT only provides basic adaptive functionality by using greyed-out menus where appropriate. For instance, if there are no valid CAMs currently available, the *Deploy* menu will be disabled – this is analogous to *Link Disabling* in Brusilovsky's Taxonomy[12], [13].

However, in the case of both MOT and GAT, such adaptive behaviour is limited to the context of the material being edited at the time. Neither toolset provides functionality that adapts to the need of user. Such functionality could potentially be implemented, especially with regard to the clear *separation of roles* that occurs within these tools.

9.3.3.2 Simple Access

As described in Chapter 7, MOT3.1 implements various search functions for concept maps, concepts, and goal maps. Similarly, the tree-based interface attempts to simplify the process of reordering, labelling, copying & pasting and linking. As mentioned in Chapter 7, MOT3.1 introduces a new search algorithm for related concepts, based on TF-IDF[127], [136]. This improves search accuracy, by weighting search terms according to their frequency, and is designed to assist authors with finding relevant content (e.g. for reuse).

GAT also allows for various search functions for its domains, pedagogical relations and courses, although since resource content in GAT is not stored within the tool, it would be more difficult to implement a tool that searches within the content of domain concepts.

9.3.3.3 Shallow Learning Curve

Both MOT and GAT aim to provide a shallow learning curve, but their approaches are different. Each system divides complex functionality into simpler basic functions. The *Clear Definition of Roles* principle is designed to prevent the user from becoming overwhelmed, and this is implemented through the separate tools within each toolset. For instance, beginner users need not be concerned with the

specifics of writing adaptation code. However, such separation is only implemented at quite a coarse level, for instance entire tools are hidden from the user. As described above, one of the ways that this could further be improved is through the use of adaptive functionality, which would encourage teachers to *learn* about how to use more advanced features of the tools, rather than simply using the basic parts that they already know how to use.

9.3.3.4 Familiarity

9.3.3.4.1 Consistency with existing applications and systems

The familiarity is addressed in each system by using well-known computing paradigms. For example, MOT represents hierarchical domain and goal map structures using a tree control. This extends on the hierarchical tree file-folder metaphor that is common on many file managers as described in Chapter 3.

GAT has a graphical approach, using less familiar user interface controls. However, the graphical display does follow reasonably simple paradigms, such as directed graphs and flowcharts to display domains and courses respectively. The drag-and-drop methods of creating (and selecting) the elements of these tools will be familiar to users who have used graphical drawing tools.

9.3.3.4.2 Interoperability with familiar course creation formats

MOT3.1 expands on MOT3.0's ability to import content from any MediaWiki, and parses the article's WikiText. As with MOT3.0, MOT3.1 allows authors to import presentations, allowing domain models to be automatically populated with concepts containing attributes with an image and text from presentation slides. Rather than allowing users to edit content directly, GAT allows authors to link to

external web resources. This means that GALE (and therefore GAT) can utilize any document that can be displayed by a web browser. However, no specific import or editing of these resources is supported.

Maurice Hendrix's thesis [46] (p161) quotes a comment from the initial evaluation with GAT in 2009, which said: *"Importing outside domains into the DM component would be very beneficial, for example from ontologies from outside the tool, from IMS-LD or from other standards. This would also increase reusability"*.

This comment strengthens the findings from Chapter 4, showing that users appreciate the idea of content reuse, especially based on pre-existing content from other formats. Unfortunately, no such importers were developed for GAT.

9.3.3.4.3 Interoperability with familiar course creation standards

MOT supports familiar standards for course creators, such as IMS-CP, IMS-QTI, and SCORM. Both systems support HTML, which is also largely familiar to web users. Thus overall, the MOT approach supports a greater palette of familiar course creation standards, with regards to directly importing content.

9.3.3.4.4 Interoperability with familiar course creation systems

The most familiar course creation systems currently are LMS-based systems, such as Blackboard[48], Sakai[49], Moodle[50], etc. The MOT toolset tackles this by allowing imports from standards supported by these and other popular LMSs. The GAT

[48] http://www.blackboard.com/

[49] http://sakaiproject.org/

[50] http://moodle.org/

toolset supports direct interoperability with some of these popular LMSs via an event bus which connects with the GRAPPLE project toolset[137], which also connects to Sakai, Moodle, CLIX from IMC[51] and the eXact learning solutions from Giunti labs[52].

9.3.3.5 Coherence

GAT provides an integrated, coherent toolset, allowing adaptation to be specified within the same application as the content definition.

Conversely, the MOT approach requires content to be specified and labelled within MOT3.1, but the adaptation needs to be specified in a separate tool (PEAL). This may provide a slightly less coherent experience for authors who need to use both tools, however many users will specialize in only one of the tools.

In GAT, a course can be directly deployed to GALE by clicking on a '*Deploy*' menu item – this is a one-step process. However, in the MOT toolset, users must separately upload XML and LAG files from within the ADE interface.

9.3.3.6 Visualization

Amongst the fundamental differences between the MOT and GAT paradigms are the user interface controls and the ways that information is displayed to the author. GAT employs a visual, graph-based way of authoring domain concepts. This allows concepts to be quickly copied, selected and moved using drag & drop metaphors that are often used in drawing applications. Relationships are displayed to the

[51] http://www.im-c.de/

[52] http://www.giuntilabs.com/

author as arrows that form edges in the graph of concept nodes. Relationships have a name that can be used as metadata such as '*parent*' and '*is-a*'.

These principles are also used in the course tool, which displays the pedagogical rules graphically. The course tool allows the user to quickly drag & drop, insert and copy pedagogical rules, and also allows concepts to be visually dragged from a domain, straight into a rule in a course. Pedagogical rules can also be provided with a default colour, or the author can change the colour of individual instances. This allows course authors to quickly identify different types of rules.

The MOT toolset, by contrast, is much less graphical. The hierarchical display of domain and goal maps uses more standard desktop paradigms, such as the tree control (which also allows drag & drop copying).

In MOT, extra relatedness relations links can be added but these cannot be graphically shown within the overall domain; they can only be shown when viewing an individual concept. The relatedness relations can have a name (as with GAT) but unlike GAT's relationships, the relatedness relations do allow for percentage based weights to be specified (showing how closely related the two concepts are). Moreover, in MOT relationships between concepts can span between domain models – i.e., a concept can have a relatedness relation with a concept from another domain model (even if the domains are created by two separate authors).

By contrast to GAT's visual methods of specifying a course, the MOT toolset uses LAG files written in PEAL, which can optionally be visualized using PEAL2. Here, much of the difference in visualization is due to the issues of granularity (described

above). However, PEAL2 should not be seen as a direct equivalent of the course tool in GAT, since it allows the user to specify complex adaptation, and therefore is also equivalent to the PRT tool.

The PRT tool only provides a simple textbox for GALE code (see Figure 9.1), whereas PEAL provides many more IDE-style features such as code completion and syntax highlighting (shown in Figure 9.2).

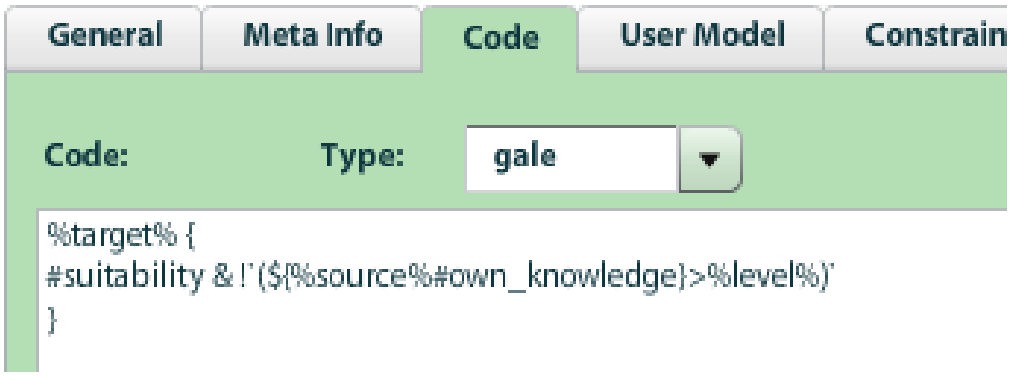

Figure 9.1: GALE code within the PRT tool

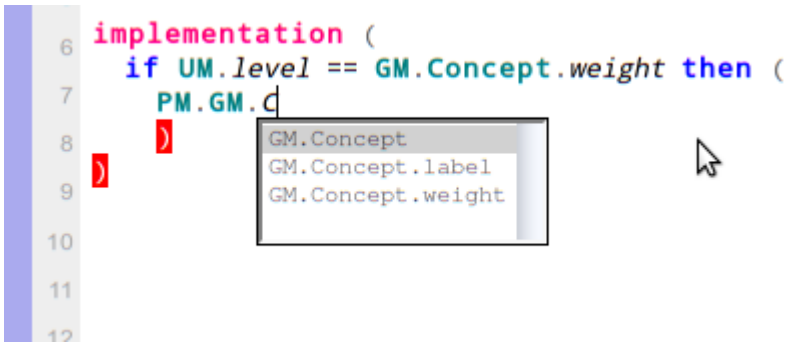

Figure 9.2: LAG Code within PEAL

9.3.3.7 Adaptive Functionality

Neither tool directly adapts to the needs of the individual user; however various parts of the tools adapt to the context. For instance, MOT3.1 aims to minimize cognitive overhead by only offering features that are directly related to the current

task. Specifically (as described in Chapter 7), MOT3.1 changes the menu options depending on whether the user is currently editing a whole domain map or just a specific concept. Similarly, the advanced mode in GAT could potentially be thought of as an adaptive feature (especially with some extra modifications, such as those described in Section 8.4). Future research could investigate this further by adding more complex assistance to beginner users, and providing more features to the more experienced users.

9.4 Comparing the roles of the individual tools

The previous section has compared the usage of the systems with respect to the imperatives. Although there are effectively three separate tools in each toolset, there is not a clear one-to-one mapping between these tools. Table 9.1 shows a direct comparison of the various terms and features used within the two systems and indicates which part of the toolset is designed to author each feature.

<table>
<tr><th colspan="2">MOT</th><th colspan="2">GAT</th></tr>
<tr><td rowspan="6">Domain Model Tool</td><td>Domain map</td><td>Domain</td><td rowspan="6">Domain Tool</td></tr>
<tr><td>Concepts</td><td>Concepts</td></tr>
<tr><td>Attributes</td><td>Resources</td></tr>
<tr><td>Parent relationships</td><td rowspan="2">Relationships</td></tr>
<tr><td rowspan="2">Relatedness relations</td></tr>
<tr><td rowspan="4">Properties</td></tr>
<tr><td rowspan="3">Goal Model Tool</td><td>Goal map structure</td></tr>
<tr><td rowspan="2">Labels/Weights</td><td rowspan="2">Sockets</td><td rowspan="3">Course Tool</td></tr>
<tr><td rowspan="2">Sublessons</td></tr>
<tr><td rowspan="2">PEAL</td><td rowspan="2">LAG strategy</td><td rowspan="2">Courses</td></tr>
<tr><td>Pedagogical Relationship Types</td><td>PRT Tool</td></tr>
</table>

Table 9.1: Comparing the various features and terminology used in the MOT and GAT toolsets

9.5 Comparing the Available Adaptation Techniques

The previous section has highlighted how each system has different expectations on the role of the author and the amount that the user is expected to learn. The imperative in Section 9.3.1.3 states that the authoring tool should allow the user to utilise a wide variety of adaptation techniques.

The primary usage of the GRAPPLE Authoring Tool is for the author to create courses via a *'learning through information exploration'* process [4]. This process only requires the usage of the Domain tool and the Course tool, and relies entirely on the initially created set of PRTs that were created specifically for the GRAPPLE project (see Chapter 8). For this reason, when comparing the adaptation that can be performed by the author, we consider beginner and advanced authors for each system.

1. *Beginner author:* A non-technical content author, similar to Professor Smith from Chapter 6. In GAT, Professor Smith can only use the domain tool and the course tool with predefined PRTs. She can design domain and goal maps using MOT3.1, but does not know how to program in LAG.
2. *Intermediate author:* A technical content-author – similar to Professor Jones from Chapter 6 – who understands about how to write code with LAG, and only wants to create reusable PRTs that can be shared with non-technical 'beginner' authors.
3. *Advanced author:* An expert programmer who has learnt both LAG and GALE. This author is therefore proficient at creating LAG strategies, but is also confident at creating new PRTs and XHTML files that contain GALE code (a technique for which GAT provides no direct assistance). The advanced author mainly creates code that is used in their own courses, and so does not need to worry about creating code that is generic enough to be reused by others.

Each of the different adaptation techniques requires at least one of the above authors. Table 9.2 explains which of these authors is needed in order to produce each of the separate types of adaptation from Knutov's [20] updated taxonomy.

	MOT		GAT		
	Content Author (reusing common strategies)	Adaptation Author	Domain/Course Author (reusing Pedagogical Rules)	Pedagogical Rule Author	GALE Expert Author
Content Adaptation Techniques (Canned Text, Multimedia)					
Inserting/removing fragments	Yes	Yes	No	Yes	Yes
Altering fragments	No	No*	No	No	Yes
Adaptive Presentation Techniques					
Dimming fragments	No**	No***	No	No	Yes
Sorting fragments	Yes	Yes	No	Yes	Yes
Stretchtext	No**	No***	No	No	Yes
Zoom/Scale	No	No	No	Yes	Yes
Layout	Yes	Yes	Yes	Yes	Yes
Link Sorting/Ordering	No	No	No	No	Yes
Link Annotation	No	No	Yes	Yes	Yes
Combinatorial techniques	No	No	No	No	Yes
Adaptive Navigation Techniques					
Link generation	No	No	No	No	Yes
Guidance (Direct/Global)	Yes	Yes	Yes	Yes	Yes
Link hiding (Disabling/Removal)	Partly****	Partly****	Yes	Yes	Yes
Key	*: To be implemented in a future version of LAG **: Could be simulated by adding a separate similar attribute ***: This has been implemented in a recent version of LAG (September 2012) ****: Internal links cannot occur within the main body of the content, so only the navigation menus can be adapted				

Table 9.2: Comparison of the authoring techniques available in MOT and GAT

Table 9.2 shows that whilst GALE supports a wider variety of adaptation techniques than ADE, many of the techniques require more advanced knowledge of programming. This is primarily because many of these techniques require inline

code (which the MOT toolset does not currently support). Such inline code is hard-coded into the content, making it impossible for a non-expert to reuse the adaptation in their own content, thus compromising GAT's adherence to the separation of concerns principle. Moreover, the GAT tool provides no direct assistance for the creation of such inline code.

9.6 Evaluation of MOT versus GAT

As described in previous chapters, the MOT toolset is used as part of the coursework for the MSc/MEng module '*CS411: Dynamic Web Based Systems*'. Chapter 7 described how MOT3.1 was used by the students as part of a long-term usage study. During this time, a two-hour seminar introduced the students to the GAT toolset. A short demonstration on how to use GAT was provided to the students, and they were then asked to create a basic course using GAT themselves, by using both the Domain and Course tools. The students were shown the basic principles of the PRT tool; however since the creation of new PRTs requires a detailed knowledge of GALE code they were encouraged to only use the pre-defined PRTs – this is in deference to other GRAPPLE evaluations[4]. During the time of evaluation, various pre-created models were already existent in the system, such as the '*Milky Way*' demonstration domain and course[130].

After each test subject had created a course, a demonstrator checked that the course was valid (and that they had therefore understood the principles of GAT), before providing the subject with a questionnaire.

9.6.1 Hypotheses

The questionnaire was designed to test the following hypotheses.

H9.1. It is easier to author content within MOT3.1 than GAT

a. Authors prefer to edit pre-existing content rather than editing from scratch.

b. Authors prefer using an integrated authoring tool.

c. Authors prefer to edit content at high-granularity (i.e. small pieces).

d. In general, MOT3.1 is easier to use than GAT.

H9.2. Concepts are easier to use in MOT3.1 than in GAT.

H9.3. Users prefer the flexibility of concept metadata provided by GAT.

H9.4. Users desire the ability to specify multiple types of relationships (as in GAT).

H9.5. MOT3.1's approach to designing the course hierarchy (via the Goal Map tool) is preferred over GAT's approach which binds the design into the domain map.

H9.6. Users prefer labelling within MOT3.1 over using sockets in GAT.

H9.7. When reusing a strategy, users prefer to reuse entire strategies over creating a new course using smaller pieces of adaptation.

H9.8. When creating a new strategy, users prefer to author entire strategies, rather than writing individual rules.

H9.9. Authors prefer to specify adaptation using the MOT toolset than using GAT.

H9.10. In general (away from the tools)

a. Users prefer graph-based tools for editing concepts.

b. Users prefer high granularity adaptation code (using small pieces as 'building blocks').

9.6.2 Questionnaire Results and Discussion

Due to limitations of the questionnaire software[53] the survey was divided into two sections. For the first section, there 19 respondents, however only 16 respondents completed the first section. The second section was completed by 13 respondents. The full questionnaire is provided in Appendix A.

Many of the questions were grouped (partly to give more structure to the questionnaire, and partly due to the limitations of the software), which meant that many of the questions required more than one answer. For instance, Figure 9.3 shows an example question that has 4 separate sub-questions. At the end of each question, a textbox allowed the respondent to provide some more detailed comments.

[53] The questionnaire was designed using free version of Survey Monkey (http://www.surveymonkey.com/) which only allows 10 questions.

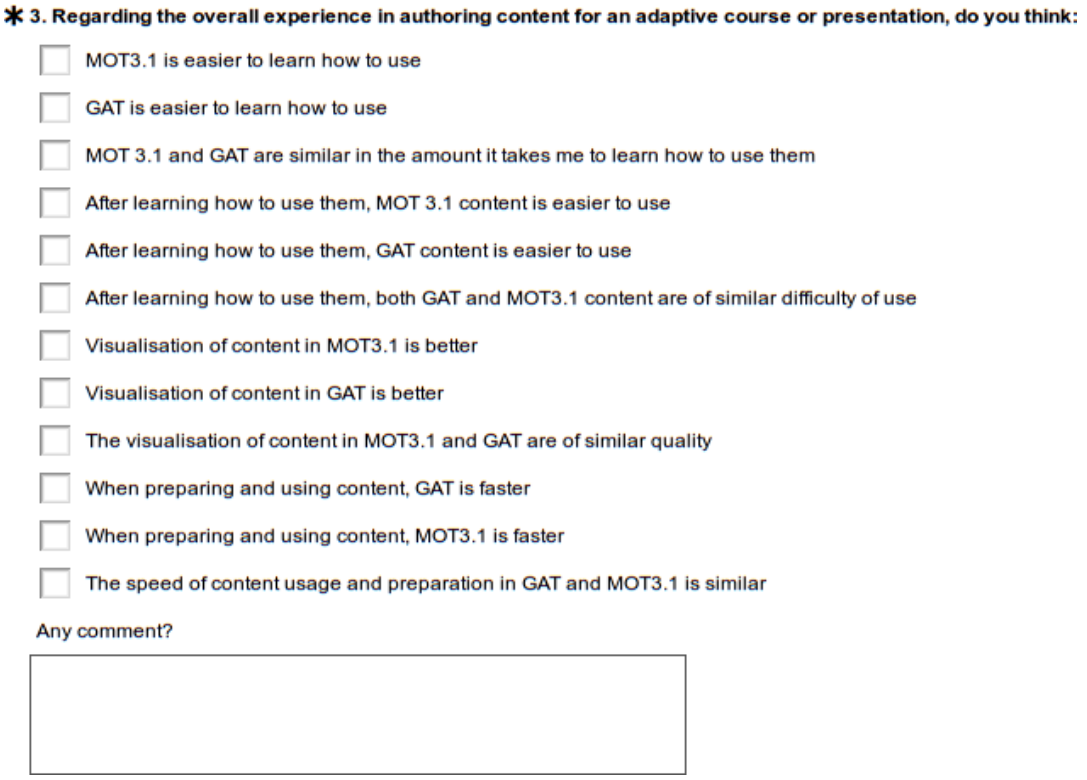

Figure 9.3: A sample question from the questionnaire – with 4 sub-questions

Where a respondent gave a nonsensical answer (i.e., they said they preferred a system but also said they were both the same), the response was removed from the sample for that particular question[54]. In most cases, the questionnaire software ensured each question was compulsory. However, it was impossible to enforce this on questions that allowed the user to select multiple options. Adding a neutral (such as 'similar' or 'don't care') answer to each question ensured that the respondent was actively expressing an opinion, rather than simply skipping the question.

Due to the categorical nature of the data, a χ^2 (Chi-Squared) Goodness-of-Fit test was used, against an expected uniform distribution (33.33% for each value) across the three possible answers (e.g., 'MOT', 'GAT' and 'Similar'). However, when

[54] This situation occurred on 8 occasions (across different questions)

applying this test only two statements can be seen to be statistically significant at a 5% confidence interval. Similarly, if all votes for 'Similar' are removed from the sample, and a separate χ^2 Goodness-of-Fit test is used against another uniform distribution (50% for each value), only two statements are statistically significant. This lack of statistical significance is largely due to the small number of test participants.

9.6.2.1 Content authoring (H9.1)

The first question had five separate options (shown with the frequency of responses in Table 9.3) to which the students were invited to select as many as they wished.

When authoring an adaptive course or presentation, do you prefer to use…	***Frequency***
Pre-existing content only	1
Pre-existent content which can be further edited	10
Editing from scratch	5
Editing HTML files in any HTML editor (like in GAT)	11
Using a dedicated, integrated editing tool (like in MOT3.1)	12

Table 9.3: Frequency of responses to Question 1

This question asks for the opinions about two related issues:

9.6.2.1.1 Should content be based on pre-existing content? (H9.1a)

Table 9.3 shows that only 1 out of the 19 respondents (5.26%) liked the idea of creating a course based on pre-existing content – this would equate to using GAT to only reference other people's resources. A similarly small number of respondents (5, 26.31%) said they preferred to edit content entirely from scratch, with most people wanting to edit pre-existing content. This would appear to support H9.1a,

but it also indicates the importance of the ability to import content from other sources – potentially using the principles that were defined in Chapter 4.

9.6.2.1.2 Should the content editor be integrated into the authoring tool? (H9.1b)

Seven of the respondents said that they preferred both "Editing HTML files…" and "Using a dedicated, integrated editing tool". These respondents remain in the data presented in Table 9.3, since a vote for both choices would imply that users have no clear preference. This lack of a clear preference appears to be supported by the fact that there was no clear preference shown by the remaining respondents. Therefore, there is no evidence to support H9.1b.

There are a few reasons that might indicate why this result was inconclusive. One of the respondents who chose the "Editing HTML files in any HTML editor (like in GAT)" as opposed to an integrated editor provided the following comment in response to this question:

"*I generally prefer to know exactly what is happening with the model, so user interfaces usually get in the way, requiring special options to change parts which are easy to edit in the code. User interfaces do have an advantage with large models, but must be very polished (especially regarding keyboard navigation, responsiveness of UI, no data-loss bugs)*"

This indicates that authors might prefer to use a stable and mature HTML editor (that they are already familiar with) rather than being forced to use the editor that is integrated into the tool. However, it must also be noted that these users are Computer Science students and are therefore comfortable with the idea of

uploading content for subsequent reuse within an adaptive course. Many non-technical authors may be less familiar with the process of acquiring web-space (indeed, for security reasons, this may be impossible within their educational institution) and uploading their own content. In such a scenario, the integrated authoring tool might be more convenient.

9.6.2.1.3 Granularity of Authoring (H9.1c)

There was also an inconclusive result pertaining to the granularity of content. Out of the 19 students, 8 (42.11%) said that they preferred using high granularity (as in MOT3.1) with 6 (31.58%) saying they preferred low granularity content (as in GAT). The remaining 5 stated that they had no preference.

One comment stated that it is "*Easier to write and think with low granularity, though I can see the benefit of high granularity.*" Another said that it depends on the type and complexity of the content, whilst another comment pointed out that high granularity provides a higher level of adaptability.

Section 9.3.1.1 described some of the issues surrounding the different levels of granularity of static content. Although high granularity does allow a greater detail to be specified when applying adaptation, low granularity usually makes the authoring process much easier. There is therefore no evidence to support H9.1c.

9.6.2.1.4 Ease of Use (H9.1d)

10 out of 18 of the students (55.56%) stated that GAT is easier to learn than MOT3.1, with 5 users (27.78%) saying that MOT3.1 was easier to learn, and 3 users (16.67%) having no preference. However, 9 out of the 18 (50%) stated that after the initial learning process, MOT3.1 is easier to use, with only 4 saying that GAT was

easier to use and the rest with no preference. Although these statistics are not statistically significant, this suggests that MOT3.1 has a steeper learning curve, but is easier to use after the user has spent time understanding the MOT authoring paradigm.

With regard to visualization of content, 12 out of the 19 students (63.16%) preferred GAT, with only 5 preferring MOT and the other 2 undecided. This is statistically significant at the 5% level when performing a χ^2 (Chi-Squared) Goodness-of-Fit test compared with an expected distribution of 33.33% for each option.

However, with respect to the speed of preparing and using content in each system, there was no clear preference, with 6, 5 and 8 students choosing 'GAT', 'MOT3.1' and 'similar' respectively.

One comment said that "*Both have issues with responsiveness which severely limits their speed of preparation, but because MOT takes a less interactive approach (generally requiring less actions but more forms to set parts up), it is faster to use overall"* whilst another user said: *"GAT seems to be more fiddly with lots of things to know and lots of steps to do. MOT is more to the point. I find MOT's tree view clearer than GAT's graphical view".* Overall, the above statistics show no evidence to support H9.1d.

9.6.2.2 Using Concepts (H9.2)

There were no significant results with respect to the usage and visualization of concepts within the systems, providing no evidence to support H9.2. One comment

said that "*comparing the visualisation is difficult; GAT provides more holistic information which is better, but this is at the cost of a very messy interface and for big models, a very long render time*".

9.6.2.3 Concept Metadata (H9.3)

8 out of 16 (50%) of the respondents said they preferred the idea of having multiple properties for domain concepts (as demonstrated by GAT). Only 2 out of 16 (12.5%) said that they preferred using the *type* of an attribute as a property, with the remaining 6 respondents having no clear opinion.

However, it can be seen that 8 out of 16 (50%) of the students thought that GAT properties were easier to use (after the initial learning period). Moreover, 7 out of 16 (43.75%) thought that GAT was better at visualizing domain properties than MOT3.1, which had only 2 out of 16 votes (12.5%), with the rest showing no preference.

Similarly, 7 out of 15 respondents thought that GAT was faster for preparing and using properties, whereas only 1 thought that MOT3.1 was faster, and the other 7 showed no preference.

The above statistics do not allow H9.3 to be accepted; however overall there was a slightly more positive response towards the way that GAT handles domain properties than towards the way that MOT3.1 handles properties. Specifically, MOT3.1 only allows attributes to be defined for a concept, but it allows the attribute to have a particular *type*. The *type* is analogous to GAT's domain properties, since it allows the author to specify metadata about the content.

However, adding a wider variety of properties to MOT's domain model would allow adaptation strategies to address certain concepts according to the concept's metadata.

9.6.2.4 Relationships (H9.4)

8 out of 16 respondents (50%) said that they preferred to have multiple relationships within the domain model (as exhibited by GAT). The remaining 8 respondents were divided equally between the "hierarchical and relatedness relations" (25%) as exhibited by MOT3.1, and "no strong opinions" (25%).

Similarly, a statistically significant 9 out of 12 respondents (75%) said that the visualization of properties of relationships in GAT is better than that in MOT3.1 (2 votes, 16.67%). This refers to the fact that the edges of the domain graph in GAT display the relationship type.

The ease of learning how to use domain relationships was valued equally, with 4 respondents choosing the options 'MOT', 'GAT' and 'similar' respectively. In terms of ease of use (after learning) there was a fairly large preference for the usage of MOT3.1 DM relations (7 out of 11, 63.64%) over GAT (3 out of 11, 27.27%) with 1 person (9.09%) saying that they're similar.

There was almost no difference with regards to the speed of using the systems (5, 4 and 2 for 'GAT', 'MOT3.1' and 'similar' respectively).

The above statistics combined show that there does appear to be a small preference for the way that relationships are catered for within GAT. However, (mainly due to the small sample size) there is no statistically significant evidence to

show that users desired the ability to use multiple relationships, therefore not confirming H9.4.

9.6.2.5 Design of the Course Hierarchy (H9.5)

8 out of 16 respondents said that they preferred the option of reusing the same domain for different goal models (as in MOT), whereas 5 preferred to keep the order of the concepts bound to the domain model (as in GAT).

Only 4 out of 16 respondents (25%) said that there should be an intermediary step (e.g., the goal map tool) to decide whether the hierarchy defined in the domain model should be used within the goal map tool. 6 out of 16 respondents said that it should be possible to directly relate the domain concept hierarchy to the final course hierarchy (like in GAT). The remaining 6 respondents had no clear preference.

There is therefore no evidence to support H9.5. These results are rather surprising, since the method of designing a hierarchy in GAT is not graphical. However, one of the comments said: "*Reusing the DM isn't important to me*", so it could be that since the students experience with GAT was limited to a single session, they were thinking primarily about how to design the domain model so that they could create a single course. One way of investigating this issue further would be to ask the students to base their course on somebody else's domain model, which would mean that they had no choice but to use the layout specified by the original domain.

9.6.2.6 Labelling versus Sockets (H9.6)

MOT and GAT have different ways of applying pedagogical metadata. In MOT3.1, (multiple) labels are used, whereas in GAT concepts are inserted into sockets. The sockets can be thought of as a label, since they tell the adaptation strategy (the GALE code within the PRT) how to treat the concepts. An alternative way of specifying metadata within GAT would be to specify properties (and values) within the domain, however this would compromise the separation of concerns principle (as described earlier in this chapter).

The students showed no preference between the labelling methods, with 6 out of 12 (50%) students preferring to use labels, and 5 out of 12 preferring to use sockets. One respondent chose 'Labelling… via the DM'. There is therefore no statistical evidence to support H9.6.

9.6.2.7 Reusing Strategies (H9.7)

The students were asked to complete the phrase "*In terms of granularity of strategy reuse, I prefer reusing…*" with each student allowed to choose more than one option. The majority of users (7 out of 11, 63.63%) said they preferred to reuse small pieces of adaptation. There were two votes for each of 'whole adaptation strategies' and 'specific pieces of adaptation'. There was also a single vote to say that each technique was of 'similar value'.

Similarly in the context of reusing existing material, 6 out of 12 (50%) of the students said that GAT PRT relations are easier to learn how to use. However, only 3 out of 11 (27.27%) of students said that after learning how to use the systems, GAT was easier to use. 4 out of 11 said that PEAL was easier to use (after learning

how to use the systems), and the remaining 4 said that they were both of similar difficulty.

Although not statistically significant, the majority of users (6 out of 11, 54.54%) showed a preference for GAT with regards to the visualization of sub-strategies. Only 1 person said they preferred the way that PEAL allows sub-strategies to be shown, with 4 indicating that they are similar.

This is not an especially surprising result since although PEAL supports the reuse of fragments of LAG code (as described in Chapter 5) it only allows the sub-strategy to be displayed textually. The features introduced by PEAL2 allow the entire strategy to be shown graphically, but this is separate from the code library interface that allows such reuse. The reuse of pedagogical rules within GAT can therefore be more clearly visualized within the course tool.

There is no statistical evidence to support H9.7 (partly due to the small sample size). Moreover, it appears that there is a general preference towards using smaller pieces of adaptation (such as the usage of pedagogical rules in GAT). Research by Javed Khan at the University of Warwick [138] is currently investigating how to display small extracts of LAG code in such a way that they can easily be combined visually by authors to create a larger adaptation strategy.

9.6.2.8 Creating New Strategies (H9.8)

The students were also asked to complete the phrase "*In terms of granularity of strategy writing, I prefer…*", and were allowed to select more than one answer. 7 out of 10 participants said that when creating a new strategy they preferred to

write ‘whole adaptation strategies’ (as exhibited by LAG). 3 users said that they prefer writing specific parts of adaptation strategies, there was 1 vote for small pieces of adaptation and 1 user said they were all of equal value.

This indicates a clear preference for writing whole strategies rather than writing small pieces of adaptation, therefore supporting H9.8.

9.6.2.9 Adaptation specification (H9.9)

Although the users were not taught how to use GALE code, they were introduced to it and encouraged to look at the code within the PRT tool.

The majority of users (7 out of 12, 58.33%) said they thought they could create strategies of greater complexity using the MOT toolset. 3 users (25%) said they thought they could create strategies with greater complexity using GAT, with 2 (16.67%) saying they could create strategies of similar complexity using either tool.

In some ways, this result is unsurprising, since the users had already spent more time using the MOT toolset (approximately 5 weeks compared with a two-hour seminar). However, the students were divided about how easy LAG and GALE code are to use; out of the 11 respondents to the question, there were 4 (36.36%) votes for LAG and 4 (36.36%) for GALE, with 3 (27.27%) undecided. However, there 5 users (45.45%) said that once learned, LAG is easier to use, and only 1 user (9.09%) thought that GALE code was easier to use.

Surprisingly, 6 out of the 11 users (54.54%) thought that the visualization of GALE code is better within the PRT tool than the visualization of LAG code within PEAL. Only 3 (27.27%) people preferred PEAL for visualization of code, with 2 people

expressing no preference. This is a surprising result, because the PRT tool (unlike PEAL) provides no syntax highlighting.

Similarly, 5 out of 11 users (45.45%) thought that the process of writing code in GAT is faster than that of writing code in PEAL. Only 2 users (18.18%) preferred the speed of PEAL, whereas 4 users (36.36%) thought they were similar.

The above statistics are unable to prove H9.9. This could be partly because the sample size was small, but also appears to be based on a misconception about the systems. It is strange that users appear to believe that the visualization of the PRT tool is better than that of PEAL. However, it could be that users misunderstood the question, and thought it was referring to the visualization of PRTs within the course tool.

9.6.2.10 General Authoring Principles (H9.10)

The final question concerned the student's general thoughts about authoring for adaptive hypermedia, and expressly asked them to think "away from GAT & MOT+PEAL". Out of the 11 respondents, 5 (45.45%) preferred graph-based tools for editing concepts and 3 preferred hierarchical tools. The remaining 3 respondents declared no preference.

Similarly, 4 out of 11 respondents preferred a high level of granularity for adaptation strategies (involving small pieces of code), with 3 students preferring low level granularity, and 4 students having no preference.

Neither H9.10a nor H9.10b can be supported based on the results of this final question. There appears to be a small preference for graph-based tools, but further research is needed.

9.7 Conclusions

This chapter has described the differences between the two toolsets that have been investigated by this thesis, and has provided a survey of the available adaptation techniques available through the use of each toolset.

This chapter presents the results of an evaluation between the two systems. The results of the evaluations indicate that each tool has both advantages and disadvantages. Moreover, the qualitative feedback has provided clues about the features that need to be present in the next iteration of the development of MOT. The design, implementation and evaluation of the next iteration are documented in the next chapter.

10 Integration of authoring approaches: Improving MOT based on GAT ideas

10.1 Overview

The previous chapter described the fundamental differences between the approaches exhibited by the MOT toolset and the GAT toolset. This chapter describes a new iteration of MOT, which investigates how some of the best features of MOT can be retained, whilst integrating some of the functionality of GAT.

10.2 Improving the MOT Toolset

The previous chapter highlighted the differences between two adaptive hypermedia authoring toolsets – MOT and GAT. Whilst the questionnaire based evaluation about the differences between the toolsets had not provided many conclusive results, there appeared to be a number of features in GAT that could extend the functionality and usability of the MOT toolset.

This chapter documents the creation of MOT4, which maintains compatibility with the LAOS framework, whilst extending the functionality and flexibility of the authoring tools.

10.3 Requirements

This section describes the basic set of requirements that were identified for the improvement of MOT4.0. These requirements are based on continual refinement of the authoring imperatives (described in Sections 2.9.2 , 5.2 and 9.2), and are designed to address features that were directly requested by the students during the real-world usage (such as the ability to use multiple labels).

10.3.1 Graphical Relationships

Analysis of the coursework that was submitted during the evaluations described in Chapters 5 and 7 has shown that there was only one instance of students using the relatedness relations feature. As described in previous chapters, the hierarchical nature of MOT's domain model limits the extent to which such relations can be visualized. However (as described in Chapters 8 and 9), GAT's domain visualizes all relationships as edges of a graph.

To enable users to more clearly visualize the types of domain relationship that can be created, it was decided to replace MOT's hierarchical domain map with a visual domain graph.

10.3.2 Multiple Labelling

Chapter 7 described how MOT3.1 introduced a way of allowing sublessons to be given more than one pedagogical label. However, the implementation of such multiple labelling was limited, particularly in regard to addressing the weights that belonged to particular labels. Section 7.6.1.3 described how the students desired the functionality of addressing such weights.

The sockets in GAT can be thought of as a form of multiple labelling; each concept can occur in more than one socket, therefore allowing disparate types of adaptation to be applied to the same concept. Moreover, as Chapter 9 described, the design of GAT encourages concepts to contain properties that (due to the unclear separation of concerns) are frequently used to denote properties that are of a pedagogical nature. For instance, if a rollout strategy was to be implemented in GAT, each concept would require a *showatmost* property to be defined within the

domain layer of individual concepts. However, the *showatmost* property is pedagogical rather than factual, and could therefore change depending on the adaptation strategy. The introduction of multiple labels to the MOT toolset ensures that such pedagogical properties can be clearly defined in the goal model, through the usage of multiple labels.

10.3.3 Domain Properties

Some properties, however, may be of a factual nature and should therefore be stored in the domain layer. For example, if a domain was to contain information about historical events with each concept representing a different event, the date could be stored within a domain property. Unlike the pedagogical labels described above, the date is factual, and therefore has no direct influence on the pedagogical strategy. However, a variety of pedagogical strategies could be created to utilise this information; for example authors could use a strategy that presents events in either chronological or reverse chronological order depending on the user's preference.

10.3.4 External Resources

The evaluation documented in Chapter 9 had shown that some users desired the ability to use their own (external) resource editor, rather than being forced to use the WYSIWYG HTML editor that was integrated into MOT3.1.

10.4 Designing the Adaptation Languages

To facilitate the additional features, both the CAF XML format and the LAG language had to be overhauled.

10.4.1 The LAF XML Format

Previous chapters have described how new features have been added to MOT whilst maintaining compatibility with the CAF format (for example, multiple labelling in MOT3.1). However, to maximise the potential of the additional features of MOT4, a new XML format named '*Learning Adaptation Format*' (LAF) was developed.

10.4.1.1 Domain Storage

Code listing 10.1 shows an extract of a LAF representation of a domain. Each domain is assigned a globally unique identifier (GUID). This ensures that (unlike in previous versions of MOT), there is no ambiguity about which domain map is used when importing the file. In MOT1.0 when a CAF was imported that contained a domain map with the same name as a pre-existing domain map, the author would be prompted to rename the domain map. This issue was fixed in MOT3.1 (and MOT4) by internally referring to domain maps using database concept IDs (rather than using XPath as recommended by the CAF specification).

```
<LAF>
  <domain id="71a4dbfc-9ffe-4c4b-9851-69d1b699d0a3">
    <metadata>
      <author>jonny</author>
      <name>XML</name>
      <date>2012-03-21 15:18:19</date>
    </metadata>
    <concept id="0">
      <attribute type="title">XML Topics</attribute>
      <attribute type="keywords"/>
```

```
      <attribute type="text">&lt;p&gt;This course
describes the main topics surrounding
XML&lt;/p&gt;</attribute>

      <metadata>

        <position x="197.59665182765" y="-
53.544829989606"/>

      </metadata>

    </concept>

    <concept id="1">

      <attribute type="title">XPath</attribute>

      <attribute type="keywords"/>

      <attribute
type="text">&lt;p&gt;&lt;strong&gt;&lt;em&gt;XPath&lt;/e
m&gt;&lt;/strong&gt; is a way of selecting groups of
nodes from an XML Document&lt;/p&gt;</attribute>

      <metadata>

        <position x="74.915666545471" y="-
18.438141049076"/>

      </metadata>

    </concept>

    <relationships>

      <relation from="0" to="1" type="is-parent-of"
weight="100"/>

      <relation from="0" to="2" type="is-parent-of"
weight="100"/>

      <relation from=”2” to=”1” type=”uses”
weight=”80”/>

    </relationships>

…</domain>…</LAF>
```

Code Listing 10.1: Extract of Domain LAF XML

10.4.1.1.1 Metadata

The LAF file now contains a *metadata* tag, which currently contains the name of the Domain, the date/time the LAF was exported from MOT, and the author's MOT user name.

10.4.1.1.2 Concepts

Concepts in LAF are stored with attributes in a very similar way to concepts in a CAF file. However, rather than storing concepts in a hierarchical XML tree (like CAF), LAF places all concept definition elements within a *domain* element.

10.4.1.1.2.1 Concept Positions

As described in Chapter 2, there is no semantic meaning to the order of concepts within a domain. However, previous research on MOT1.0 had shown that users were disorientated when the order of a domain was not preserved between sessions. This is even more important with a graph-based display, since the position of concepts can change in two dimensions (rather than simply the depth position in a hierarchy). The two-dimensional position of each concept is therefore stored in the LAF file. This allows the layout of the domain to be preserved if a user wishes to migrate the domain from one MOT4 installation to another. The position is stored in a *metadata* tag within the concept, which is not currently used by ADE. This metadata tag could be extended to include extra display information such as the size/colour of the concept node in future version of MOT.

10.4.1.2 Concept Relationships

In contrast to the hierarchical nature of the CAF XML format, LAF stores all concepts directly within the *<domain>* element. This is very similar to the CAM XML format described in Chapter 8. Relationships are defined using a relation tag such as:

<relation from="0" to="1" type="is-parent-of" weight="100"/>, which in this case would define that concept 0 is a parent of concept 1. Storing relationships in this way allows non-parent relationships to be defined.

The *relation* tag is therefore able to store the edges of the Domain graph using a similar XML structure to CAM XML:

```
<relationship>
<sourceTerm>6E536B9A-F321-242E-BFD2-
002B67B27F54</sourceTerm>
<targetTerm>C403B750-83E1-E327-6E97-
0028C5CF29CC</targetTerm>
<relationshipType
source="http://www.grapple.org/relations.xml">
parent
</relationshipType>
<metadata/>
</relationship>
```

Moreover, the LAF style of defining relations also directly replaces the functionality of the CAF-relatedness relation by storing a weight as an attribute of the relation tag. This improves on the functionality that was existent in GAT, by allowing the weight to be used as metadata within the adaptation strategy.

10.4.2 Course Storage

The functionality of the goal model is similar to that of previous versions of MOT. For this reason, the basic layout of the LAF XML for a course is very similar to that of a goal model in CAF.

10.4.2.1 Metadata

As with the domain tag, metadata is stored in the course tag containing the goal map's name, the author's name and the date/time of last modification. Code listing 10.2 shows an extract from a goal map written in CAF format.

```
<lesson weight="0" label="">
<link weight="0" label="xml:5;basic:0">XML\XML
Topics\title</link>
<link weight="0" label="xml:6;basic:1"> XML\XML
Topics\text</link>
</lesson>
```

Code listing 10.2: Extract from a CAF Goal Model

Code listing 10.3 shows the equivalent lessons written in LAF format.

```
<lesson>

<link type=”title” concept=”0” domain=”71a4dbfc-9ffe-
4c4b-9851-69d1b699d0a3”>

<label weight=”5”>xml</label>

<label weight=”0”>basic</label>

</link>

     <link type=”text” concept=”0” domain=” 71a4dbfc-
9ffe-4c4b-9851-69d1b699d0a3”>

          <label weight=”6”>xml</label>

          <label weight=”1”>basic</label>

     </link>

</lesson>
```

Code listing 10.3: Extract from a LAF Goal Model

10.4.2.2 Referring to Attributes using Sublessons

As Code listing 10.3 shows, rather than using XPath to address the concept from the domain model, the domain's GUID is used together with the concept id. Finally, the *type* is used to choose the attribute of the sublesson.

10.4.2.3 Multiple Labels

The main change to the LAF format's storage of goal models is the way the native support for multiple labels. This allows the weight of each label to be treated separately, and therefore interpreted by the LAG strategy.

10.4.3 The LAG4 Grammar

In parallel to the development of MOT4 (and hence LAF), a new version of LAG (LAG4) was created by Joshua Scotton at the University of Warwick. The update to the LAG grammar was aimed to provide more flexibility to the author by providing new constructs, such as '*for-each*' which loops through a subset of concepts.

Similarly, the notion of filters is introduced to LAG4, which allows groups of concepts to be listed based on the value of a property. For instance the phrase *for-each(GM.Concepts.type == "introduction")* can also be expressed (using a filter) as *for-each(GM.Concepts[type == "introduction"])*.

10.4.3.1 Layout

Another major change introduced in LAG4 allows the strategy to choose which area of the screen to modify. ADE contains 5 separate screen areas – Centre, *North*, *East*, *South*, and *West.* By default ADE displays the navigation menu in the west pane, a todo list in the east pane and the content in the centre. However, using LAG4 code, the function of each of these areas can be adapted. For example, the code shown in Code Listing 10.4 shows how the todo list in the east pane can be replaced with a "Course finished" message when the user has visited every concept.

```
if(UM.GM.todoCount < 1) then(
Layout[E].type = "text"
Layout[E].content = "Course finished"
)
```

Code Listing 10.4: Sample LAG code to change the layout of a course

10.4.3.2 Multiple Labels

Whereas previous versions of the MOT toolset had required usage of the LIKE syntax to identify usage of multiple labels, the new format of LAF allowed ADE to separately address each label and weight. Therefore, the expression `GM.Concept.Labels['beginner']` will evaluate to true if (and only if) the

current sublesson contains a label called '*beginner*'. Similarly, the expression `GM.Concept.Labels['level'].value` will return the weight that is associated with the *level* label for the current sublesson.

10.4.3.3 Domain Properties

Previously, attributes within the MOT toolset have been used only for display purposes. However, the new LAG grammar supports the ability to address the value of one of their attributes. For instance, if a domain was created that contained a concept for each town in Warwickshire; the following code would show all the towns that had a population of greater than 80,000.

```
PM.GM.Concepts[DM.population > 80000].show = true
```

Such a property is directly analogous to the type of properties that are defined in the Domain tool of GAT. However, as Chapter 8 described, such properties within GAT are often used for the storage of pedagogical information. By contrast, the MOT toolset recommends that only factual information is stored as a domain property, and all pedagogical information is added using labels within the course tool (at the goal model level). This therefore maintains the MOT toolset's respect of the separation of concerns principle, and therefore allowing domains to be reused within a variety of different pedagogical strategies.

10.5 Implementing My Online Teacher 4

10.5.1 Terms Used

Following the same reasoning as the redefinition of GAT terms described in Chapter 8. An attempt was made to simplify the language used within MOT4 by changing

the terms *Domain Map* and *Goal Map* to *Domain* and *Course* respectively. Note, however, that the term 'GM' is still used in the LAG syntax.

10.5.2 Domain Design

Feedback from the evaluations described in the previous chapter had shown that users preferred the visualization of relationships using GAT. Similarly, the analysis of the submitted coursework described in Chapter 5 and Chapter 7 showed that few authors were using non-parent relationships.

The graph-based display of the MOT4 domain is implemented using JavaScript InfoVis Toolkit[55]. In MOT4, users can optionally drag-and-drop concepts to change their position (like in GAT), but unlike GAT they can also utilise the toolkit's automatic placement feature which is based on a force-directed placement algorithm[139]. For ease of display (and readability) only the properties of the relationships that involve the selected concept are displayed. For instance, whilst Figure 10.1 shows that there are relationships between 'The game of Gipf' and its 6 neighbouring concepts, only the relationships directly involving the selected concept ('The rules') are highlighted, with their relationship types and weights.

[55] http://thejit.org/

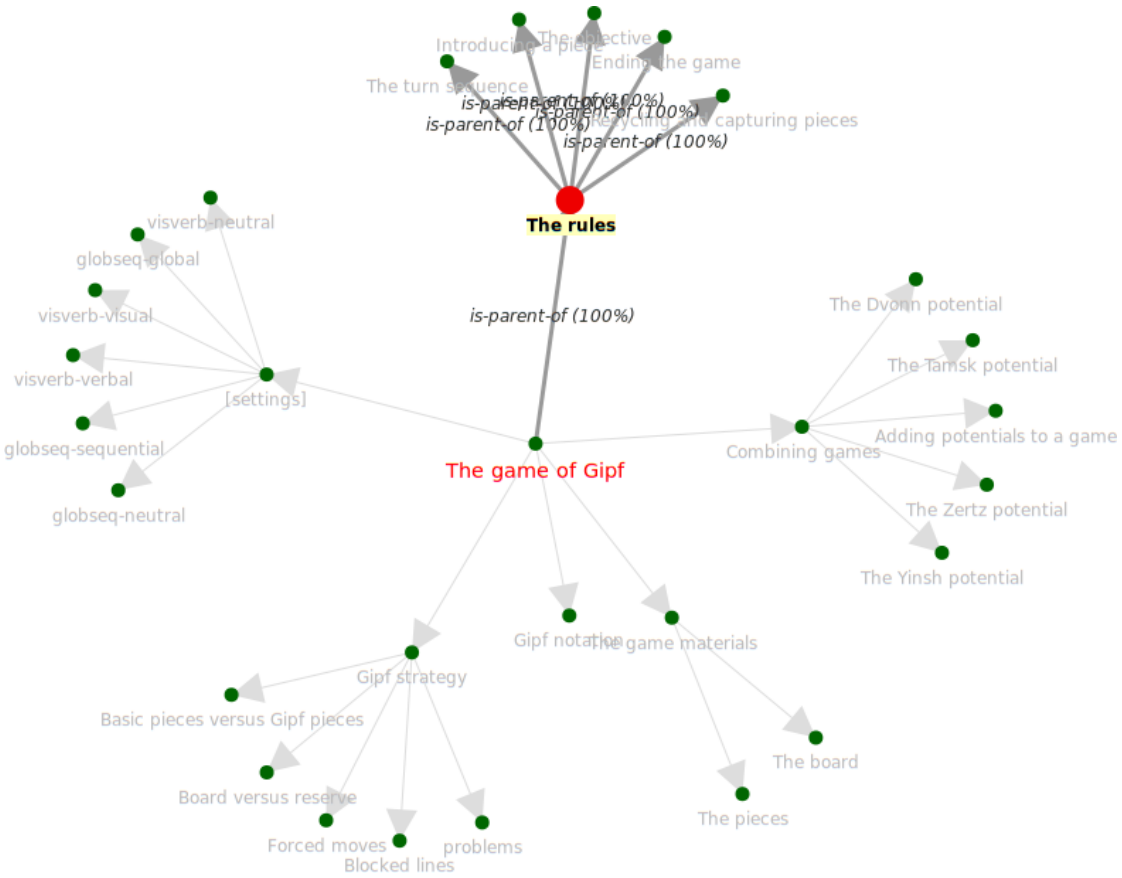

Figure 10.1: A domain in MOT4 with the concept 'The rules' selected

10.5.2.1 Domain Relationships

The evaluation had also suggested that the users preferred MOT3.1's interface for managing relationships (such as that shown in Figure 10.2). For this reason, MOT4 retained the style of the MOT3.1 'New Relation' feature (including the ability to calculate the strength of relatedness relations, as described in Chapter 7). Rather than using a label to define the type of relationship, MOT4's new relationship dialog adds an editable autocomplete text box to allow the user to choose the relationship type. Moreover, the '*Calculate Relatedness*' feature that was implemented for MOT3.1 was retained, and can be used to assist the author in adding relationships between concepts.

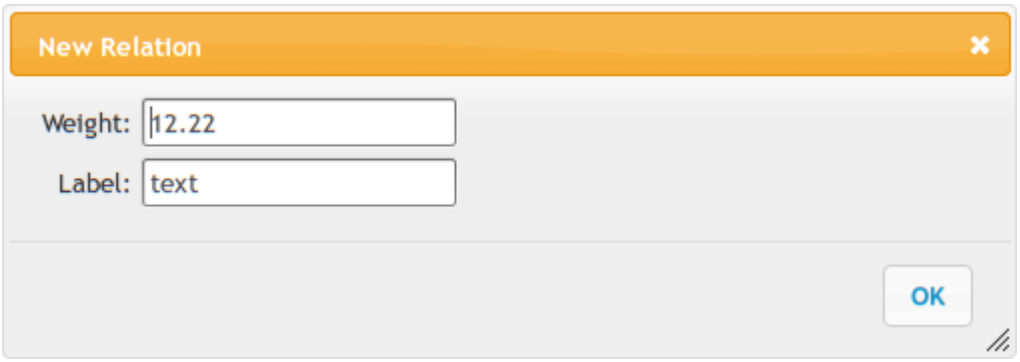

Figure 10.2: Adding a relationship in MOT3.1

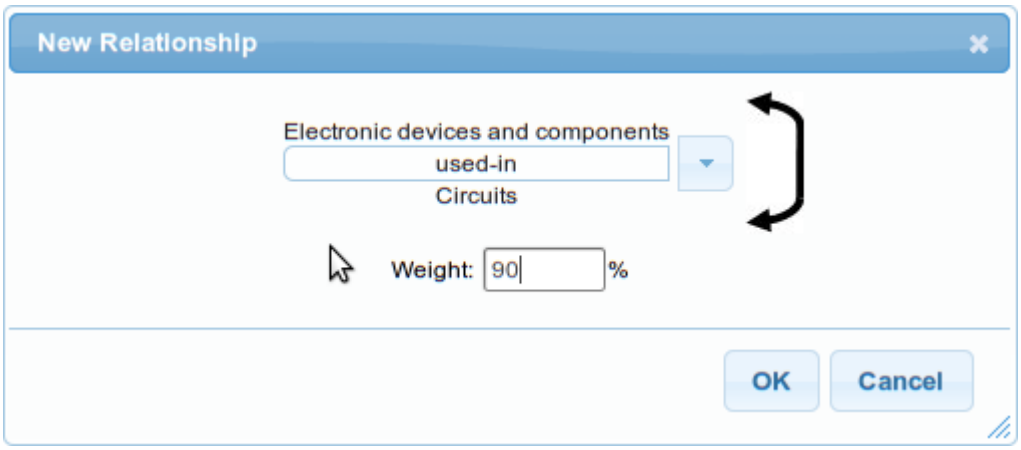

Figure 10.3: Adding a relationship in MOT4

Having all previously used relationship types available within a drop-down autocomplete menu encourages authors to reuse existing relationship types, thus creating a type of folksonomy of relationship types [140].

As with GAT (and other versions of MOT), only the parent relationship has any intrinsic meaning in the MOT toolset. It was also decided to clarify the meaning of the parent relationship, by renaming it to *is-parent-of*. The *is-parent-of* relationship is directly analogous to the hierarchical parent relationship in previous versions of MOT, and is used when generating the goal model (and therefore the hierarchical menu in ADE). Please note, however, that unlike *parent* relationships in GAT the *is-*

parent-of relationship in MOT merely provides a template for the hierarchy of the final navigation menu. The layout of the navigation menu can be fine-tuned by the user using the *Course* (Goal Model) tool.

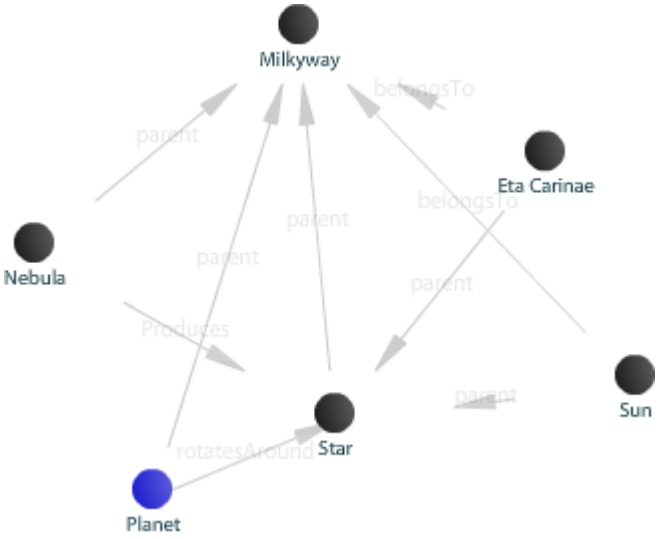

Figure 10.4: Extract of the Milkyway Domain within GAT

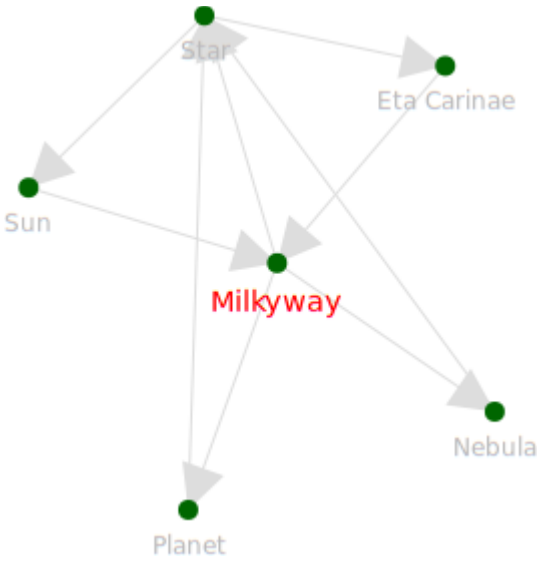

Figure 10.5: Extract of the Milkyway Domain within MOT4

Since the parent relationship is important in the process of designing the course hierarchy, another option allows the user to rearrange the domain in a 'tree layout' mode (see Figure 10.6).

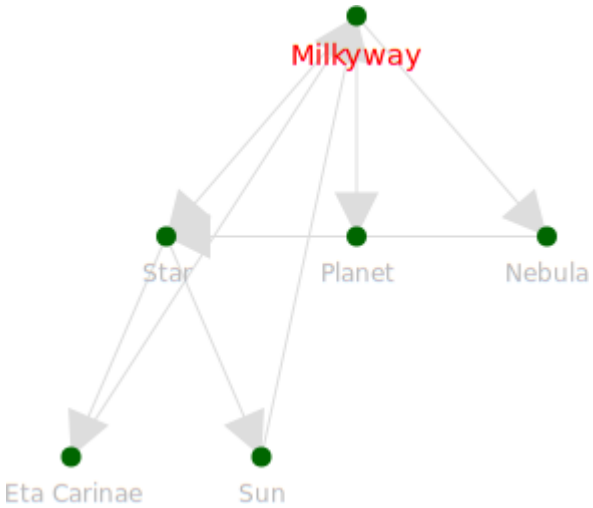

Figure 10.6: Tree layout mode in MOT4

10.5.2.2 Concept Operations

When a user creates a concept, it is immediately added as a child concept of the currently selected concept. A new feature named '*Add Many Concepts*' allows the author to create several concepts at once by inputting a list of concepts (separated by *newline* characters).

Based on the user interface techniques employed in GAT, MOT4 allows multiple concepts to be selected at a time using a rubber-band technique[135].

10.5.2.3 Creating New Attributes

As with the definition of relationship types (described above), users can specify new attribute types or select existing ones using an autocomplete drop-down list (similar to that used with relationship types as described above). As with relationship types, the custom attribute list is designed to encourage users to reuse existing attribute types, thus producing a folksonomy of attribute types.

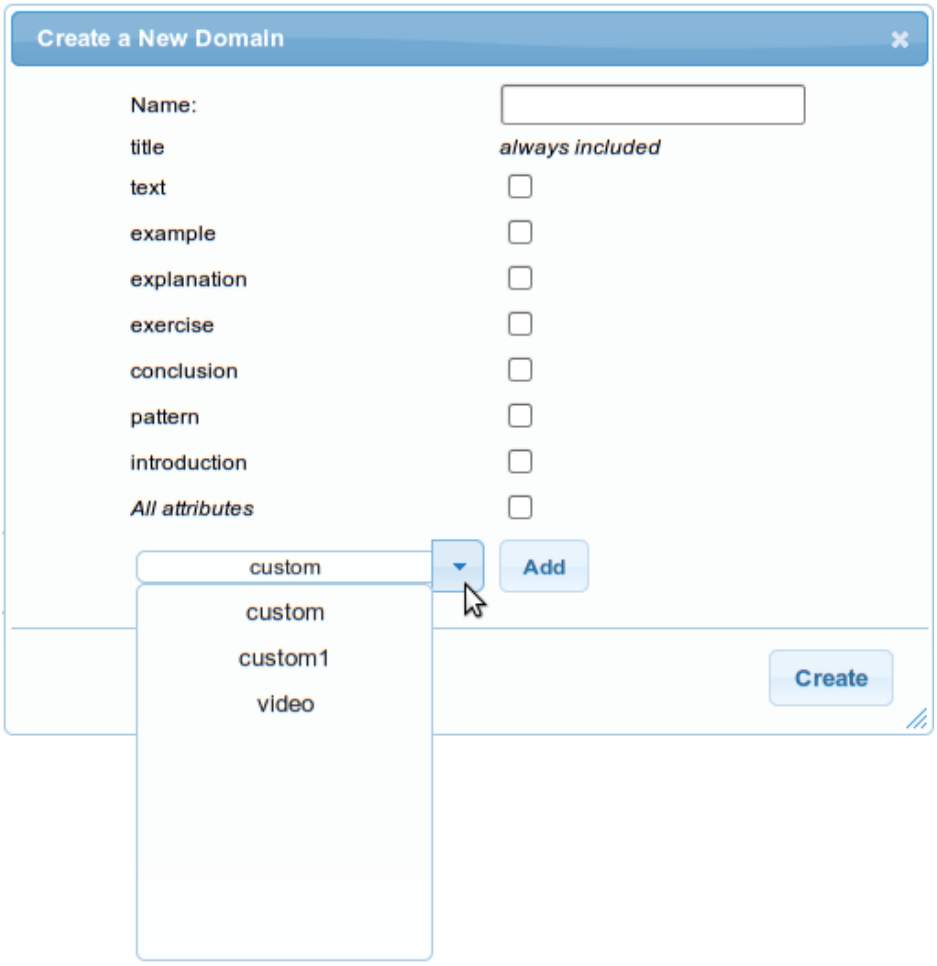

Figure 10.7: Creating a domain with custom attributes in MOT4

10.5.3 Course Design

Instead of using the JavaScript-based tree component used in MOT3.1 (see Chapter 7), the hierarchical display of a goal model was redesigned to use HTML, CSS and jQuery, based around the nestedSortable jQuery plugin[56]. This was chosen because it uses the same components as jQuery UI[57], and therefore provides a coherent look-and-feel to the whole MOT4 system. The tree is based around a series of

[56] http://mjsarfatti.com/sandbox/nestedSortable/

[57] http://jqueryui.com/

nested HTML '*ordered lists*' (`<ol>`), with each sublesson being represented by a separate list item (`<li>`). This allows more customization of the individual items than was previously possible with the dedicated JavaScript component used in MOT3.1.

Figures 10.8 and 10.9 compare the more text-based way of displaying labels within MOT3.1 to the slightly more graphical way of separating labels into separate boxes. The MOT4 method aims to reduce the cluter and therefore improve the readability of the labels. Similarly, the truncated preview of the sublesson has been removed from the goal model tree. Rather than using three panes for the layout (as in MOT3.1), the MOT4 interface only uses two panes by moving the sublesson preview feature into a tab (see Figure 10.10)

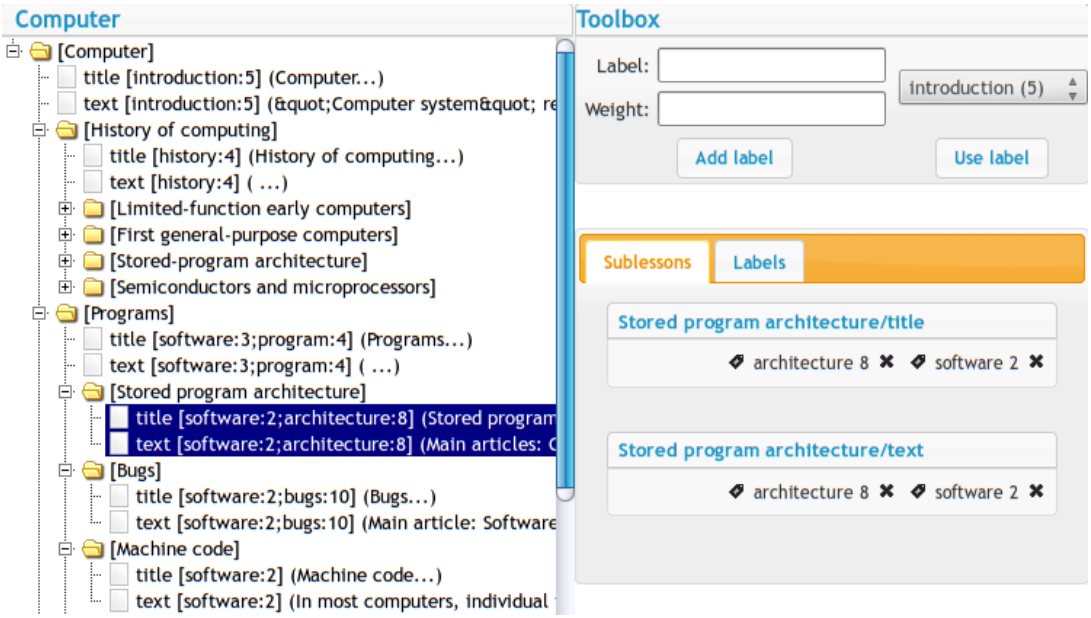

Figure 10.8: A goal map in MOT3.1

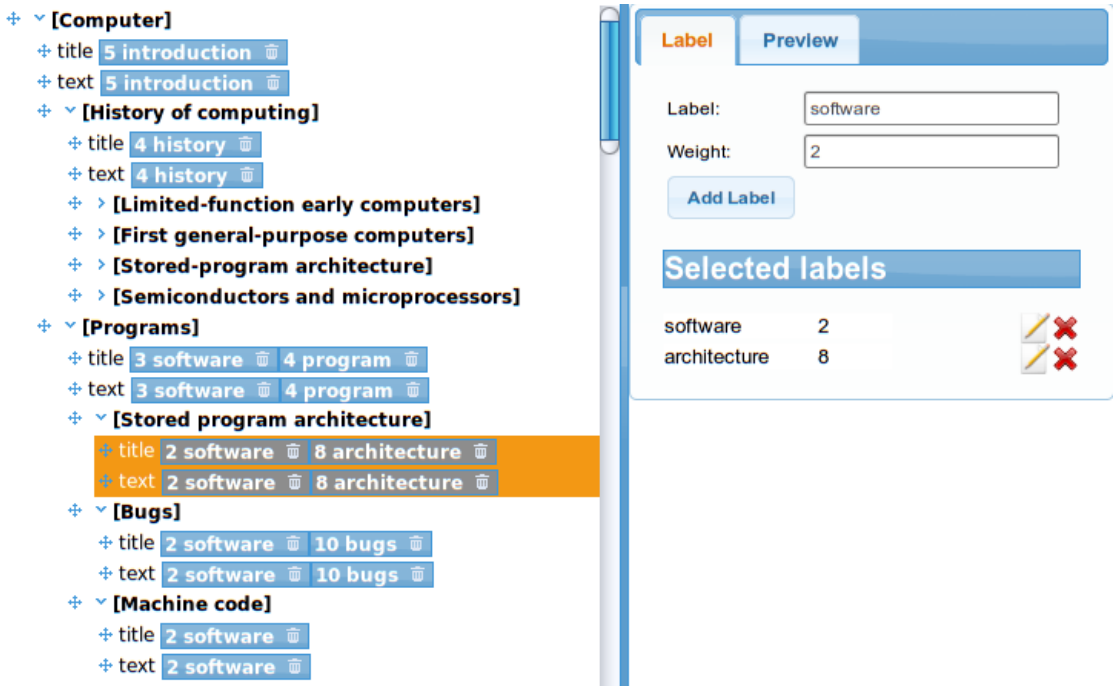

Figure 10.9: A Course (Goal Model) view in MOT4

10.6 Evaluations

Based on the improvements made to MOT4, the following hypotheses were created.

H10.1. Users create more (non-parent) domain relationships using MOT4 than students previously have done using other versions of MOT.

H10.2. Users add more labels to each sublesson than in previous versions of MOT.

H10.3. Users appreciate the automatic layout feature for arranging domain graphs.

H10.4. Users create less complex LAG files when being forced to use a text-editor.

This section describes the evaluations that took place to test these hypotheses.

10.6.1 Evaluation of MOT4.0

The fourth year MSc module 'Dynamic Web-based systems' at the University of Warwick used the updated MOT toolset to carry out a similar coursework to that described in Chapters 5 and 7.

Six groups were formed of 2-3 students. As with the evaluations described in Chapters 5 and 7, groups were required to choose two subjects (relevant to the module) and create a domain about each of their chosen subjects. From each of these domains, they were required to create two adaptation strategies, to produce a total of four different adaptive courses. Whereas in previous years the coursework deadline had been approximately 6 weeks after the student's initial exposure to the system, this particular cohort was allowed to continue working until after the Easter break. However, due to the collaborative nature of the coursework, the extra time may not have been overly beneficial to the groups. Indeed, whereas in previous years the students were continually providing feedback and requesting features during the module's lectures and seminars, many groups did not begin the coursework until the end of the module. This meant that there was less interaction between the students and the researchers.

An introductory seminar demonstrated the usage of both MOT4 and ADE to the students, after which the students were asked to start using the system and ask a lab assistant about any usage problems. At the start of term, LAF import support for ADE was still experimental so the students were advised to use CAF during their initial experiments. The students were later notified (in a subsequent coursework

workshop session) when the LAF support had been implemented to allow enhanced support for multiple labels, as described in Section 10.4.2.3 .

The radical change in the syntax of LAG had caused the existing PEAL and PEAL2 editors were out of date. The students therefore used standard text editors to author the strategies.

10.6.1.1 Analysis of Submitted Content

This section presents a quantitative analysis of the CAF/LAG files that were submitted as coursework by the students.

10.6.1.1.1 Domain Map Content

As with the previous evaluations described in Chapters 5 and 7, it is necessary to analyse the content from three perspectives: *All Domain Maps, Best Domain Map per Group* and *Average Domain Content per Group*.

Table 10.1 shows that the number of concepts created by the students in the 2011 cohort has actually decreased compared with the usage of MOT3.1 in 2010 although it is still slightly higher than that of 2008 with MOT1.0. Similarly, there is a reduction in the number of attributes used.

None of these differences are statistically significant when considering either the '*Best DM per group'* or the '*Average DM per group'*. However, when considering '*All DMs',* the reduction in the number of attributes is statistically significant (with a two-tailed significance of $p = 0.061$), and the reduction in the number of attributes per concept is also statistically significant (two-tailed significance of $p = 0.006$). Moreover, when considering the 'All DMs' category, the number of attributes that

contained more than two different HTML tags was significantly higher when using MOT4 than when using MOT3.1 ($t(43) = -2.526$, $p < 0.05$, $r = 0.36$).

Interestingly, none of the groups that used MOT4 used any non-parent relationships at all. This could be partly to do with the fact that ADE had not completely implemented the syntax that is required to address the new LAF style relationships. Moreover, none of the students submitted any LAF files, preferring to remain with the legacy CAF format.

	N	*Concepts*	*Depth of Average Concept*	*Number of Attributes*	*Attributes per Concept*	*Number of Custom Attribute Types*	*Percentage Attributes (excluding title) containing more than 2 unique HTML tags*
MOT1.0: All DMs	4	22 SD: 12.44	2.75 SD: 0.31	63 SD: 37.28	2.8 SD: 0.56	1.25 SD: 1.89	2.43 SD: 4.86
MOT3.1: All DMs	21	26.38 SD: 11.71	2.77 SD: 0.25	75.43 SD: 40.25	2.81 SD: 0.59	1.71 SD: 1.9	15.98 SD: 17.13
MOT4: All DMs	24	23.67 SD: 10.54	2.95 SD: 0.57	56.33 SD: 25.44	2.36 SD: 0.45	1.67 SD: 2.55	35.49 SD: 31.53
MOT3.1: Best DM Per Group	8	32.75 SD: 11.51	2.86 SD: 0.19	90.88 SD: 45.68	2.66 SD: 0.58	1.5 SD: 2.07	19.1 SD: 20.73
MOT4: Best DM Per Group	6	29.33 SD: 11.96	3.14 SD: 0.49	68.5 SD: 24.4	2.36 SD: 0.37	1.67 SD: 1.37	41.12 SD: 35.87
MOT3.1: Average DM per Group	8	27.19 SD: 7.37	2.77 SD: 0.13	77.9 SD: 30.89	2.81 SD: 0.55	1.65 SD: 1.8	17.7 SD: 18.08
MOT4: Average DM per Group	6	23.67 SD: 4.73	2.86 SD: 0.11	56.33 SD: 14.11	2.36 SD: 0.35	1.67 SD: 2.06	35.75 SD: 34.39

Table 10.1: Domain Production for 2010 compared with 2009 and 2008

10.6.1.1.2 Goal Map Content

As with the analysis of previous evaluation results, the goal map analysis considers the following criteria: *All GMs*, *Best GM per Group* and *Average GM per Group*.

As described above, no groups used the LAF format. This meant that no groups used the more advanced features concerning multiple labelling. Moreover, only one group used the old MOT3.1-style multiple labels, and even this group only used a single pair of labels (represented as "*tod:1;inter:1*") to apply to two of their sublessons. Indeed, the LAG strategy that was submitted to accompany this goal map did not use the correct syntax to address such multiple labels, so the authors must have used this label by mistake.

Similarly, many of the labels that were utilized by the groups appeared to be based on very basic (non-pedagogical) terms. For instance, one group used 10 separate labels named '*labelA*', '*labelB*', '*labelC*', etc.

Another group used the label '*important*' which was applied to every 'title' sublesson, and always had a weight of 5. Apart from the fact that such sublessons could easily be identified by using the `DM.Concept.type` expression, using such monotonous labels makes the labels irrelevant.

	Number of Cases	*Average Number of Distinct Labels*	*Average Number of Sublessons*	*Percentage of Labelled Sublessons*	*Percentage of Weighted Sublessons (i.e., labels with weight other than 0)*
2010 (MOT3.1): All GMs	24	5.88 SD: 10.78	83.17 (SD: 48.56)	68.26 (SD: 37.27)	32.26 (SD: 41.17)
2011 (MOT4): All GMs	24	3.96 SD: 3.544	57.25 (SD: 24.47)	69.71 (SD: 39.83)	55.74 (SD: 42.13)
2010 (MOT3.1): Best GM per group	8	11.25 SD: 17.96	106 (SD: 63.05)	84.44 (SD: 31.44)	67.27 (SD: 40.75)
2011 (MOT4): Best GM per group	6	4.67 SD: 4.59	74 SD: 22.17	65.3 (SD: 40.76)	47.77 (SD: 45.11)
2010 (MOT3.0): Average GM per group	8	5.3 SD: 4.87	86.96 SD: 42.14	69.55 (SD: 28.25)	39.53 (SD: 25.41)
2011 (MOT4): Average GM per group	6	3.96 SD: 3.3	57.25 SD: 11.66	70.87 (SD: 30.41)	56.64 (SD: 27.46)

Table 10.2: Comparison the Goal Maps created using MOT4 (in 2011) to those created in MOT3.1 (in 2010)

10.6.1.1.3 LAG Strategies

This section provides a quantitative analysis of the LAG strategies that were submitted as part of the coursework, according to the various measures that were defined in Chapter 5.

	N	***Lines of Code***	***Lines of Description***	***Lines of Comments***	***Characters of Code***	***Characters of Description***	***Characters of Comments***
PEAL2 (2010) All Strategies	31	48.94 SD: 37.18	10.94 SD: 9.1	10.19 SD: 9.59	964.32 SD: 840.3	711.1 SD: 573.61	574.52 SD: 448.44
Text Editor (2011) All Strategies	25	33.76 SD: 28.34	4.4 SD: 2.96	2.12 SD: 2.88	678.4 SD: 599.83	324 SD: 195.65	111.88 SD: 167.8
PEAL2 (2010) Best Strategy per Group	9	76.56 SD: 51.79	11.67 SD: 11.68	16 SD: 14.49	1560.33 SD: 1211.04	650.33 SD: 682.43	818.78 SD: 206.66
Text Editor (2011) Best Strategy per Group	6	51.17 SD: 38.88	5.67 SD: 3.27	2,67 SD: 3.88	1036.5 SD: 809.35	374.17 SD: 184.96	149.83 SD: 244.49
PEAL2 (2010) Average Strategy per Group	9	47.75 SD: 22.22	10.06 SD: 7.57	9.03 SD: 6.65	932.94 SD: 503.48	639.64 SD: 475.7	510.61 SD: 349.68
Text Editor (2011) Average Strategy per Group	6	34.96 SD: 23.99	337.5 SD: 140.37	2.04 SD: 2.89	697.7 SD: 515.2	337.5 SD: 140.37	105.17 SD: 162.09

Table 10.3: A quantitative analysis of the strategies submitted in 2010 and 2011

Table 10.3 shows that many of the quantitative measures of the complexity of the strategies actually decreased between 2010 and 2011.

Table 10.4 shows the effect size of all the measures that were found to be statistically significant (at 95% confidence interval) when performing a one-tailed t-test.

Measure	***Category***	***t-value***	***Effect size (r)***
Lines of Description	All strategies	t(37.571) = 3.76	0.52 (large)
Lines of Comments	All strategies	t(36.506) = 4.447	0.59 (large)
	Best strategy per group	t(9.636) = 2.623	0.65 (large)
	Average strategy per group	t(11.68) = 2.782	0.63 (large)
Lines of Code	All strategies	t(54) = 1.683	0.22 (small)
Characters of Description	All strategies	t(38.284) = 3.513	0.49 (medium)
Characters of Comments	All strategies	t(39.821) = 5.302	0.64 (large)
	Best strategy per group	t(11.193) = 2.915	0.66 (large)
	Average strategy per group	t(13) = 2.633	0.59 (large)

Table 10.4: Effect size of the statistically significant differences within the quantitative analysis of the created strategies

By all measures, the strategies submitted in 2011 had significantly fewer lines (and characters) of comments, with a large effect size. Similarly, there is some evidence to say that there were significantly fewer lines (and characters) of description in 2011, but only when using the '*All Strategies*' category. There was also a small (but statistically significant) drop in the number of lines of code submitted in 2011 when considering all strategies submitted by the students.

A number of factors may have affected the quantity of code within the strategies submitted in 2011. Firstly, the new LAG4 syntax has simplified many operations (as described in Section 10.4.3) this may have led to the strategies being able to describe strategies using fewer lines/characters of code. Secondly, the change in format of the coursework (as described in Section 10.6.1) may have affected the potential for collaboration between the students, and therefore discouraged the

groups from creating innovative strategies. This seems plausible based on the fact that there was also a reduction in the quality of the submitted goal maps, however further research would be required to ascertain the pedagogical quality of each submitted strategy. Finally, the students did not have the support of any dedicated LAG editors (such as PEAL or PEAL2). Similarly, the small decrease in the amount of code could be used to support the need for IDE-style editors.

10.6.1.2 Evaluating MOT4 against GAT

As with the previous year's module, towards the end of the term a seminar introduced the students to GAT (as described in Section 9.6). The students were given a short introduction to the tool, and were encouraged to use the tool within a two-hour seminar, as with the previous year's evaluation, the students were shown the PRT tool, but were not required to create their own PRTs.

After each participant had finished creating a simple adaptive course using GAT, a teaching assistant provided the participant with a link to an online survey.

The survey design software[58] allowed for a greater customization of the questions than the software that was used for the previous MOT versus GAT evaluation (described in Chapter 9), allowing for more detailed questions to be asked. In particular, there were no neutral options provided, ensuring that the students had to give an opinion.

Unlike the previous year's evaluation (which had taken place approximately two weeks before the coursework deadline), the longer period of time allowed for the

[58] The student version of Survey Gizmo was used: http://students.sgizmo.com/

coursework (described in Section 10.6) meant that the GAT seminar took place approximately seven weeks before the coursework deadline. For this reason, many students still had very fundamental misconceptions about the process of adaptive hypermedia. Indeed some students said that they hadn't used MOT at all yet, and therefore found it impossible to compare between the two toolsets, such students were did not complete the questionnaire. Those who did respond were asked "*Other than the initial demonstration seminar, please estimate the number of hours you (personally) have spent working with MOT*" – the average response was 5 hours (SD: 3.55).

There were only 8 respondents to the questionnaire – the full questionnaire can be found in Appendix B. Many of the results were inconclusive, showing either a 50% split between each system, or a 5:3 split between the options.

There were only three questions that showed any clear majority.

- 6 out of the 8 students said that GAT was better for cloning concepts.
- 6 out of the 8 students said they preferred authoring adaptation by using "*smaller 'building blocks' (as in GAT)*" over using "*entire strategies at a time (as in MOT)".*
- Similarly, 6 out of the 8 students said they preferred creating a "*GAT course based on existing PRTs*" over writing a "*whole LAG file*".

Since this questionnaire had no neutral (undecided) position, the results could be analysed using a binomial test, but none of these results are statistically significant. However, out of the three questions that provided such a 6 out of 8 (75%) result,

two of them concerned the usage of fine-grained adaptation strategy. This therefore appears to rule out the possibility that the students were randomly selecting their answers. However, the small sample size suggests that the result should not be considered particularly significant.

As well as having basic "*which do you prefer…*" style questions, the survey asked the students how confident they felt with using the various tools. Each question had a *Likert* [8] scale with the options: *Very Uncertain*, *Uncertain*, *Neutral*, *Confident* and *Very Confident*. For the purposes of analysis, these categories can be represented as numbers (with -2 representing '*Very Uncertain*' and 2 representing '*Very Confident*'). Scoring the answers in this way assumes a monotony of the *Likert* scale, which seems to be a reasonable assumption because of the natural language of the terms (and the fact that it was presented in the questionnaire as a linear scale). The average score given for each of the tools is presented in Table 10.5.

Tool	***Average Score***
MOT Domain	0.5
MOT Goal	0.75
LAG Code	0.375
GAT Domain	0.625
GAT Course	0.375
GAT PRT (Creating from Scratch)	0

Table 10.5: The students 'confidence' score for the tools

Interestingly, none of the average scores for the tools are negative, suggesting that the students were not especially uncertain about the usage of the tools. However, the difference between each tool and the small sample size prevents a conclusive statement from being made about the varying complexities of the tools.

At the end of each question, the students were invited to make a comment. Some of the most interesting comments included:

"*Linking to URIs is nice as it allows you to use additional content, but to create attributes in MOT is easier to use and it's nice to have all the content in one place.*"

"*The possibility of using external URLs is definitely an advantage.*"

Chapter 9's evaluation showed a lack of a clear preference between using an integrated editor and using external resources. The above comments also suggest that users would appreciate the option to use *either* an integrated authoring tool or external responses. This feature could potentially be implemented in MOT by providing a template for an `iframe` HTML tag.

Concerning the authoring of adaptation (using reuse), one student said:

"*The GAT approach is more like I would expect such a tool to be like, but the UI was too complex.*"

Other comments included:

"GAT is possibly more intuitive, and it looks better!"

"*GAT has more operation steps*"

"*MOT has less operations*"

"*GAT looks difficult but it is delivered easier than MOT*"

10.7 MOT4.01

After the UK evaluations, various minor adjustments (including some minor bug fixes) were made to MOT4.

10.7.1 Look and Feel

The look-and-feel of MOT4 had primarily utilised the *jQuery UI*[59] library. However, the theme that had been previously chosen was rather grey and boring. One of the comments described in the previous section had said that GAT looks better. Chapter 8 described how the GAT '*Welcome Screen*' had been designed to be child-like by using bright colours. A similar principle was applied to MOT4.01, by ensuring that a consistent theme was applied to all controls. Similarly, more intuitive controls were added to the course tool; Figure 10.10 shows how the original MOT4 used a resizable pane to differentiate between the labelling tool and the sublesson preview. Figure 10.11, uses tabs and clearer headings to differentiate between these two features.

[59] http://jqueryui.com/

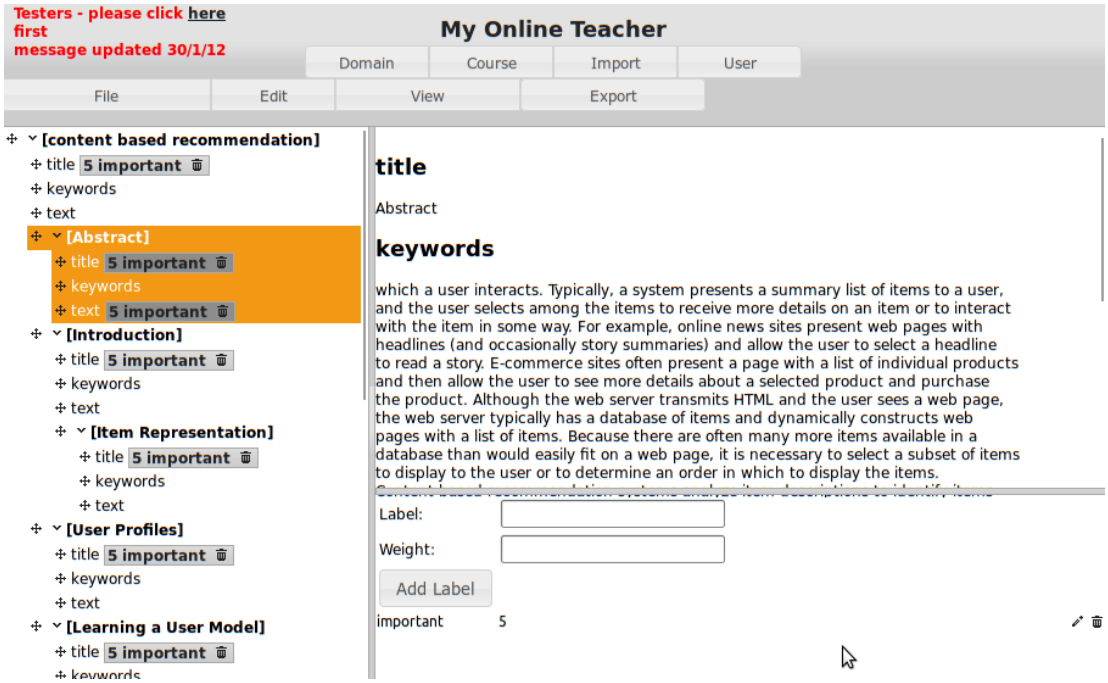

Figure 10.10: Screenshot of the course tool within MOT4

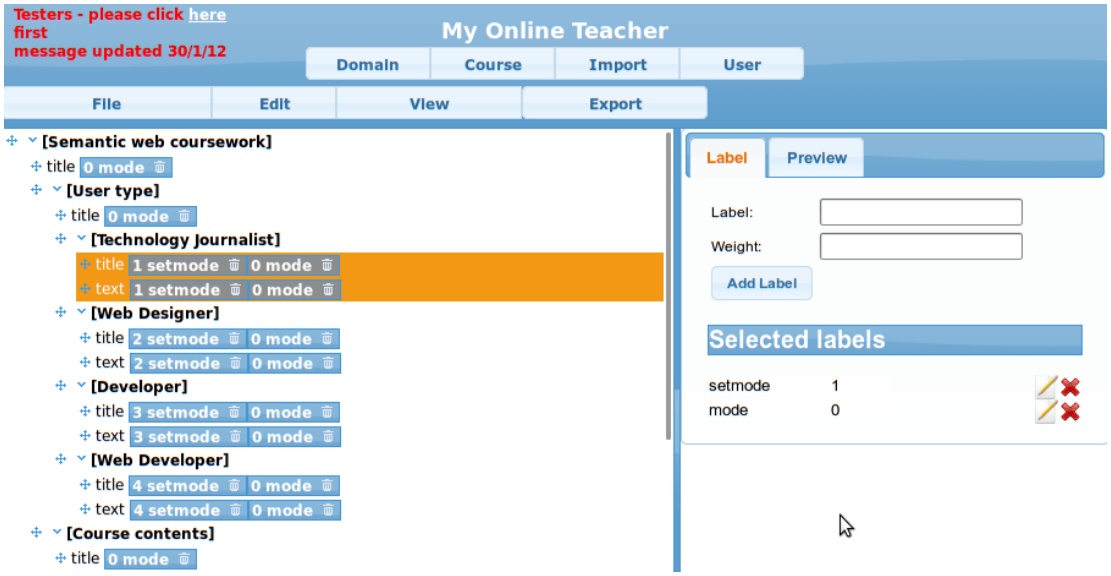

Figure 10.11: Screenshot of the course tool within MOT4.01

10.7.2 Tooltips

In an attempt to provide more assistance to the author, a context-sensitive tooltip feature was added. Tooltips were added to a large variety of different controls. For instance, Figure 10.12 shows that the labelling feature has two types of tooltips; a yellow assistant tooltip (activated when users place their mouse over the control)

and a red warning tooltip (in this case activated when an invalid character is found in the weight textbox).

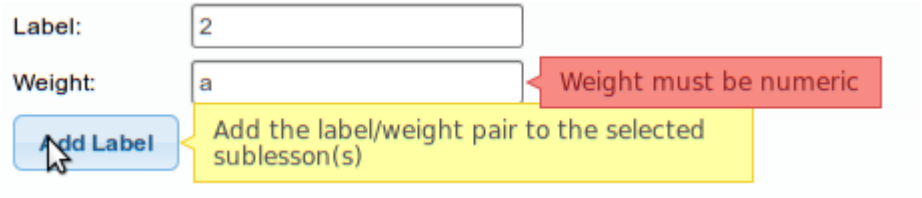

Figure 10.12: Tooltips when adding labels

10.7.3 Evaluation of MOT4.01

The previous evaluation had not demonstrated the usage of non-parent relationships. This was largely due to the fact that the LAG syntax had only recently allowed such relationships to be addressed.

A further evaluation was performed at the Politehnica University of Bucharest, Romania with a class of students who were studying a module named '*Semantic Web*'. As part of the module, they are expected to learn information about XML, and are therefore expected to be familiar with topics such as structuring information and applying metadata. For this reason, a piece of coursework was devised to assess their knowledge of XML related topics.

The students are not taught about adaptive hypermedia, so the evaluation focussed on the students' usage of MOT4.01 rather than the creation of adaptation strategies.

10.7.4 Hypotheses

The evaluation of MOT4.01 needed to clarify the findings of the evaluation described in the previous section. For this reason, two of the initial hypotheses are kept.

H10.1. Users create more (non-parent) domain relationships using MOT4 than student's previously have done using other versions of MOT.

H10.3. Users appreciate the automatic layout feature for arranging domain graphs.

The teaching of the LAG language was beyond the scope of the *Semantic Web* module so H10.2 and H10.4 were removed from this evaluation. It was decided not to test H10.2 because the labels are closely related to the strategy.

Two further hypotheses can also now be investigated.

H10.5. The tooltips are considered helpful.

H10.6. MOT4.01 is consistent with other applications.

H10.6 was added to support the familiarity imperative defined in Chapter 9.

The students were divided into 8 groups of 3-4 students and were given 4 weeks to build domain maps and goal maps about various XML topics (including XML, XSLT and XPath). The seminar that introduced the coursework demonstrated the basics of ADE (so that they could preview their content) and described the basics of the LAG language.

10.7.5 Example Strategies

The students were not expected to design LAG strategies, but were taught how to reuse some example strategies. LAG files expressing the following strategies were given to the students to reuse.

- A 'set*mode*' strategy is based around a hypothetical set of users (*Technology Journalist*, *Web Designer*, *Developer,* and *Web Developer*) who each require different amounts of information about the XML related topics. When the learner views a sublesson that has a label named '*setmode*', the weight of that label becomes stored in the user model ('`UM.mode`'). When the user views a sublesson that has a label named '*mode*', the sublesson is hidden unless the weight of the label matches the mode in the user model.
- A 'ChooseRel' strategy was designed to encourage the users to use the non-parent relations feature, and thus create more complicated domain maps. The strategy allows learners to choose a relationship type by clicking on a sublesson that has a label named '*setrel*'. If the sublesson has a second label (for example '*composedOf*') the name of the second label becomes the active relationship type. A ToDo list on the right-hand side of the screen shows the relationship type, but also all concepts that are related to the currently viewed concept by the active relationship type.

10.7.6 Analysis of Submitted Content

In contrast to the previous evaluation of MOT4, 8 out of the 9 groups submitted a file in LAF XML format. Table 10.6 shows that 5 out of the 8 groups used some non-

parent relationships, and those that did created an average of 8.4 instances of such non-parent relationships. This does show that the students were able to create more complex concept relationships – thus supporting H10.1.

Group	*Number of Concepts*	*Number of Non-Parent Relationship Types*	*Number of Non-Parent Relationship Instances*
1	51	0	0
2	18	1	9
3	73	0	0
4	34	0	0
5	39	1	9
6	49	2	7
7	14	1	9
8	37	1	8
Average	39.38	0.63	4.13

Table 10.6: Number of Concepts and Relationships used with MOT4.01

Table 10.7 shows the Bucharest students (using MOT4.01) created a higher number of concepts than the Warwick students (using MOT4.0), with a higher number of attributes. The depth of the average concept was also higher, as was the number of custom attribute types. However, there were fewer attributes per concept and a smaller percentage of attributes. None of these differences are statistically significant.

	N	*Concepts*	*Depth of Average Concept*	*Number of Attributes*	*Attributes per Concept*	*Number of Custom Attribute Types*	*Percentage Attributes (excluding title) containing more than 2 unique HTML tags*
MOT4: Best DM Per Group	6	29.33 StdDev: 11.96	3.14 StdDev: 0.49	68.5 StdDev: 24.4	2.36 StdDev: 0.37	1.67 StdDev: 1.37	41.12 StdDev: 35.87
MOT4.01: DM	8	39.38 StdDev: 18.88	3.45 StdDev: 0.92	83.88 StdDev: 39.02	2.29 StdDev: 0.69	7.22 StdDev: 10.66	24.21 StdDev: 16.58

Table 10.7: Comparing Domain Content submitted by the UK students (using MOT4) to that submitted by the Romanian students (using MOT4.01)

The main reason for the difference is due to the fact that two very different coursework tasks assigned to the students. Whereas the UK students had to create courses that used original strategies, the Romanian students only needed to reuse existing strategies. Indeed, there was a large variety in the quality of the domain maps submitted by the Romanian students. For instance one group decided to concentrate on only the structure of the Domain – this group created 73 concepts but only provided 'title' attributes (with no actual content).

The types of strategies supplied to the students also clearly had an effect on the quality of the domains. The UK evaluation (described in Section 10.6.1) had not supplied any example LAG strategies that utilised the new relationships feature, which meant that no groups decided to use the feature. By contrast, one of the two example strategies supplied to the Romanian students did use the new relationship syntax, which encouraged the students to utilise the more advanced features of the MOT4 Domain tool.

Similarly, a large (but statistically insignificant) rise in the number of custom attribute types defined is due to the fact that some students decided to address the '*setmode*' strategy by defining typenames such as 'Web developer', rather than simply adding labels. In some ways, this suggests that the users were trying to add pedagogical information into the domain. This could be because the students were not taught about the basic principles of adaptive hypermedia (such as the separation of concerns principle), and were therefore not designing their domains to be reused.

Similarly, the quality of the goal maps submitted is entirely influenced by the strategies supplied to the students. It is therefore unnecessary to compare the goal map content to that submitted by the UK students. However, every group used the labelling feature: on average each group added at least one label to 70.23% of their sublessons (SD: 31.14). Moreover, on average each group added at least two labels to 26.08% of their sublessons (SD: 32.94).

Although no statistical significance can be found within these statistics, they show that the students spent a reasonable amount of time using the system, and it was valuable to gather feedback about their usage of the system.

10.7.7 MOT4.01 Questionnaire

As part of the coursework submission process, the Romanian students were asked to submit a questionnaire. There were 28 responses, and as with the UK evaluations each participant was asked how many hours they had personally spent using MOT, the average response was 9.93 hours (SD: 5.34). The full questionnaire can be found in Appendix C, whilst this section describes the most interesting results

The questionnaire provided a 4-point Likert scale, and asked the respondent to rate a series of statements. The options in the scale were *Strongly Disagree*, *Disagree*, *Agree* and *Strongly Agree* which for ease of analysis were coded as 1-4 respectively. As with previous evaluations described in this thesis, the analysis used assumes a monotony of the *Likert* scale. Each statement had 28 responses.

The first set of statements was designed to find the students' opinion about how consistent MOT4.01 is with other tools. Table 10.8 shows the statements and the most frequent response together with the mean and standard deviation.

Statement	***Most frequent answer (Mode)***	***Mean***	***Standard Deviation***
MOT is consistent with other applications I have used	3	2.5	0.64
The interface for editing (adding/deleting/copying concepts) a Domain structure was consistent with other software I have used	3	2.64	0.56
The interface for adding/deleting attributes was consistent with other software I have used	3	2.5	0.69
The interface for editing the content of attributes was consistent with other software I have used	3	2.64	0.62
The interface for rearranging a course (by dragging the tree) was consistent with other software I have used	3	2.5	0.64
The interface for adding labels to a course was consistent with other software I have used	3	2.57	0.69

Table 10.8: Likert-scale responses about consistency

Table 10.8 shows that overall the students appeared to have no strong opinion about how consistent the tool is with other software. At the end of the question, they were asked to name any pieces of software that they had previously used that were similar to MOT4.01. Four of the respondents mentioned blogging and content

management services such as Blogspot[60], Wordpress[61] and Joomla[62]. Two respondents mentioned Zooma[63], which is a text-based tool for mapping ontologies. Other students mentioned Enity-Relationship and UML diagram software such as astah[64]. Each of these tools has a separate purpose to MOT; however future research may be able to build on the usability techniques employed by these different tools.

Table 10.9 shows the statements and responses to the next question, which focussed on ease of use.

Statement	***Most frequent answer (Mode)***	***Mean***	***Standard Deviation***
It is easy to create concepts in MOT	3	3.14	0.65
It is easy to copy concepts in MOT	3	2.75	0.8
It is easy to delete concepts in MOT	3	3.1	0.57
It is easy to reuse concepts in MOT	3	2.57	0.79
It is easy to add attributes in MOT	3	2.96	0.84
It is easy to edit attributes in MOT	3	2.89	0.79
It is easy to create relationships in MOT	3	2.96	0.58
It is easy to delete relationships in MOT	3	3.04	0.58
It is easy to add labels in MOT	3	3.1	0.79
It is easy to delete labels in MOT	3	3.14	0.76

Table 10.9: Responses to questions about 'ease of use'

This indicates that most students thought the tool was fairly easy to use. Based on the average response, it appears that copying concepts and reuse of content was the most difficult. However (as described earlier in this section) the students' task

[60] http://www.blogger.com

[61] http://www.wordpress.com

[62] http://www.joomla.co.uk

[63] http://zooma.sourceforge.net/index.html

[64] http://astah.net/

was to only create one domain and one goal map, this meant that they did not need to use many of the reuse features (such as the Wikipedia import, and copying concepts between domains). Similarly, 19 out of the 28 respondents (67.86%) answered '*yes*' to the question "*Do you think reuse (of existing content) is sufficiently supported by MOT4.0.1?*". Also, when asked "*Did you like the fact that you could use the same Domain for different strategies, and the same strategy for different Domains, or would you have preferred them to be edited together?"* 23 out of the 28 respondents said they appreciated the fact that they could use the same domain for different strategies.

With regard to the '*Calculate relatedness*' feature, 11 of the 28 respondents (39.29%) said they tried it once, with 9 saying they didn't know about the feature, 7 said they '*used it sometimes*' and 1 person said they'd used it '*often*'.

21 out of the 28 respondents (75%) agreed with the statement "*The MOT toolset is coherent*". A further 2 respondents strongly agreed with the statement whilst 5 respondents disagreed. One of the comments provided said that "*Importing stuff over and over in ADE is painful*". This is an issue that has been identified in Chapter 7, and could be addressed by closely integrating the two tools in the future.

23 out of the 28 respondents (82%) said that they felt they had enough flexibility to label the content in an appropriate way according to their strategy. The only negative comment was that "*If I convert a domain to a course, I can't edit the domain without losing the labels.*" This comment could be based on a misconception about the usage of the tools, since editing attributes will automatically update the sublessons in a course. However, it is true that if a domain

is dramatically restructured, the author will need to rebuild the course. Future work could investigate how to update a goal model based on a modified domain model.

To investigate whether users prefer to manually or automatically rearrange the domain graph, the users were asked to rate the series of statements shown in Table 10.10. The options presented to the respondents were *Never*, *Sometimes*, *Often* and *Very Often*, which were coded 0-3 respectively. The question began:

"*When editing/viewing a Domain, how often did you use the following arrangement features…?*"

When editing/viewing a Domain, how often did you use the following arrangement features…	***Most frequent answer (Mode)***	***Mean***	***Standard Deviation***
Manual (Drag & Drop)	2	2.07	0.86
Automatic Rearrange	3	2.1	0.92
Automatic Rearrange with some manual (Drag & Drop) adjustment	3	2.1	0.88
Tree Layout	1	1.54	1.1
Tree Layout with some manual (Drag & Drop) adjustment	1	1.43	0.96

Table 10.10: The usage of the Domain rearrangement features

These results appear to show that the automatic rearrange feature was preferred over the manually rearranging the concepts (thus supporting Hypothesis H10.1).

Similarly, the users were asked which layout provided the clearest view of their domain. The options in the scale were *Strongly Disagree*, *Disagree*, *Agree* and *Strongly Agree* which were coded as 1-4 respectively. The results are shown in Table 10.11.

When editing/viewing a Domain, how often did you use the following arrangement features…	*Most frequent answer (Mode)*	*Mean*	*Standard Deviation*
The Domain Auto Arrange feature provided a clear view of my final Domain map	3	3.04	0.79
The Domain Tree View feature provided a clear view of my final Domain map	3	2.93	0.9
The Course feature provided a clear view of the structure of my course	3	3.25	0.52

Table 10.11: The usage of the hierarchical Domain rearrangement features

Interestingly, when viewing a Domain the users appeared to prefer the automatic rearrange functions over the tree layout. However, when viewing the course they seemed to appreciate the hierarchical view. This could be because the tree view of a Domain still involves a graph-based display (as shown in Figure 10.13), however the tree display of a Course is more text-based (as shown in Figure 10.14), and is therefore more readable.

Indeed, when a large number of concepts are added to the domain, the tree view placement occupies a much larger space and becomes more difficult to use. Further research could add an alternative view to the Domain tool showing a hierarchical view similar to that of the Course tool. However, such a view would prevent the non-parent relationships from being displayed, and would therefore be functionally equivalent to MOT3.1 (see Chapter 7).

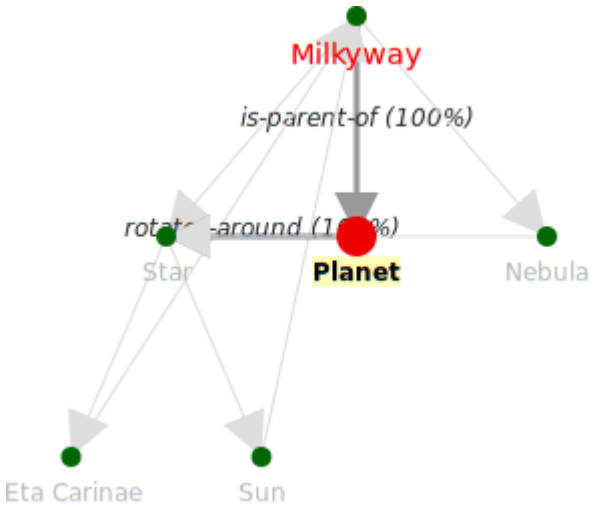

Figure 10.13: The tree view of a Domain in MOT4.01

- **[Milkyway]**
  - title
  - text
  - **[Star]**
    - title
    - text
    - **[Eta Carinae]**
      - title
      - text
    - **[Sun]**
      - title
      - text
  - **[Planet]**
    - title
    - text
  - **[Nebula]**
    - title
    - text

Figure 10.14: The tree view of a Course in MOT4.01

The students were also asked what they thought to the tooltips. 23 out of the 28 respondents said there were '*Enough*' tooltips, whilst 3 said there were '*Too many*'. A further 2 students said there were '*Not enough*'. Similarly, 24 out of the 28 students (85.7%) said that the information in the tooltips was '*Helpful*', with a further 2 people saying they were '*Very helpful*' and another 2 saying they were '*Unhelpful*'. Overall these results show that the tooltips were useful, thus supporting Hypothesis H10.5.

10.7.7.1 SUS Test

The final part of the survey was based around the System Usability Scale (SUS) [88] questionnaire, which asks the users to rate statements based on a 5-point Likert scale ('*Strongly Disagree*' to '*Strongly Agree*', numbered 1-5 respectively). The statements and their response averages are shown in Table 10.12. Note that each row alternates between a positive and a negative statement. The score for each element is also presented. In accordance with the procedure for analysing the SUS questionnaire, the score is calculated by subtracting 1 from all responses to the odd numbered items, and subtracting the responses to all even numbered items from 5.

Statement	*Mode*	*Mean*	*Standard Deviation*	*Score*
1. I think that I would like to use this system frequently	3	2.93	0.9	1.93
2. I found the system unnecessarily complex	2	2.25	0.75	2.75
3. I thought the system was easy to use	4	3.61	0.74	2.61
4. I think that I would need the support of a technical person to be able to use this system	1	1.96	0.96	3.04
5. I found the various functions in this system were well integrated	4	3.54	0.88	2.54
6. I thought there was too much inconsistency in this system	2	2.46	0.88	2.54
7. I would imagine that most people would learn to use this system very quickly	4	3.71	0.81	2.71
8. I found the system very cumbersome to use	2	2.61	1.03	2.39
9. I felt very confident using the system	4	3.75	0.84	2.75
10. I needed to learn a lot of things before I could get going with this system	3	2.64	1.02	2.36

Table 10.12: Responses to each part of the SUS study

The next stage of the SUS process is to sum all the scores, and multiply by 2.5. The final SUS score for the system is therefore 64. According to Sauro [141], this is slightly below the average mark of 68. Figure 10.15 shows the mean of each score displayed on a radar chart. This style of display was used in [46], [54], and – due to the alternating positive and negative nature of the statements – ideally the graph should be star shaped. However, the graph indicates that the students appeared to broadly agree with most of the statements; positive and negative. Out of the positive statements, the smallest response was for "*I think that I would like to use this system frequently*". This is unsurprising, since the students were not taught the

background of the long-term usage of adaptive hypermedia – they were merely asked to create domain maps and goal maps. They may therefore be feeling that they have no further use for the tools. Interestingly, the highest of the negative statements was "*I needed to learn a lot of things before I could get going with this system*" with a mean of 2.64, yet the statement "*I would imagine that most people would learn to use this system very quickly*" yielded an even higher mean of 3.71.

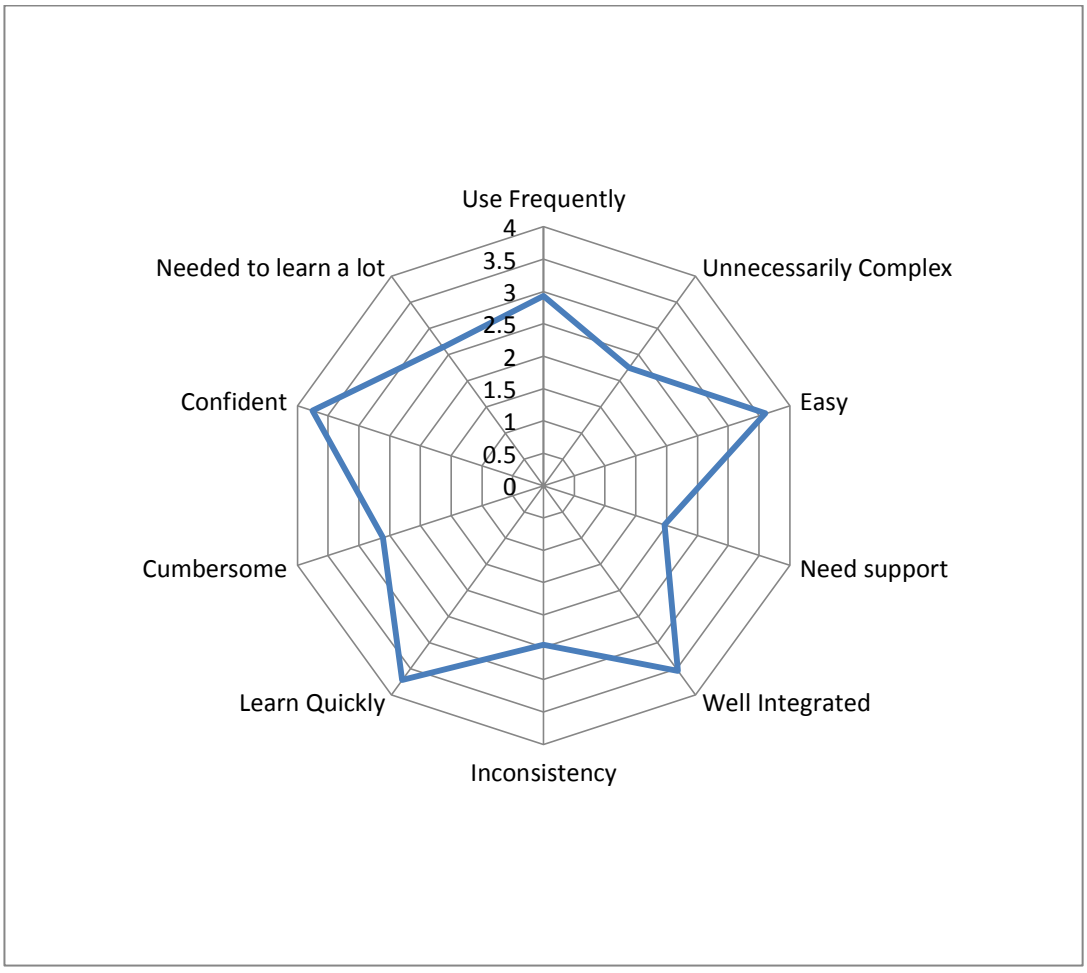

Figure 10.15: The SUS results shown as a radar graph

10.8 Conclusions

This chapter has documented the creation of MOT4, which was designed to extend the functionality of the MOT toolset. Specifically, MOT4 was designed to mimic (and improve upon) the more graphical methods of visualizing Domain maps, thus attempting to encourage users to specify domains that specify more complex relationships between concepts than the usual hierarchical parent relationships that are typically used within MOT.

The long-term usage evaluation of MOT4.0 (described in Section 10.6.1) did not show an increase in the complexity of the domain maps or goal maps created by the students. However, it is suggested that this was due to external factors, including a change in the schedule of the coursework and a lack of documentation for the new features within LAG.

The chapter described the improvements that were made in response to qualitative feedback from this initial evaluation, thus creating MOT4.01. The evaluation of MOT4.01 was carried out by providing students with sample strategies that specifically used concept relationships, thus allowing the students to use more complex domains.

In the follow-up questionnaire, the students were undecided about how consistent MOT4.01 is with other applications, thus not confirming H10.6. This indicates that the process of authoring adaptation is still rather different to the style of authoring that many computer users are familiar with.

The students appeared to like the automatic rearrangement of concepts in the domain visualization (thus supporting H10.3). However, they appeared to prefer the text-based layout (as exhibited by the goal map view) over the graphical tree structure. This suggests that the Domain should allow users to choose whether they want to view a graph-style domain (as in MOT4 and GAT) or a tree-style domain (as in MOT3.1).

11 Conclusions and Future Work

11.1 Introduction

This thesis has investigated ways of improving the authoring process for adaptive hypermedia. Although the thesis has primarily explored the authoring process through the evaluation of two separate tools, the evaluations have attempted to isolate the various features that contribute to the usability of the tool, so that such features can be integrated into other authoring systems.

11.2 Summary of Evaluation Results

The research looked at how visualization can help users with the adaptation authoring process. The evaluation presented in Chapter 5 showed that the PEAL IDE appeared to improve the quantitative complexity (statistically significant with a large effect size when considering the amount of code) of the submitted strategies, compared with strategies that had previously been submitted. This indicates the importance of providing basic editing features (including code completion, syntax highlighting and a code library). However, when introducing the visual features of PEAL2, there was no evidence to suggest that the submitted LAG strategies had improved. It seems likely that this was due to the fact that complex strategies extended beyond the browser's window, making it difficult for the user to visualize the entire strategy.

Further research into the visualization of adaptation strategies was presented in Chapters 9 and 10 with reference to the GRAPPLE Authoring Tool. The evaluation in Chapter 9 indicated that most users preferred to write whole adaptation strategies (as in LAG, Section 9.6.2.8) rather than focussing on specific smaller parts of

adaptation, however, the users also prefer to reuse small pieces of adaptation (Section 9.6.2.7). However the evaluations found no conclusive results about whether authors prefer writing with high granularity or low granularity of static content.

In general, there was an inconclusive result concerning the visualization of the (sub)-strategies. One of the reasons for this may be that the evaluation participants were Computer Science students – this means that they are familiar with the process of writing code.

The evaluation of MOT4 presented in Section 10.6.1.1 investigated whether the introduction of a graph-based structure had any effect on the quality of the domain maps compared with the tree-based structure of MOT3.1. There was a small (but statistically insignificant) reduction in the number of concepts created in MOT4. This could be an indication that a tree-based structure more clearly conveys the structure of a course.

Interestingly, the graph-based structure did nothing to encourage the students in the initial MOT4 evaluation to create non-parent relationships. This could be due to the fact that the LAG4 adaptation language implemented within ADE [63] was relatively new, and as such did not provide concrete examples for the usage of such relationships.

The evaluation with MOT4.01 did however attempt to encourage the usage of such complex relationships by providing an example strategy. When asked what they thought about the visualization of these relationships, the users appeared to prefer

the automatic (force-directed [139]) rearrangement or even a manual placement rather than utilizing the tree layout function.

The UK students in the initial MOT4 evaluation only used hierarchical relationships, this may indicate that in most cases domains can be authored using only a hierarchical tree view. For this reason, a future tool could offer students two views of the domain; one with a hierarchical text-based tree (such as the one used in MOT3.1 or the course tool of MOT4), and another view with the graph-based structure (as used in MOT4).

The ability to address content using more than one pedagogical label was investigated in MOT3.1. MOT4 later refined this with the LAF XML format; however in the evaluation, none of the students used multiple labelling. This could have been due to a lack of multiple labelling examples in the LAG4 documentation.

When investigating whether students preferred labelling (in MOT) or sockets (in GAT) in the Chapter 9 evaluation (H9.6), there was no clear preference shown.

An approximately equal number of students said they liked to edit the content within the tool compared with editing in an external editor. Some of the evaluation comments described in Chapter 10 suggested that some users would appreciate the option of being able to directly link to external resources, rather than having to use the internal editor. Future research could therefore provide users with the option to link to external resources within MOT4.

11.3 Importing Content from Linear Sources

Chapter 4 described a set of imperatives for creating an importer that can generate adaptive content based on static material. The chapter described the creation of two such importers – a Wiki importer and a presentation importer. Evaluations (with both students and course authors) demonstrated that authors generally desired the functionality provided by each of these importers. In response to user feedback, the Wiki importer was extended to also extract the images from the article.

11.4 Future Research

Chapter 4 described how the Wiki importer could further be extended to crawl Wikipedia pages, and import several related articles at a time. An initial prototype has investigated using DBPedia [142], which is a structured data set of information that has been extracted from Wikipedia pages. This information can be downloaded into RDF triples [91] that represent the relationships between two articles. Using the graph-based relationship types made available by MOT4, it is possible to create a domain that contains concepts (generated from Wikipedia articles) that have semantically meaningful relationships with other concepts. For instance, a concept named '*PHP*' would be involved in an '*influencedBy*' link with the '*C++*' concept. A LAG strategy could therefore be created that can display all concepts that are '*influencedBy*' C++. This would effectively be a semantic enricher for the Wikipedia importer, with clear parallels to the work by Hendrix et al. [89] (described in Chapter 4).

11.5 Future Applications

Initiatives such as Coursera[143] and FutureLearn[144] aim to engage learners through the use of Massive Open Online Courses (MOOCs), allowing any member of the public to follow an online course. However, MOOCs often have a high student-to-teacher ratio, often with thousands of students and a single teacher[145]. This means that learners may not receive tuition that is specifically directed to them. Personalisation technologies (such as adaptive hypermedia) could be used to simulate the guidance of a real-world tutor. Currently, many universities use VLEs such as Moodle [104] to deliver interactive activities, such as quizzes and forums. However, such content is rarely adaptive or personalised since it does not store a user model, and is therefore limited in how it can recommend content. Therefore, there is an even greater need for educators to be able to create engaging teaching materials through the expression of complex personalisation techniques. To assist authors with this challenging task, it is vital that usable authoring tools are created. Such authoring tools could be created based on the principles established in this thesis.

Indeed, the thesis has described the design and development of a number of authoring tools, and has described the results of evaluations to identify how to make the authoring process easier. The evaluations presented in this thesis have been primarily targeted at Computer Science students, who may have different preferences to those of non-technical educators. Nevertheless, it was important that the approval of technical users was sought before expecting the tool to be usable by non-technical authors.

Further research could concentrate more heavily on the pedagogical utility of delivering adaptive hypermedia. This could be achieved by testing adaptation tools in a long-term teaching environment, enriching existing real-world teaching content with adaptation techniques. This would allow the long-term educational benefits of various adaptation strategies to be evaluated, as well as providing a greater understanding of how authoring tools can be used by non-technical educators.

Encouraging a variety of authors to engage with the process of authoring for adaptive hypermedia would aim to create a richer variety of static content and pedagogical adaptation strategies. By employing the principles of reuse documented in this thesis, a wider variety of educators would be able to share, reuse and collaborate with each other on more engaging, personalised courses. This would provide a larger number of case studies to demonstrate the pedagogical utility of adaptive hypermedia.

Appendix A: MOT3.1/PEAL2 versus GAT Questionnaire

In the following, you will be asked a number of questions on your authoring experience with two tools, GAT, which you have just used at a recent demo, and MOT3.1, with which you have been authoring your coursework.

Please read the questions carefully, as they are asking very specific opinions on your experience. If you have not understood the question, it's best to ask someone or to comment on this in the comment field provided for each question.

The first set of questions is for beginner authors: these can edit and modify content, concepts and some relations, but do not have to directly edit adaptation strategies, sub-strategies or pieces of strategies.

Please answer first these questions on content creation and manipulation in domains in the authoring process of adaptive presentations.

*

1. When authoring an adaptive course or presentation, do you prefer to use (select as many options as you wish):

☐ Using a dedicated, integrated editing tool (like in MOT3.1)

☐ Editing from scratch

☐ Pre-existing content only

☐ Pre-existent content which can be further edited

☐ Editing HTML files in any HTML editor (like in GAT)

Any comment?

2. Regarding the granularity of the content you are creating when authoring an adaptive course or presentation, do you prefer:

○ Don't mind either way

○ High granularity (small 'chunks' of content that can be reordered in a page, like in MOT3.1)

○ Low granularity (big 'chunks' of content that basically form a page, like in GAT)

Any comment?

*

3. Regarding the overall experience in authoring content for an adaptive course or presentation, do you think:

☐ MOT3.1 is easier to learn how to use

☐ GAT is easier to learn how to use

☐ MOT 3.1 and GAT are similar in the amount it takes me to learn how to use them

☐ After learning how to use them, MOT 3.1 content is easier to use

☐ After learning how to use them, GAT content is easier to use

☐ After learning how to use them, both GAT and MOT3.1 content are of similar difficulty of use

☐ Visualisation of content in MOT3.1 is better

☐ Visualisation of content in GAT is better

☐ The visualisation of content in MOT3.1 and GAT are of similar quality

☐ When preparing and using content, GAT is faster

☐ When preparing and using content, MOT3.1 is faster

☐ The speed of content usage and preparation in GAT and MOT3.1 is similar
Any comment?

Please answer next these questions about concept creation and manipulation in domains when authoring adaptive presentations.

*

4. Regarding the difficulty in learning how to use and manipulate concepts, which system do you think takes less effort to learn:

	MOT3.1	**GAT**	**both the same**	**N/A**
cloning	○ * cloning MOT3.1	○ cloning GAT	○ cloning both the same	○ cloning N/A
reusing	○ reusing MOT3.1	○ reusing GAT	○ reusing both the same	○ reusing N/A
copy	○ copy MOT3.1	○ copy GAT	○ copy both the same	○ copy N/A
creation	○ creation MOT3.1	○ creation GAT	○ creation both the same	○ creation N/A
dragging	○ dragging	○ dragging	○ dragging	○ dragging

	MOT3.1	**GAT**	**both the same**	**N/A**
deleting	MOT3.1 deleting MOT3.1	GAT deleting GAT	both the same deleting both the same	N/A deleting N/A

Any comment?

*

5. Regarding the overall experience in authoring concepts for an adaptive course or presentation, do you think:

After learning how to use them, MOT 3.1 concepts are easier to use

After learning how to use them, GAT concepts are easier to use

After learning how to use them, both GAT and MOT3.1 concepts are of similar difficulty of use

Visualisation of concepts in MOT3.1 is better

Visualisation of concepts in GAT is better

The visualisation of concepts in MOT3.1 and GAT are of similar quality

When preparing and using concepts, GAT is faster

When preparing and using concepts, MOT3.1 is faster

The speed of concepts usage and preparation in GAT and MOT3.1 is similar

Any comment?

*

6. Regarding order of concepts and usability, I think that:

Reusing the same DM for different GM orders of concepts and thus for different final presentations is better (like in MOT3.1).

Preserving the order of concepts in the DM for the final presentation is better (like in GAT).

The two ordering mechanisms are of equal value to me.

Any comment?

*

7. Regarding the properties of concepts, I think that:

I have no clear opinion about the best number of properties for concepts.

Having only one property for DM concepts (the type of the attributes that constitute it) used in multiple attributes is better (like in MOT3.1).

Having multiple properties for DM concepts is better (like in GAT).

Any comment?

*

8. Regarding the overall experience in authoring properties of concepts for an adaptive course or presentation, do you think:

MOT3.1 properties of concepts (types) are easier to learn how to use

GAT properties of concepts is easier to learn how to use

MOT 3.1 and GAT properties of concepts are similar in the amount it takes me to learn how to use them

After learning how to use them, MOT 3.1 properties of concepts (types) are easier to use

After learning how to use them, GAT properties of concepts are easier to use

After learning how to use them, both GAT and MOT3.1 properties of concepts are of similar difficulty of use

Visualisation of properties of concepts (types) in MOT3.1 is better

Visualisation of properties of concepts in GAT is better

The visualisation of content in MOT3.1 and GAT of properties of concepts are of similar quality

When preparing and using properties of concepts, GAT is faster

When preparing and using properties of concepts (types), MOT3.1 is faster

The speed of properties of concepts usage and preparation in GAT and MOT3.1 is similar

Any comment?

*

9. For relations in the DM, I prefer:

Multiple types of relations (predefined or created by the author, as in GAT)

Two types relations (hierarchical and relatedness, as in MOT 3.1)

I have no strong opinions on the number of relation types necessary for a DM.
Any comment?

*

10. Regarding the parent-child relation in the DM and issues of reuse of the DM, I believe that:

It doesn't make any difference to me if the hierarchy is defined in the DM or in a separate step (GM or PRT).

There should be an intermediary step to decide if this DM concept hierarchy is to be interpreted as a hierarchy in the final presentation (like in MOT3.1, where the hierarchy is only established in the GM)

It should be possible to directly relate this DM concept hierarchy to the final presentation (like in GAT)
Any comment?

11. Regarding the overall experience in authoring properties of DM relations for an adaptive course or presentation, do you think:

MOT3.1 DM relations are easier to learn how to use

GAT DM relations are easier to learn how to use

MOT 3.1 and GAT DM relations are similar in the amount it takes me to learn how to use them

After learning how to use them, MOT 3.0 DM relations are easier to use

After learning how to use them, GAT DM relations are easier to use

After learning how to use them, both GAT and MOT3.1 DM relations are of similar difficulty of use

Visualisation of properties of DM relations in MOT3.1 is better

Visualisation of properties of DM relations in GAT is better

☐ The visualisation of DM relations in MOT3.1 and GAT are of similar quality

☐ When preparing and using DM relations, GAT is faster

☐ When preparing and using DM relations, MOT3.1 is faster

☐ The speed of DM relations usage and preparation in GAT and MOT3.1 is similar
Any comment?

12. Regarding adaptation labelling and reusability, as well as expressivity, do you prefer:
☐ Labelling for adaptation strategies via the socket mechanism (potentially multiple sockets; like in GAT, CAM tool)

☐ Labelling: for adaptation strategies via multiple labels and weights in the GM (like in MOT 3.1)

☐ Labelling for adaptation strategies via the DM (like in GAT, DM tool)
Any comment?

*

13. In terms of granularity of strategy reuse, I prefer reusing:
☐ Specific parts of adaptation strategies (such as initialization and implementation, like in PEAL+LAG)

☐ Small pieces of adaptation (like in GAT, CAM tool)

☐ Whole adaptation strategies (like in PEAL+LAG)

☐ I consider reuse of whole adaptation strategies, specific parts, or small pieces of adaptation of similar value.
Any comment?

*

14. Regarding the complexity of the resulting adaptation, I believe that I can create strategies of greater complexity with:
○ MOT 3.1

○ GAT

○ I can create strategies of similar complexity with both GAT and MOT3.1
Any comment?

*

15. Regarding the overall experience in authoring PRTs (pedagogical relations in GAT, CAM tool) or reusing multiple (sub-)strategies in PEAL (with reuse syntax for LAG as given at course, reuse of initialization or of implementation) for an adaptive course or presentation, do you think:

☐ PEAL (sub-)strategy reuse syntax is easier to learn how to use

☐ GAT PRT relations are easier to learn how to use

☐ PEAL and GAT sub-strategy use are similar in the amount it takes me to learn how to use them

☐ After learning how to use them, PEAL (sub-)strategies are easier to use

☐ After learning how to use them, GAT PRT relations are easier to use

☐ After learning how to use them, both GAT PRT and PEAL (sub-)strategies are of similar difficulty of use

☐ Visualisation of (sub-)strategies in PEAL is better

☐ Visualisation of PRTs in GAT is better

☐ The visualisation of PEAL (sub-)strategies and GAT PRT are of similar quality

☐ When preparing and using PRTs, GAT is faster

☐ When preparing and using (sub-)strategies, MOT3.0 is faster

☐ The speed of PEAL (sub-)strategies and GAT PRT usage and preparation is similar

Any comment?

16. Regarding the overall experience in delivering an adaptive course or presentation, do you think:

☐ ADE is easier to learn how to use

☐ GALE is easier to learn how to use

☐ ADE and GALE are similar in the amount it takes me to learn how to use them

☐ After learning how to use it, GALE is easier to use

☐ After learning how to use it, ADE is easier to use

☐ After learning how to use them, both ADE and GALE are of similar difficulty of use

☐ Visualisation in ADE is better

☐ Visualisation in GALE is better

☐ The visualisation in GALE and ADE are of similar quality

☐ GALE is faster

☐ ADE is faster

☐ The speeds of ADE and GALE are similar
Any comment?

Authors are considered advanced if they author the code of an adaptation strategy.

Please answer these questions about editing strategy code, as you've been working as advanced authors in PEAL, and have seen what type of work needs done by advanced authors in GAT (via the CRT tool).

*

17. In terms of granularity of strategy writing, I prefer writing:
☐ Small pieces of adaptation (like in GAT, CRT tool)

☐ Specific parts of adaptation strategies (such as initialization and implementation, like in PEAL+LAG)

☐ Whole adaptation strategies (like in PEAL+LAG)

☐ I consider writing of whole adaptation strategies, specific parts, or small pieces of adaptation of similar difficulty.
Any comment?

*

18. Regarding the overall experience in writing code in an adaptation language for an adaptive course or presentation, do you think:
☐ LAG code is easier to learn how to use

☐ GALE code is easier to learn how to use

☐ LAG and GALE code are similar in the amount it takes me to learn how to use them

☐ After learning how to use it, GALE code is easier to use

☐ After learning how to use it, LAG code is easier to use

☐ After learning how to use them, both LAG and GALE code are of similar difficulty of use

☐ Visualisation of LAG code is better (in PEAL)

☐ Visualisation of GALE code is better (in GAT, PRT tool)

☐ The visualisations of GALE and LAG code are of similar quality (in GAT, PRT tool, respectively PEAL)

☐ GALE code writing is faster (in GAT, PRT tool)

☐ LAG code writing is faster (in PEAL)

☐ The speeds of writing LAG and GALE code are similar (in PEAL, respectively GAT, PRT tool)
Any comment?

*

19. Away from GAT & MOT+PEAL, would you prefer:
☐ Graph-based tools for concept editing

☐ hierarchical tools for concept editing

☐ I have no definite preference for graph-based tools or hierarchical tools

☐ definition of high level granularity of adaptation strategies (small pieces)

☐ definition of low level granularity of adaptation strategies (large pieces)

☐ I have no definite preference for high or low level granularity for adaptation strategies (small or large pieces)
Any comment?

Appendix B: MOT4 versus GAT survey

Thanks for looking at this questionnaire, we are trying to make our adaptive hypermedia authoring systems as easy to use as possible, and we would really value your opinion.

1) How experienced are you with traditional Web Development*

( ) None - I've never done Web development

( ) Basic - I have used WYSIWYG editors or content management systems

( ) Advanced - I am confident in directly writing source code for HTML/CSS

( ) Expert - I can do all the above, and have used server side technologies

2) Other than the initial demonstration seminar, please estimate the number of hours you (personally) have spent working with MOT*

Please Comment (if necessary)

Learning and Using

Authoring systems may differ in the time you need to invest in learning them. For instance, if a system is radically different to anything you've ever used before, it may take you a fair amount of time to learn how to use it. We are interested here in your estimation of the time it takes you to learn how to use a system.

3) For each of the following common Domain operations, which system do you think is easier to learn?*

	MOT	GAT
Creating concepts	( )	( )
Copying concepts	( )	( )
Cloning concepts	( )	( )
Reusing concepts	( )	( )
Create/Edit relationships	( )	( )

Please comment

4) Considering only the course creation tools, which system do you think is easier to learn?*

( ) MOT (Goal Model tool)

( ) GAT (Course tool)

Please comment

5) Overall, which toolset do you think is easier to learn?*

( ) The MOT Toolset (Domain + Goal + LAG)

( ) The GAT Toolset (Domain + Pedagogical rules + Course tools)

Please comment

Considering you have already learned how to use them well, systems can differ nevertheless in how easy it is to use them. We are interested here in your estimation of how easy it is in your opinion to use these systems, once you have learned them.

6) For each of the following common operations, which system do you think is easier to use?*

	MOT	GAT
Creating concepts	( )	( )
Copying concepts	( )	( )
Cloning concepts	( )	( )
Reusing concepts	( )	( )
Create/Edit relationships	( )	( )

Please comment

7) Considering only the course creation tools, which system do you think is easier to use?*

( ) MOT (Goal Model tool)

( ) GAT (Course tool)

Please comment

8) Overall, which toolset do you think is easier to use?*

( ) The MOT Toolset (Domain + Goal + LAG)

( ) The GAT Toolset (Domain + Pedagogical rules + Course tools)

Please comment

Content visualization

9) The main difference between the display of the Domain in MOT and GAT is the auto-arranging of concepts in MOT. Did you use this feature, or did you prefer to arrange concepts yourself?*

( ) Auto arrange (as in MOT)

( ) Manual arrange (as in GAT)

Please comment:

10) MOT provides you with a separate tool (the Goal Map) to design the hierarchy of your course. In GAT you design the hierarchy by using Domain relationships and properites. Which of these hierarchy creating methods did you prefer?*

( ) Creating hierarchies in MOT

( ) Creating hierarchies in GAT

Please comment

Functionality

11) In GAT you create concepts which link to entire URIs. In MOT you create many attributes to describe a concept. Which of these did you prefer?*

( ) Attributes (as in MOT)

( ) Linking to URIs (as in GAT)

Please comment

12) In GAT course adaptation is specified by using pedagogical rules to build an entire course. In MOT, strategies are applied to entire goal maps. Which style of adaptation authoring did you prefer?*

( ) Using entire strategies at a time (as in MOT)

( ) Using smaller 'building blocks' (as in GAT)

Please comment

Both MOT and GAT attempt to allow content (both static content and adaptation) to be shared between users. GAT allows you to make your content private, shared or world-writeable. Whereas in MOT all content is readable by other users, but can only be shared by creating group accounts or sending XML and LAG files to each other.

13) Which method of user permissions and sharing do you prefer?*

( ) Group accounts (or sharing LAG/XML files) in MOT

( ) Public/Shared/Private permissions in GAT

Please comment

Authoring flexibility

14) Imagine you had created your own domain content and you wanted to reuse somebody else's adaptation. Would you rather...*

( ) Label your content in MOT, and reuse an entire adaptation strategy

( ) Create a Course in GAT using existing pedagogical rules

Please comment

15) Imagine you wanted to present content according to a particular adaptive strategy, which has not yet been written. Would you rather...*

( ) Create a GAT course from existing PRTs

( ) Write a whole LAG file

Please comment

Overall

16) How confident are you in using the following areas...*

	Very Uncertain	Uncertain	Neutral	Confident	Very Confident
MOT Domain	( )	( )	( )	( )	( )
MOT Goal	( )	( )	( )	( )	( )
LAG Code	( )	( )	( )	( )	( )
GAT Domain	( )	( )	( )	( )	( )
GAT Course	( )	( )	( )	( )	( )
GAT Pedagogical Rules (creating from scratch)	( )	( )	( )	( )	( )

Any comments?

17) If you were given enough training in each system, which authoring system do you believe would provide you with the greatest flexibility?*

( ) MOT

( ) GAT

Please explain your answer

Thank You!

Thank you for taking our survey. Your response is very important to us.

Appendix C: MOT4.01 Survey

Thank you for taking this questionnaire about MOT4.

Please be assured that your answers will not affect your coursework mark (your name is required to ensure that you have submitted the questionnaire).

Name:

1) How experienced are you with traditional Web Development*

( ) None - I've never done Web development

( ) Basic - I have used WYSIWYG editors or content management systems

( ) Advanced - I am confident in directly writing source code for HTML/CSS

( ) Expert - I can do all the above, and have used server side technologies

2) Which coursework option did you choose?*

( ) Option 1 (using ADE and LAG files)

( ) Option 2 (using XSLT/XPath/XQuery to query the LAF file)

Why did you choose this option?*

_____

3) Within your coursework group, which of these tasks did you personally do?*

[ ] Content
[ ] Domain/Relationships
[ ] Course Labelling
[ ] XSLT/XPath/XQuery (Option 2 only)
[ ] Other (please specify)

4) Approximately how many hours did you personally spend using MOT?*

_____

Familiarity

5) Please rate the following statements:*

	Strongly Disagree	Disagree	Agree	Strongly Agree
MOT is consistent with other applications I have used	( )	( )	( )	( )
The interface for editing (adding/deleting/copying concepts) a Domain structure was consistent with other software I have used	( )	( )	( )	( )
The interface for adding/deleting attributes was consistent with other software I have used	( )	( )	( )	( )
The interface for editing the content of attributes was consistent with other software I have used	( )	( )	( )	( )
The interface for rearranging a course (by dragging the tree) was consistent with other software I have used	( )	( )	( )	( )
The interface for adding labels to a course was consistent with other software I have used	( )	( )	( )	( )

Please comment. What other software have you used that is similar to this?

6) Please rate the following statements:*

	Strongly disagree	Disagree	Agree	Strongly agree
It is easy to create concepts in MOT	( )	( )	( )	( )
It is easy to copy concepts in MOT	( )	( )	( )	( )
It is easy to delete concepts in MOT	( )	( )	( )	( )
It is easy to reuse concepts in MOT	( )	( )	( )	( )
It is easy to add attributes in MOT	( )	( )	( )	( )

It is easy to edit attributes in MOT	( )	( )	( )	( )
It is easy to create relationships in MOT	( )	( )	( )	( )
It is easy to delete relationships in MOT	( )	( )	( )	( )
It is easy to add labels in MOT	( )	( )	( )	( )
It is easy to delete labels in MOT	( )	( )	( )	( )

Please comment

Features

7) Do you think reuse (of existing content) is sufficiently supported by MOT4.0?*

( ) Yes

( ) No

Please comment

8) Did you like the fact that you could use the same Domain for different strategies, and the same strategy for different Domains, or would you have preferred them to be edited together?*

( ) Same domain for different strategies (edited separately)

( ) Edit adaptation together with domain

Please comment

9) Are you familiar with the LAOS framework for specifying adaptation?*

( ) Very Familiar

( ) Familiar

( ) Slightly familiar

( ) Unfamiliar

10) Please rate the following statement: "My knowledge of the LAOS framework helped me in dealing with the MOT4 system"*

( ) Strongly disagree
( ) Disagree
( ) Agree
( ) Strongly agree
Please comment

11) How difficult was it to distribute roles within your group whilst working with MOT4.0?*

( ) Very Easy
( ) Easy
( ) Difficult
( ) Very Difficult
Please comment

Support and Functionality

Did you like the fact that MOT4.0 exports the content to XML? If you would have preferred another format, which one would that be and why?*

13) Did you know that MOT4.0 can help you with suggesting related concepts? Have you used this facility?*

( ) I used it frequently
( ) I used it sometimes
( ) I tried it once
( ) I didn't know about this
What other adaptive facilities would you have wanted?*

Please rate the following statement: "The MOT4.0 toolset is coherent"*

( ) Strongly Agree
( ) Agree
( ) Disagree
( ) Strongly Disagree

Please comment

15) Did you feel that you had enough flexibility to label the content in an

appropriate way according to your desired strategy?*

( ) Yes
( ) No
Please explain your answer, especially what you might have missed.

Visualization

16) When editing/viewing a Domain, how often did you use the following

arrangement features?*

	Never	Sometimes	Often	Very Often
Manual (Drag&Drop)	( )	( )	( )	( )
Automatic Rearrange	( )	( )	( )	( )
Automatic Rearrange with some manual (Drag&Drop) adjustment	( )	( )	( )	( )
Tree Layout	( )	( )	( )	( )
Tree Layout with some manual (Drag&Drop) adjustment	( )	( )	( )	( )

Please comment:

17) Please rate the following statements:*

	Strongly disagree	Disagree	Agree	Strongly agree
The Domain Auto Arrange feature provided a clear view of my final Domain map	( )	( )	( )	( )
The Domain Tree View feature provided a clear view of my final Domain map	( )	( )	( )	( )
The Course feature provided a clear view of the structure of my course.	( )	( )	( )	( )

Please comment:

Support and Documentation

18) What did you think to the number of tooltips?*

( ) Too many
( ) Enough
( ) Not enough

19) The information contained in the tooltips was:*

( ) Very Unhelpful
( ) Unhelpful
( ) Helpful
( ) Very Helpful

20) The documentation (http://ade.dcs.warwick.ac.uk/ADE-LAG4/) was:*

( ) Very Unhelpful
( ) Unhelpful
( ) Helpful
( ) Very Helpful

21) How often did you refer to the documentation

(http://ade.dcs.warwick.ac.uk/ADE-LAG4/)?*

( ) Frequently (every time I used MOT)
( ) Often (when I needed to learn something new)
( ) Occassionally (if I needed to check something)
( ) Never (I understood how to use the system immediately)

Please comment

Usability

22) Please rate the following statements about MOT (please ignore factors such as

LAG code, or XSLT/XQuery).*

	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I think that I would like to use this system frequently	( )	( )	( )	( )	( )
I found the system unnecessarily complex	( )	( )	( )	( )	( )
I thought the system was easy to use	( )	( )	( )	( )	( )
I think that I would need the support of a technical person to be able to use this system	( )	( )	( )	( )	( )
I found the various functions in this system were well integrated	( )	( )	( )	( )	( )
I thought there was too much inconsistency in this system	( )	( )	( )	( )	( )
I would imagine that most people would learn to use this system very quickly	( )	( )	( )	( )	( )
I found the system very cumbersome to use	( )	( )	( )	( )	( )
I felt very confident using the system	( )	( )	( )	( )	( )

I needed to learn a lot of things before I could get going with this system	( )	( )	( )	( )	( )

23) If you'd be interested in providing further feedback via a Skype conversation,

please write your email address here...

_____

Please use this space to write any other comments about the system

Thank You!

Thank you for taking our survey. Your response is very important to us.

References

[1] P. Brusilovsky and W. Nejdl, “Adaptive hypermedia and adaptive web,” in Practical Handbook of Internet Computing, M. P. Singh, Ed. CRC Press, 2003.

[2] P. Brusilovsky, S. Sosnovsky, and M. Yudelson, “Adaptive hypermedia Services for E-learning,” in Workshop of Adaptive Hypermedia Techniques to Service Oriented Environment at Adaptive Hypermedia Conference, AH 2004, Eindhoven, The Netherlands, 23-26 Aug 2004, pp. 470–479.

[3] A. I. Cristea, D. Smits, and P. De Bra, “Towards a generic adaptive hypermedia platform: a conversion case study,” Journal of Digital Information, vol. 8, no. 3, 2007.

[4] P. De Bra, D. Smits, K. Sluijs, A. Cristea, J. Foss, C. Glahn, and C. M. Steiner, “GRAPPLE: Learning Management Systems Meet Adaptive Learning Environments,” in Intelligent and Adaptive Educational-Learning Systems, vol. 17, A. Peña-Ayala, Ed. Springer Berlin Heidelberg, 2013, pp. 133–160.

[5] B. W. Boehm, “A spiral model of software development and enhancement,” Computer, vol. 21, no. 5, pp. 61–72, May 1988.

[6] M. J. Norušis and SPSS Inc., SPSS professional statistics 6.1. Prentice Hall, 1994.

[7] A. Field, Discovering Statistics Using SPSS (Introducing Statistical Methods series). Sage Publications Ltd, 2009.

[8] A. Field and G. J. Hole, How to Design and Report Experiments. Sage Publications Ltd, 2003.

[9] R. Likert, “A Technique for the Measurement of Attitudes,” Archives of Psychology, vol. 22, no. 140, pp. 1–55, 1932.

[10] J. Robertson, “Likert-type scales, statistical methods, and effect sizes,” Communications of the ACM, vol. 55, no. 5, pp. 6–7, May 2012.

[11] P. Cairns and A. L. Cox, “Using statistics in usability research,” in Research Methods for Human-Computer Interaction, P. Cairns and A. L. Cox, Eds. Cambridge University Press, 2008, pp. 112–137.

[12] P. Brusilovsky, “Methods and techniques of adaptive hypermedia,” User Modeling and User-Adapted Interaction, vol. 6, no. 2–3, pp. 87–129, Jul 1996.

[13] P. Brusilovsky, “Adaptive Hypermedia,” User Modeling and User-Adapted Interaction, vol. 11, no. 1–2, pp. 87–110, 2001.

[14] P. Brusilovsky, “Developing adaptive educational hypermedia systems: From design models to authoring tools,” in Authoring Tools for Advanced Technology Learning Environment, T. Murray, S. Blessing, and S. Ainsworth, Eds. Dordrecht: Kluwer Academic Publishers, 2003, pp. 377–409.

[15] P. De Bra, L. Aroyo, and A. Cristea, “Adaptive Web-based Educational Hypermedia,” in Web dynamics adaptive to change in content size topology

and use, M. Levene and A. Poulovassilis, Eds. Springer Verlag, 2004, pp. 387–410.

[16] P. De Bra, “Adaptive educational hypermedia on the web,” Communications of the ACM, vol. 45, no. 5, pp. 60–61, May 2002.

[17] N. V. Stash, A. I. Cristea, and P. De Bra, “Authoring of Learning Styles in Adaptive Hypermedia: Problems and Solutions,” in Proceedings of the 13th International World Wide Web conference on Alternate track papers & posters - WWW Alt. ’04, New York, NY, USA, 17-22 May 2004, pp. 114–123.

[18] P. Paredes and P. Rodriguez, “A Mixed Approach to Modelling Learning Styles in Adaptive Educational Hypermedia,” Advanced Technology for Learning, vol. 1, no. 4, pp. 210–215, 2004.

[19] N. Stash, A. Cristea, and P. De Bra, “Learning Styles Adaptation Language for Adaptive Hypermedia,” in Proceedings of the Adaptive Hypermedia (AH) 2006 conference, Dublin, Ireland, 2006, 21-23 Jun 2006, pp. 323–327.

[20] E. Knutov, P. De Bra, and M. Pechenizkiy, “AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques,” New Review of Hypermedia and Multimedia, vol. 15, no. 1, pp. 5–38, Apr 2009.

[21] P. Brusilovsky and L. Pesin, “Visual annotation of links in adaptive hypermedia,” in CHI ’95 Conference Companion on Human Factors in Computing Systems, New York, New York, USA, 7–11 May 1995, pp. 222–223.

[22] P. De Bra, P. Brusilovsky, and G.-J. Houben, “Adaptive hypermedia: from systems to framework,” ACM Computing Surveys, vol. 31, no. 4es, p. 12–es, Dec 1999.

[23] P. De Bra and J.-P. Ruiter, “AHA! Adaptive Hypermedia for All,” in Proceedings of WebNet 2001 – World Conference on the WWW and the Internet, Orlando, FL, USA, 23-27 Oct 2001, pp. 262–268.

[24] G. Weber and P. Brusilovsky, “ELM-ART: An Adaptive Versatile System for Web-based Instruction,” International Journal of Artificial Intelligence in Education (IJAIED), vol. 12, pp. 351–384, 2001.

[25] P. De Bra and N. Stash, “AHA ! Adaptive Hypermedia for All,” in Proceedings Of The System Administration and Network Engineering (SANE) 2002 Conference, Maastricht, The Netherlands, 27-31 May 2002, pp. 411–412.

[26] D. Smits and P. De Bra, “GALE : A Highly Extensible Adaptive Hypermedia Engine,” in Proceedings of The 22nd ACM Conference on Hypertext and Hypermedia (Hypertext 2011), Eindhoven, The Netherlands, 6-9 Jun 2011, pp. 63–72.

[27] J. Eklund and P. Brusilovsky, “InterBook: An Adaptive Tutoring System,” UniServe Science News, vol. 12, no. 3, pp. 8–13, 1999.

[28] P. De Bra, T. Santic, and P. Brusilovsky, “AHA! meets Interbook, and more...,” in Proceedings of World Conference on E-Learning (E-Learn), Phoenix, Arizona, USA, 7-11 Nov 2003, pp. 57–64.

[29] H. Wu, G. Houben, and P. De Bra, “Aham: A reference model to support adaptive hypermedia authoring,” in Proceedings of the Conference on Information Science, Antwerp, Belgium, 12-16 Jul 1998, pp. 77–88.

[30] P. De Bra, G. Houben, and H. Wu, “AHAM: a Dexter-based reference model for adaptive hypermedia,” Proceedings of the ACM Conference on Hypertext and Hypermedia (Hypertext), 21-25 Feb, 1999, Darmstadt, Germany. pp. 147–156, 1999.

[31] F. Halasz and M. Schwartz, “The Dexter hypertext reference model,” Communications of the ACM, vol. 37, no. 2, pp. 30–39, Feb 1994.

[32] P. De Bra, A. Aerts, D. Smits, and N. Stash, “AHA! meets AHAM,” in Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002), Málaga, Spain, 29-31 May 2002, pp. 381–384.

[33] P. Vrieze, P. Bommel, and T. Weide, “A Generic Adaptivity Model in Adaptive Hypermedia,” in Adaptive Hypermedia and Adaptive Web-Based Systems, P. De Bra and W. Nejdl, Eds. Springer Berlin Heidelberg, 2004, pp. 344–347.

[34] N. Koch and M. Wirsing, “The Munich reference model for adaptive hypermedia applications,” in Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2002), Málaga, Spain, 29-31 May 2002, pp. 213–222.

[35] A. I. Cristea and A. De Mooij, “LAOS: Layered WWW AHS Authoring Model and their corresponding Algebraic Operators,” in The Twelfth International

World Wide Web Conference (WWW'03), Budapest, Hungary, 20–24 May 2003.

[36] A. Cristea and K. Kinshuk, "Considerations on LAOS, LAG and their Integration in MOT," in Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia), Honolulu, Hawaii, 23–28 Jun 2003, pp. 511–518.

[37] M. Hendrix, P. De Bra, M. Pechenizkiy, D. Smits, and A. Cristea, "Defining Adaptation in a Generic Multi Layer Model: CAM: The GRAPPLE Conceptual Adaptation Model," Times of Convergence Technologies Across Learning Contexts, vol. 5192, pp. 132–143, 2008.

[38] GRAPPLE, "Welcome to the GRAPPLE project Website," 2012. [Online]. Available: http://www.grapple-project.org/. [Accessed: 23-Sep-2012].

[39] Microsoft, "Microsoft Word 2010 - Get started with Word 2010," 2012. [Online]. Available: http://office.microsoft.com/en-gb/word. [Accessed: 22-Sep-2012].

[40] P. De Bra, N. Stash, and B. De Lange, "AHA! Adding adaptive behavior to websites," in Proceedings of the NLUUG Conference, Ede, The Netherlands, 22 May 2003.

[41] P. De Bra and N. Stash, "Multimedia adaptation using AHA!," in Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia), Lugano, Switzerland, 21-26 Jun 2004, pp. 563–570.

[42] P. De Bra, D. Smits, K. Van Der Sluijs, A. I. Cristea, and M. Hendrix, “GRAPPLE: Personalization and Adaptation in Learning Management Systems,” in Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (EdMedia), Toronto, Canada, 28 Jun-2 Jul 2010, pp. 3029–3038.

[43] M. Hendrix, A. I. Cristea, and C. Stewart, “Adaptation languages for learning: the CAM meta-model,” in Ninth IEEE International Conference on Advanced Learning Technologies (ICALT 2009), Riga, Latvia, 15-17 Jul 2009, pp. 104–106.

[44] A. Mazzetti, K. van der Sluijs, and M. Dicerto, “Data models and related documentation-final version,” 2010. [Online]. Available: http://wwwis.win.tue.nl/grapple/public-files/deliverables/GRAPPLE-D7.2c-Data models-v1.0.pdf. [Accessed: 23-Sep-2012].

[45] D. Albert, A. Nussbaumer, C. M. Steiner, M. Hendrix, and A. I. Cristea, “Design and development of an authoring tool for pedagogical relationship types between concepts,” in Proceedings of the 17th International Conference on Computers in Education (ICCE 2009), Hong Kong, 30 Nov-4 Dec 2009.

[46] M. Hendrix, “Supporting Authoring of Adaptive Hypermedia,” PhD Thesis, Department of Computer Science, University of Warwick, 2010.

[47] A. I. Cristea and A. de Mooij, “Adaptive course authoring: My Online Teacher,” in 10th International Conference on Telecommunications (ICT-2003), Papeete, French Polynesia, 23-28 Feb 2003, pp. 1762–1769.

[48] N. Stash and P. De Bra, “Building Adaptive Presentations with AHA! 2.0,” in Proceedings of the PEG Conference, Saint Petersburg, Russia, 28 Jun-1 Jul 2003.

[49] A. Cristea, “Adaptive Course Creation for All,” in International Conference on Information Technology: Coding and Computing (ITCC’04), Las Vegas, Nevada, 5-7 Apr 2004, pp. 718–722.

[50] J. Scotton and A. I. Cristea, “Reusing Adaptation Strategies in Adaptive Educational Hypermedia Systems,” in 10th IEEE International Conference on Advanced Learning Technologies (ICALT 2010), Sousse, Tunisia, 5-7 Jul 2010, pp. 528–532.

[51] O. Conlan, V. Wade, C. Bruen, and M. Gargan, “Multi-model, metadata driven approach to adaptive hypermedia services for personalized elearning,” in Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH’02), Málaga, Spain, 29-31 May 2002, pp. 100–111.

[52] J. A. Muzio, T. Heins, and R. Mundell, “Experiences with reusable E-learning objects,” The Internet and Higher Education, vol. 5, no. 1, pp. 21–34, Jan 2002.

[53] A. I. Cristea, D. Smits, J. Bevan, and M. Hendrix, “LAG 2.0: Refining a reusable Adaptation Language and Improving on its Authoring,” in 4th European Conference on Technology Enhanced Learning (ECTEL’09), Nice, France, 29 Sep-2 Oct 2009, pp. 7–21.

[54] A. Cristea, C. Stewart, T. Brailsford, and P. Cristea, “Evaluation of Interoperability of Adaptive Hypermedia Systems: testing the MOT to WHURLE conversion in a classroom setting,” in International Workshop on Authoring of Adaptive Adaptable Educational Hypermedia (A3EH 2005), Amsterdam, The Netherlands, 19 Jul 2005.

[55] C. Stewart, A. I. Cristea, T. Brailsford, and H. Ashman, “‘Authoring Once, Delivering Many’: Creating Reusable Adaptive Courseware,” in 4th IASTED International Conference on WebBased Education, Grindelwald, Switzerland, 21-23 Feb 2005, pp. 21–23.

[56] A. I. Cristea, D. Smits, and P. De Bra, “Writing MOT, Reading AHA! - converting between an authoring and a delivery system for adaptive educational hypermedia,” in International Workshop on Authoring of Adaptive Adaptable Educational Hypermedia (A3EH 2005), Amsterdam, The Netherlands, 19 Jul 2005, pp. 36–45.

[57] F. Ghali, A. Cristea, and C. Stewart, “My Online Teacher 2.0,” in 3rd European Conference on Technology Enhanced Learning (EC-TEL 2008), IGACLE workshop, Maastricht, The Netherlands, 16 Sep 2008.

[58] F. Ghali, “Social Personalized E-Learning Framework,” PhD Thesis, Department of Computer Science, University of Warwick, 2010.

[59] F. Ghali and A. I. Cristea, “Social Reference Model for Adaptive Web Learning,” in Advances in Web Based Learning (ICWL), Aachen, Germany, 19-21 Aug 2009, pp. 162–171.

[60] F. Ghali and A. I. Cristea, “Evaluation of Interoperability between MOT and Regular Learning Management Systems,” in Third European Conference on Technology Enhanced Learning (EC-TEL 2008), Maastricht, The Netherlands, 16-19 Sep 2008, pp. 104–109.

[61] A. Cristea and M. Verschoor, “The LAG grammar for authoring the adaptive web,” in International Conference on Information Technology: Coding and Computing (ITCC’04), Las Vegas, Nevada, Apr 2004, vol. 1, pp. 382–386.

[62] A. I. Cristea and M. Hendrix, “LAG Strategies,” 2010. [Online]. Available: http://prolearn.dcs.warwick.ac.uk/strategies.html. [Accessed: 20-Sep-2012].

[63] J. Scotton, S. Moebs, J. McManis, and A. Cristea, “Merging strategies for authoring QoE-based adaptive hypermedia,” Journal of Universal Computer Science, vol. 16, no. 19, pp. 2576–2779, 2010.

[64] M. Larranaga, I. Niebla, U. Rueda, A. Arruarte, and J. A. Elorriaga, “Towards Collaborative Domain Module Authoring,” in Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007), Niigata, Japan, 18-20 Jul 2007, pp. 814–818.

[65] M. Dougiamas and P. Taylor, “Moodle: Using Learning Communities to Create an Open Source Course Management System,” in Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA), Honolulu, Hawaii, 23-28 Jun 2003, pp. 171–178.

[66] J. Farmer and I. Dolphin, “Sakai: eLearning and more,” EUNIS: Leadership and Strategy in a Cyber-Infrastructure World, 2005. [Online]. Available: http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/IMM/S050622F.pdf . [Accessed: 04-Mar-2013].

[67] T. Lewin, “Universities reshaping education on the web,” The New York Times. Retrieved from, 2012. [Online]. Available: http://www.immagic.com/eLibrary/ARCHIVES/GENERAL/GENPRESS/N120717L.pdf. [Accessed: 26-Sep-2012].

[68] M. Freire and P. Rodriguez, “Comparing graphs and trees for adaptive hypermedia authoring,” in Third International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia (A3EH) at the 12th International Conference on Artificial Intelligence in Education (AIED), Amsterdam, The Netherlands, 18 Jul 2005.

[69] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, “Scratch: programming for all,” Communications of the ACM, vol. 52, no. 11, pp. 60–67, Nov 2009.

[70] D. Wolber, “App inventor and real-world motivation,” in Proceedings of the 42nd ACM Technical Symposium on Computer Science Education (SIGCSE), Dallas, TX, USA, 9-11 Mar 2011, pp. 601–606.

[71] Microsoft, “on{X} - automate your life,” 2012. [Online]. Available: http://www.onx.ms. [Accessed: 25-Sep-2012].

[72] H. Ossher and P. Tarr, “Multi-Dimensional Separation of Concerns and the Hyperspace Approach,” in Software Architectures and Component Technology, M. Akşit, Ed. Springer US, 2002, pp. 293–323.

[73] M. Pohja, M. Honkala, and P. Vuorimaa, “An XHTML 2.0 Implementation,” Web Engineering, vol. 3140, pp. 402–415, 2004.

[74] P. De Bra and L. Calvi, “AHA! a generic Adaptive Hypermedia System,” in Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia at Hypertext’98, Pittsburgh, PA, USA, 20 Jun 1998, pp. 5–12.

[75] A. I. Cristea, F. Ghali, and M. Joy, “Social, personalized lifelong learning,” in E-Infrastructures and Technologies for Lifelong Learning: Next Generation Environments, G. Magoulas, Ed. Hershey, PA: IGI Global, 2011, pp. 90–125.

[76] N. Baloian, J. Pino, and O. Motelet, “Collaborative Authoring, Use, and Reuse of Learning Material in a Computer-Integrated Classroom,” in Groupware: Design, Implementation, and Use, vol. 2806, J. Favela and D. Decouchant, Eds. Springer Berlin Heidelberg, 2003, pp. 199–207.

[77] J. Nielsen, “The usability engineering life cycle,” Computer, vol. 25, no. 3, pp. 12–22, Mar 1992.

[78] M. Hendrix and A. I. Cristea, “A Qualitative and Quantitative Evaluation of Adaptive Authoring of Adaptive Hypermedia,” in Second European Conference on Technology Enhanced Learning (EC-TEL 2007), Crete, Greece, 17-20 Sep 2007, pp. 71–85.

[79] M. Soegaard, “Affordances,” 2010. [Online]. Available: http://www.interaction-design.org/encyclopedia/affordances.html. [Accessed: 04-Mar-2013].

[80] M. Hendrix, A. Cristea, and W. Nejdl, “Authoring adaptive educational hypermedia on the semantic desktop,” International Journal of Learning Technology, Special Issue on Authoring of Adaptive and Adaptable Hypermedia, vol. 3, no. 3, p. 230, 2007.

[81] J. S. Zepeda and S. V Chapa, “From Desktop Applications Towards Ajax Web Applications,” in 4th International Conference on Electrical and Electronics Engineering. (ICEEE 2007), Mexico City, Mexico, 5–7 Sep 2007, pp. 7–10.

[82] J. J. Garret, “Ajax: A New Approach to Web Applications,” 2005. [Online]. Available: http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. [Accessed: 04-Mar-2013].

[83] B.-M. Yu and S.-Z. Roh, “The effects of menu design on information-seeking performance and user’s attitude on the World Wide Web,” Journal of the

American Society for Information Science and Technology, vol. 53, no. 11, pp. 923–933, 2002.

[84] S.-M. Huang, K.-K. Shieh, and C.-F. Chi, “Factors affecting the design of computer icons,” International Journal of Industrial Ergonomics, vol. 29, no. 4, pp. 211–218, Apr 2002.

[85] F. Ghali and A. I. Cristea, “MOT 2.0: A Case Study on the Usefuleness of Social Modeling for Personalized E- Learning Systems,” in 14th International Conference on Artificial Intelligence in Education (AIED 2009), Brighton, UK, 6-10 Jul 2008, vol. 200, pp. 333–340.

[86] T. Berners-Lee, R. Fielding, and H. Frystyk, “HyperText Transfer Protocol,” 1996. [Online]. Available: http://www.w3.org/History/19921103-hypertext/hypertext/WWW/Protocols/HTTP.html. [Accessed: 22-Sep-2012].

[87] F. Ghali and A. I. Cristea, “Interoperability between MOT and learning management systems: converting CAF to IMS QTI and IMS CP,” in Fifth International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH’08), Hannover, Germany, 29 Jul-1 Aug 2008, pp. 296–299.

[88] J. Brooke, “SUS-A quick and dirty usability scale,” in Usability evaluation in industry, P. W. Jordan, B. Thomas, I. L. McClelland, and B. Weerdmeester, Eds. Taylor & Francis, 1996, pp. 189–194.

[89] M. Hendrix and A. I. Cristea, “A spiral model for adding automatic, adaptive authoring to adaptive hypermedia,” Journal Of Universal Computer Science,

Special Issue on Authoring of Adaptive and Adaptable Hypermedia, vol. 14, no. 17, pp. 2799–2818, 2008.

[90] P.-A. Chirita, S. Costache, W. Nejdl, and R. Paiu, “Beagle++ : Semantically Enhanced Searching and Ranking on the Desktop,” in 3rd European Semantic Web Conference (ESWC), Budva, Montenegro, 11-14 Jun 2006, pp. 348–362.

[91] W3C, “W3C RDF/XML Syntax Specification (Revised),” 2004. [Online]. Available: http://www.w3.org/TR/REC-rdf-syntax. [Accessed: 26-Sep-2012].

[92] “Microsoft PowerPoint 2010 - Buy, try and explore.” [Online]. Available: http://office.microsoft.com/en-gb/powerpoint/. [Accessed: 22-Sep-2012].

[93] “Apache OpenOffice - The Free and Open Productivity Suite.” [Online]. Available: http://www.openoffice.org/. [Accessed: 22-Sep-2012].

[94] “LaTeX – A document preparation system.” [Online]. Available: http://latex-project.org/. [Accessed: 22-Sep-2012].

[95] “Wikipedia.” [Online]. Available: http://www.wikipedia.org/. [Accessed: 22-Sep-2012].

[96] “MediaWiki.” [Online]. Available: http://www.mediawiki.org/wiki/MediaWiki. [Accessed: 22-Sep-2012].

[97] “Extension:WikiEditor - MediaWiki.” [Online]. Available: http://www.mediawiki.org/wiki/Extension:WikiEditor. [Accessed: 22-Sep-2012].

[98] D. Raggett, A. Le Hors, I. Jacobs, and others, “HTML 4.01 Specification,” W3C Recommendation, 24 Dec 1999. [Online]. Available: http://www.w3.org/TR/html4/. [Accessed: 06-Mar-2013].

[99] Adobe, “PDF Reference and Adobe Extensions to the PDF Specification.” [Online]. Available: http://www.adobe.com/devnet/pdf/pdf_reference.html. [Accessed: 06-Mar-2013].

[100] E. Ramp, P. De Bra, and P. Brusilovsky, “Authoring and Delivery of Adaptive Electronic Textbooks made Easy,” in Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (E-Learn), Vancouver, British Columbia, Canada, 24-28 Oct 2005, pp. 142–149.

[101] E. Brown, A. Cristea, C. Stewart, and T. Brailsford, “Patterns in authoring of adaptive educational hypermedia: a taxonomy of learning styles,” Educational Technology & Society, vol. 8, no. 3, pp. 77–90, 2005.

[102] “OWL - Semantic Web Standards,” 2012. [Online]. Available: http://www.w3.org/2001/sw/wiki/OWL. [Accessed: 22-Sep-2012].

[103] P. Winston and B. Horn, Lisp. Reading, MA: Addison Wesley, 1986.

[104] G. Rößling and A. Kothe, “Extending moodle to better support computing education,” in Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE’09), New York, New York, USA, 6 Jul 2009, vol. 41, no. 3, pp. 146–150.

[105] M. Grabe and K. Christopherson, “Optional student use of online lecture resources: resource preferences, performance and lecture attendance,” Journal of Computer Assisted Learning, vol. 24, no. 1, pp. 1–10, Feb 2007.

[106] Wikipedia, “PHP,” 2012. [Online]. Available: http://en.wikipedia.org/wiki/PHP. [Accessed: 24-Sep-2012].

[107] B. Luyt, T. C. H. Aaron, L. H. Thian, and C. K. Hong, “Improving Wikipedia’s accuracy: Is edit age a solution?,” Journal of the American Society for Information Science and Technology, vol. 59, no. 2, pp. 318–330, Jan 2008.

[108] J. Giles, “Internet encyclopaedias go head to head,” Nature, vol. 438, no. 7070, pp. 900–1, Dec 2005.

[109] MediaWiki, “API:Main page.” [Online]. Available: http://www.mediawiki.org/wiki/API. [Accessed: 22-Sep-2012].

[110] Wikipedia, “Help:Category,” 2012. [Online]. Available: http://en.wikipedia.org/wiki/Help:Category. [Accessed: 22-Sep-2012].

[111] L. Page, S. Brin, R. Motwani, and T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web.,” Stanford InfoLab, 1999.

[112] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’ networks,” Nature, vol. 393, no. 6684, pp. 440–442, 1998.

[113] S. Dolan, “Six degrees of Wikipedia,” 2008. [Online]. Available: http://mu.netsoc.ie/wiki/. [Accessed: 22-Sep-2012].

[114] Moodle, “Import PowerPoint - MoodleDocs,” 2012. [Online]. Available: http://docs.moodle.org/22/en/Import_PowerPoint. [Accessed: 22-Sep-2012].

[115] K. van Deemter and R. Power, “Authoring multimedia documents using WYSIWYM editing,” in Proceedings of the 18th Conference on Computational Linguistics (COLING’00), Morristown, NJ, USA, 31 Jul-4 Aug 2000, vol. 1, pp. 222–228.

[116] TeX Users Group, “TeX4ht,” 2012. [Online]. Available: http://tug.org/tex4ht/. [Accessed: 22-Sep-2012].

[117] T. Mikkonen and A. Taivalsaari, “Web Applications: Spaghetti code for the 21st century,” in Sixth International Conference on Software Engineering Research, Management and Applications (SERA’08), Prague, Czech Republic, 20-22 Aug 2008, pp. 319–328.

[118] M. Kruk, “PDFTOHTML conversion program,” 2011. [Online]. Available: http://pdftohtml.sourceforge.net/. [Accessed: 22-Sep-2012].

[119] A. Cristea and L. Calvi, “The Three Layers of Adaptation Granularity,” in 9th International Conference on User Modeling (UM’03), Johnstown, PA, USA, 22-26 Jun 2003, pp. 4–14.

[120] G. D. Ruxton, “The unequal variance t-test is an underused alternative to Student’s t-test and the Mann-Whitney U test,” Behavioral Ecology, vol. 17, no. 4, pp. 688–690, Apr 2006.

[121] J. Cohen, “A power primer,” Psychological bulletin, vol. 112, no. 1, pp. 155–159, 1992.

[122] R. M. Felder and B. A. Solomon, “Index of learning styles,” 2010. [Online]. Available: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSpage.html. [Accessed: 22-Sep-2012].

[123] YouTube, “YouTube On Your Site,” 2012. [Online]. Available: http://www.youtube.com/youtubeonyoursite. [Accessed: 20-Sep-2012].

[124] A. Mesbah and A. Van Deursen, “Invariant-Based Automatic Testing of AJAX User Interfaces,” in Proceedings of the 31st International Conference on Software Engineering (ICSE’09), Vancouver, BC, Canada, 16-24 May 2009, pp. 210–220.

[125] E. Folmer, M. van Welie, and J. Bosch, “Bridging patterns: An approach to bridge gaps between SE and HCI,” Information and Software Technology, vol. 48, no. 2, pp. 69–89, 2006.

[126] J. Tennison, “Hash URIs,” W3C Blog, 12 May 2011. [Online]. Available: http://www.w3.org/QA/2011/05/hash_uris.html. [Accessed: 22-Sep-2012].

[127] A. Aizawa, “An information-theoretic perspective of tf–idf measures,” Information Processing & Management, vol. 39, no. 1, pp. 45–65, 2003.

[128] I. Nassi and B. Shneiderman, “Flowchart techniques for structured programming,” ACM SIGPLAN Notices, vol. 8, no. 8, pp. 12–26, Aug 1973.

[129] GRAPPLE, “Summary,” 2011. [Online]. Available: http://www.grapple-project.org/summary. [Accessed: 23-Sep-2012].

[130] E. L. M. Ploum, “Authoring of Adaptation in the GRAPPLE Project,” Technische Universiteit Eindhoven, 2009.

[131] GRAPPLE, “GRAPPLE Tutorial.” [Online]. Available: http://gale.win.tue.nl:10080/gale/concept/gale://gale.tue.nl/course/grapple/GALE. [Accessed: 23-Sep-2012].

[132] A. Nussbaumer, “Final Implementation of the Concept Relationship Type Tool,” 30 Dec 2010. [Online]. Available: http://wwwis.win.tue.nl/grapple/public-files/deliverables/D3.2c-WP3-CRTToolFinal-v1.0.pdf. [Accessed: 23-Sep-2012].

[133] F. Abel, D. Heckmann, E. Herder, and D. H. Fabian Abel, “Mashing up user data in the Grapple User Modeling Framework,” in Workshop on Adaptivity and User Modeling in Interactive Systems (ABIS’09), 21-23 Aug 2009, pp. 1–2.

[134] Microsoft, “Access Database Software and Applications,” 2012. [Online]. Available: http://office.microsoft.com/en-gb/access/. [Accessed: 24-Sep-2012].

[135] G. J. Wills, “Selection: 524,288 ways to say ‘this is interesting’,” in Proceedings IEEE Symposium on Information Visualization (INFOVIS’96), San Francisco, CA, USA, 28 Oct 1996, pp. 54–60, 120.

[136] M. Pazzani and D. Billsus, “Content-Based Recommendation Systems,” in The Adaptive Web, P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Springer Berlin Heidelberg, 2007, pp. 325–341.

[137] L. Oneto, F. Abel, E. Herder, and D. Smits, “Making today’s Learning Management Systems adaptive,” in Learning Management Systems meet Adaptive Learning Environments, Workshop at European Conference on Technology Enhanced Learning (EC-TEL), Nice, France, 29 Sep-2 Oct 2009.

[138] J. Khan, A. Cristea, and C. Stewart, “Adaptive Authoring of Adaptive Hypermedia Towards, Role-based, Adaptive Authoring,” in Computers and Advanced Technology in Education (CATE), Cambridge, UK, 11-13 Jul 2011.

[139] T. M. J. Fruchterman and E. M. Reingold, “Graph drawing by force-directed placement,” Software: Practice and Experience, vol. 21, no. 11, pp. 1129–1164, 1991.

[140] F. Ghali, M. Sharp, and A. Cristea, “Folksonomies and Ontologies in Authoring of Adaptive Hypermedia,” in 6th International Workshop on Authoring of Adaptive and Adaptable Hypermedia Workshop (A3H) at 5th Int. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems (AH’08), Hannover, Germany, 29 Jul-1 Aug 2008.

[141] J. Sauro, “Measuring Usability with the System Usability Scale (SUS),” Measuring Usability, 2 Feb 2011. [Online]. Available: https://www.measuringusability.com/sus.php. [Accessed: 25-Sep-2012].

[142] Sören Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: A Nucleus for a Web of Open Data," in 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, 11-15 Nov 2007, vol. 4825, pp. 722–735.

[143] Coursera, "Coursera," 2013. [Online]. Available: https://www.coursera.org/. [Accessed: 16-Feb-2013].

[144] M. Parry, "Leading British Universities Join New MOOC Venture - Wired Campus - The Chronicle of Higher Education," 13 Dec 2012. [Online]. Available: http://chronicle.com/blogs/wiredcampus/leading-british-universities-join-new-mooc-venture/41211. [Accessed: 16-Feb-2013].

[145] J. Mackness, S. Mak, and R. Williams, "The ideals and reality of participating in a MOOC," in Proceedings of the Seventh International Networked Learning Conference, Aalborg, Denmark, 3–4 May 2010.