

Zakaria, Mohamed Ramzy (2004) The hybrid model, and adaptive educational hypermedia frameworks. PhD thesis, University of Nottingham.

Access from the University of Nottingham repository:

<http://eprints.nottingham.ac.uk/14247/1/404032.pdf>

Copyright and reuse:

The Nottingham ePrints service makes this work by researchers of the University of Nottingham available open access under the following conditions.

- Copyright and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners.
- To the extent reasonable and practicable the material made available in Nottingham ePrints has been checked for eligibility before being made available.
- Copies of full items can be used for personal research or study, educational, or not-for-profit purposes without prior permission or charge provided that the authors, title and full bibliographic details are credited, a hyperlink and/or URL is given for the original metadata page and the content is not changed in any way.
- Quotations or similar reproductions must be sufficiently acknowledged.

Please see our full end user licence at:

http://eprints.nottingham.ac.uk/end_user_agreement.pdf

A note on versions:

The version presented here may differ from the published version or from the version of record. If you wish to cite this item you are advised to consult the publisher's version. Please see the repository url above for details on accessing the published version and note that access may require a subscription.

For more information, please contact eprints@nottingham.ac.uk

**The Hybrid Model,
And
Adaptive Educational Hypermedia
Frameworks**

By

Mohamed Ramzy Zakaria
BSc, MSc

Thesis Submitted to the
University of Nottingham
for the degree of Doctor of Philosophy,
July 2003

Table of Contents

Abstract	vii
Declaration	viii
Dedication	ix
Acknowledgments	x
List of Figures	xi

PART I: Literature Review

Chapter I: Introduction

1.1 Introduction	1
1.1.1 ITS background.....	2
1.1.2 Hypertext background.....	4
1.1.3 Why adaptive educational hypermedia?	6
1.2 Problems Under Study	8
1.3 Proposed Solution Overview.....	10
1.4 Research Motivation and Objectives	13
1.5 Thesis Structure.....	13

Chapter II: User Modelling and User Modelling Systems

2.1 Introduction	15
2.2 Types of User Modelling.	18
2.2.1 Standard User Models and Individual User Models	18
2.2.2 Automatic (Implicit) Models and Collaborative (Explicit) Models.....	19
2.2.2.1 Automatic (implicit) user modelling.....	19
2.2.2.2 Collaborative (explicit) user models	23
2.2.3 Long-term Models and Short-term Models	24
2.3 Representation Techniques	25
2.3.1 Knowledge Representation Techniques.....	25
2.3.1.1 The overlay model.....	26
2.3.1.2 The differential model.....	27
2.3.1.3 Perturbation and Bug Models	28
2.3.1.4 Constraint-based model.....	29

2.3.2 Stereotyping	30
2.4 Generic User Modelling Systems.....	32
2.4.1 GUMS	33
2.4.1.1 Stereotypes	34
2.4.1.2 Default reasoning with rules	35
2.4.1.3 Failure as negation	36
2.4.2 Academic User Modelling Shells.....	36
2.4.3 Commercial User Modelling Servers.....	37
2.5 Summary	38

Chapter III: Adaptation in Hypermedia

3.1 Introduction.....	40
3.2 Content Level Adaptation	40
3.3 Links Level Adaptation.....	44
3.4 Summary	46

PART II: Technology Review

Chapter IV: Survey

4.1 Introduction:.....	48
4.2 Adaptive Educational Hypermedia Systems.....	49
4.2.1 AHA!.....	51
4.2.2 ELM-ART.....	61
4.2.3 NetCoach.....	65
4.2.4 Interbook	67
4.2.5 Metadoc.....	71
4.2.6 Hypadapter	73
4.2.7 CHEOPS	77
4.2.8 TANGOW	78
4.3 Systems Analysis	80
4.4 Summary	83

Chapter V: Technology

5.1 Introduction.....	85
5.2 XML and HTML.....	85
5.3 Related Technology	89
5.3.1 Cascading Style Sheets (CSS).....	89
5.3.2 eXtensible Style Sheet (XSL)	90
5.3.2.1 eXtensible Style Sheet: Transformation (XSLT).....	90
5.3.2.2 eXtensible Style Sheet: Formatting Object (XSL-FO)	92
5.3.3 XML languages.....	93
5.3.3.1 XPath.....	93
5.3.3.2 XLINK	94
5.3.3.3 XPOINTER.....	95
5.3.4 Xinclude	95
5.4 Cocoon Publishing Framework.....	96
5.4.1 eXtensible Server Page (XSP)	98
5.4.2 ESQL tag library (ESQL logicsheet)	100
5.5 Summary	100

PART III: The Contribution

Chapter VI: The Hybrid Model

6.1 Introduction.....	102
6.2 The Hybrid Model Architecture.....	104
6.2.1 Overlay Model	104
6.2.2 Stereotype Model	105
6.2.3 Information Pool	106
6.3 The Hybrid Model and WHURLE.....	108
6.3.1 WHURLE Overview.....	108
6.3.2 The Hybrid Model Information Pool	111
6.3.3 Knowledge Domains.....	111
6.3.4 Knowledge Levels.....	114
6.3.5 Adaptive Lesson Plans	115
6.3.6 Adaptation Mechanism	117

6.4 Summary	121
-------------------	-----

Chapter VII: Implementation

7.1 Introduction	123
7.2 Implementation Algorithm	123
7.3 Database Design	125
7.3.1 System's tables	126
7.3.2 Users' tables	127
7.4 System Components	130
7.4.1 Authentication system	131
7.4.2 Adaptation filter	133
7.4.3 Quiz engine	136
7.4.3.1 Quiz	136
7.4.3.2 Auto-marking engine	139
7.4.3.3 Upgrade engine	140
7.5 Administrative Tools	141
7.5.1 Students' registration tool	142
7.5.2 Lessons' registration tool	143
7.6 Summary	145

PART IV: User Trial and Discussion

Chapter VIII: User Trial

8.1 Introduction	146
8.2 Experiment Design	146
8.2.1 Lesson plan design	146
8.2.2 Pre-quiz design	147
8.3 Methodology	149
8.4 Data Analysis	151
8.5 Students opinion	156
8.5.1 Quantitative questions	156
8.5.2 Qualitative question	157
8.6 Conclusion	159

8.7 Summary159

Chapter IX: Discussion

9.1 Introduction161
9.2 Why the Hybrid Model?161
9.3 The Hybrid Model – Implicit and Explicit.....164
9.4 WHURLE-HM and Adaptation Techniques.....165
9.5 Limitations167
9.6 Further research.....167
9.7 Conclusion169

Appendices

Appendix A: Source Code171
A.1 Authentication system files171
A.2 Adaptation filter180
A.3 History files.....183
A.4 Quiz engine185
 A.4.1 Auto-marking187
 A.4.2 Upgrade engine191
A.5 Administrative tools196
 A.5.1 Students’ registration tool196
 A.5.2 Lessons’ registration tool199
Appendix B: Database Tables204
Appendix C: Snapshots210
C.1 Authentication screens.....210
C.2 Quiz210
 C.2.1 Quiz Screen210
 C.2.2 Results screen212
C.3 Lessons screen213
C.4 Administrative tools214
Appendix D: Guideline215
D.1 How to create an adaptive lesson Plan215

D.2 How to create quiz questions	216
D.2.1 Pre-quizzes	217
D.2.2 Post-quizzes.....	217
Appendix E: Users Quotes	219
Appendix F: Related Publications	221
<i>“The Hybrid Model for Adaptive Educational Hypermedia”</i>	222
<i>“User modelling, and Adaptive Hypermedia Frameworks for Education”</i>	228
<i>“Pluggable user models for adaptive hypermedia in education”</i>	240
Bibliography	244

Abstract

The amount of information on the web is characterised by being enormous, as is the number of users with different goals and interests. User models have been utilized by adaptive hypermedia systems generally and adaptive educational hypermedia systems (AEHS) particularly to personalize the amount of information they have with respect to each individual's knowledge, background and goals.

As a result of the research described herein, a user model called the Hybrid Model has been developed. This model is both generic and abstract, and it extends other models used by AEHS by measuring users' knowledge levels with respect to different knowledge domains simultaneously by utilising well known techniques in the world of user modelling, specifically the Overlay model (which has been modified) and the Stereotype model. Therefore, using the Hybrid Model, AEHS will not be restricted to a single knowledge domain at any one time. Thus, by implementing the Hybrid model, those systems can manage users' knowledge globally with respect to the deployed knowledge domains.

The model has been implemented experimentally in an educational hypermedia system called WHURLE (Web-based Hierarchal Universal Reactive Learning Environment) to verify its aim – managing users' knowledge globally. Moreover, this implementation has been tested successfully through a user trial as an adaptive revision guide for a Biological Anthropology Course.

Furthermore, the infrastructure of the WHURLE system has been modified to embrace the objective of the Hybrid Model. This has led to a novel design that provides the system with the capability of utilising different user models easily without affecting any of its component modules.

Declaration

I hereby declare that this thesis illustrates my own research work and that I composed this thesis fully myself

A handwritten signature in black ink, appearing to read 'Mohamed Ramzy Zakaria', written in a cursive style.

Mohamed Ramzy Zakaria
School of Computer Science and Information Technology
University of Nottingham
July 2003

Dedication

This thesis is dedicated to my mother, Mrs Zeinab Zakaria, and my father, Dr. Ramzy Zakaria. I am everlastingly thankful and grateful for the support, love, and guidance you provided me to reach that stage. Thank you is not enough for what you have offered and done for me. God bless you both.

Acknowledgments

I would like to thank my supervisor Dr. Tim Brailsford for his help, support and guidance during this long journey.

Special thanks are reserved to Dr. Helen Ashman and Dr. Peter Davies for their help and valuable advice during this work.

I would also like to express my appreciation to Dr. Adam Moore, Craig Stewart, and WTG (Web Technology Group) members for their valuable discussions.

I acknowledge with the deepest gratitude my family for their love, support and help that I have received during this challenge.

At last but not least, my sincere thanks to my friends for their understanding and moral support. You were great guys.

List of Figures

1.1 CAI Structure	3
1.2 Domains and Topics	8
2.1 Adaptation Process	15
2.2 Collaborative User Modelling.....	23
2.3 Overlay Model	26
2.4 Differential Model	27
2.5 Perturbation and Bug Models	28
2.6 Constraint-based Model	29
2.7 Stereotype Used in Grundy System	30
2.8 GUMS Interaction	33
2.9 GUMS Stereotypes	34
2.10 GUMS Stereotype Components	35
4.1 AHA! Domain Concepts	58
4.2 Used Adaptation Technique	81
4.3 Used User Modelling Methods	82
5.1 Tree Diagram	87
5.2 XML Document Life Cycle	88
5.3 A Web server and Cocoon interaction	97
6.1 The Components of the Hybrid Model	106
6.2 WHURLE Components' architecture	109
6.3 An Example of WLPML	110
6.4 Graphical representation for the code in Figure 6.3	110
6.5 The Hybrid Model Information Pool in WHURLE-HM	111
6.6 Knowledge Levels	115
6.7 A Simple Extraction from an adaptive lesson plan about HTML	116

6.8 Adaptation Snapshots	118
6.9 History Window	119
6.10 Choosing Lessons	120
6.11 Users Interaction	120
7.1 WHURLE-HM Flowchart	124
7.2 WHURLE-HM Components	131
7.3 WHURLE Old Infrastructure	133
7.4 WHURLE New Infrastructure	134
7.5 Adaptive Lesson Plan	134
7.6 Quiz Engine Components	136
7.7 Quiz File	137
8.1 An Extract from the Anthropology Lesson Plan	147
8.2 An Extract from the Anthropology Pre-quiz	148
8.3 Pre-quiz Results	149
8.4 Post-quiz Results	150
8.5 Normality Test	151
8.6 Distribution Graphs	153
8.7 Statistical Results	154
8.8 Times of Access	155
8.9 Quantitative Questions Results	156
D.1 Adaptive Lesson Map	216

Chapter I: Introduction

1.1 Introduction

The brave new era of the information age has ramifications for all disciplines, at the most fundamental of levels. From education, to commerce and music, the Internet impinges on every field where data and knowledge are important. Arising out of this world-wide network of communications comes the globalisation of information – in which hypermedia tools are at the forefront enabling direct user access to information. It is now easier than ever to create an interactive web site; and with the rise of DHTML (Dynamic Hypertext Markup Language), the various XML (eXtensible Markup Language) technologies, Java and others, this trend promises to continue. As these technologies become more capable and widely accessible, the most important area of research becomes the application of the technology rather than the technology itself.

The application of web technology within the educational arena offers many important benefits to both teachers and students – particularly in the realms of classroom and platform independence. In Web-based educational systems, the educational material can be installed, supported and maintained at one central facility, while likely thousands of users can process it regardless of where they are, what kind of computers they have, what kind of platform they use or what kind of internet connection they have [Brusilovsky 1998].

To give a full background of the evolution of web-based educational systems, the next subsections, Intelligent Tutoring systems (ITS) background and Hypertext background, will give a brief history about Intelligent tutoring systems and hypertext respectively. Both of these technologies contributed to the presence of what are called adaptive educational hypermedia systems (explained below). Furthermore, advantages of using adaptive educational hypermedia are explained in the ‘Why adaptive educational hypermedia?’ subsection.

1.1.1 ITS background

The involvement of technology in education can be traced back to Pressy in 1926 [Pressy 1927, Pressy 1926] when he created his instructional machine that is packed with teachers' multiple-choice questions and answers. This machine delivered questions; thereby learners should answer each question correctly to be able to move to the second one and so on (an immediate feedback). The size of that machine is the same as a typewriter, and it is composed of a bar on the left hand side upon which a sheet of questions is spinning. In addition, to the right hand side there are four keys corresponding to the possible four answers of the given question where a user press any of them to give the associated answer. Furthermore, that machine has two different settings one for testing and the other for teaching. If the machine is set for testing, therefore, every correct answer a user/student provides is counted by a small counter, but if the user answered incorrectly the counter will not count and the user can proceed to the following question. On the other hand, if the machine is set for teaching, therefore, if a user answered a question incorrectly then he/she will not be able to proceed to the following one unless the right answer is supplied. Albeit this machine was mechanical, it incorporated modern learning theories, such as feedback, and educational strategies in its design [Shute and Psotka 1996].

Skinner [Skinner 1954] argued that instrumentals' aid has a profound impact on enhancing human learning. Furthermore, he described a machine, which is close in functionality to Pressy's machine that provides immediate feedback to a student when he/she gives a wrong answer to a question by not showing the following question. He also added that this device *"makes it possible to present carefully designed material in which one problem can depend upon the answer to the proceeding and where, therefore, the most efficient progress to an eventually complex repertoire can be made. Provision has been made for recording the commonest mistakes so that the tapes can be modified as experience dictates"*[Skinner 1954].

The principle upon which Skinner's machine acting towards an answered question either by ringing a bell if the question is answered correctly or not to proceed to the following question if the provided answer to the former one is wrong acts as a main concept behind Computer Assisted Instruction (CAI) or Computer Based Training

(CBT). CAI or CBT is simply a computer program that embedded Programmed Instruction (PI), where PI “refers to any instructional methodology that utilizes a systematic approach to problem decomposition and teaching”[Shute and Psotka 1996, p 571].

CAI systems present to users/students materials to be studied and then presents a problem belonging to that part of the material to be solved. If a student/user answered correctly then he/she will move to the next one, but if not, remediation is called for, which is a part of the program, that presents simpler problems to the user that gradually move towards the depth of the original material. Figure 1.1 illustrates the structure of CAI.

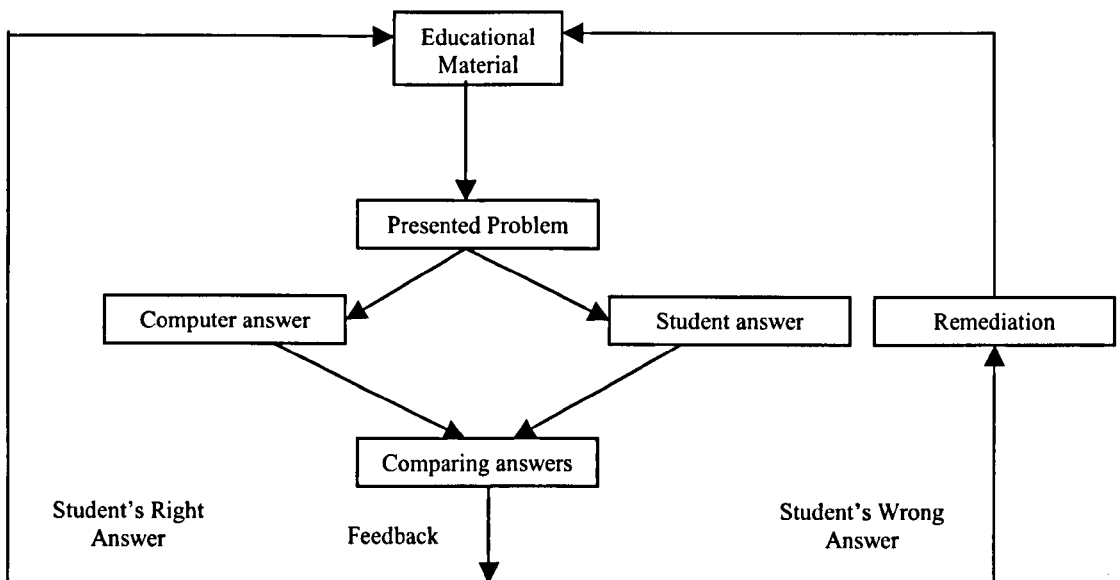


Figure 1.1 CAI Structure - CAI systems compare students' answers with predefined answers (computer answers). If both of them matched then the answer is correct and the student will then go to the next part in the curriculum, but if not the remediation part of the system is activated. This diagram is a modification of what is in [Shute and Psotka 1996].

It is important to notice that the system is not doing that intelligently, but the teacher constructs all branches of the program ahead of time. CAI cannot differentiate between different wrong answers, and it provides the same guidance for all students that answer the same question incorrectly. For that reason, intelligent computer assisted instruction (ICAI) programs evolved.

ICAI provides a kind of personalized tutoring for each student individually, i.e., different students may answer the same question incorrectly but according to the

given answer by each of them, tailored tutoring is provided. For ICAI to achieve that, to view errors intellectually, much research in different fields related to artificial intelligent area, such as efficient representation and knowledge retrieval techniques, and to cognitive psychology area, such as knowledge representation in humans' memory, are carried out [Shute and Psotka 1996].

ITS (Intelligent Tutoring system) considered to be a specific type of ICAI, because it possess a domain model, knowledge about teaching strategies and a student model (knowledge about learners) but ICAI is a domain knowledge free. [Hartly and Sleeman 1973, Ohlsson 1987, Kass 1989].

Albeit ITS or ICAI systems are keen about tutoring students in solving problems, the presented educational material (manual) is the same for all users or students, despite the fact that students themselves are different.

1.1.2 Hypertext background

The term “hypertext” was invented by Ted Nelson in 1965, when he defined it as: “*a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper*” [Nelson 1965].

From the above quotation, Nelson has described the structure of hypertext as different materials connected to each other. The prefix “hyper” has been used to emphasise the inability of these materials to be connected in a linear manner. McKnight [McKnight et al. 1996] clarified that hyper means “more than”; therefore hypertext means “more than text”.

Smith and Weiss defined hypertext as: “*an approach to information management in which data is stored in a network of nodes connected by links*” [Smith and Weiss 1988]. Moreover, Hypermedia is a more general concept than hypertext, as the connected material could be text and other media [Conklin 1987].

Conklin indicated that reference books, dictionaries and encyclopaedia present an old form of hypertext: “*Another kind of manual hypertext is a reference book, exemplified by the dictionary and the encyclopaedia. In the sense that each of these*

can be viewed as a graph of textual nodes joined by referential links, they are very old forms of hypertext” [Conklin 1987].

Such kinds of documents “*are linear sequences of independent units*” [Smith and Weiss 1988]. The reader, through that kind of complex document, usually searches them to locate an article or a definition, and then starts reading that part sequentially. Furthermore, the reader is likely to pass by various cross-references, such as “see the article written by X”. To follow those references, the reader must find the appropriate volume, entry, and then the related article or definition. Therefore, that method of linking documents together is found through references or “see also” lists. “*Hypertext electronic documents*” [Smith and Weiss 1988] solved many problems of conventional paper documents, for example, from the flexibility perspective, a reader within seconds can follow a reference or find a definition or information without searching through entire volumes. From the variety aspect, conventional paper documents are limited to graphics and text, but electronic hypertext nodes, defined as a network of linked nodes, could be sounds, animation, video sequencing and so on [Smith and Weiss 1988].

The idea of applying the hypertext concept using machines could be traced back to Bush (1945), in his article “As We May Think”, when he described a machine called Memex: “*A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory*” [Bush 1945]. From the above quotation, Bush stressed that the essential part of Memex is the ability to join two items together. That machine proposed dry photograph and photocells technology at that time to do this linking.

The first serious attempts to get the Memex machine into reality came two decades after Bush’s description when Douglas Engelbart in 1968 conducted a demonstration for his Augment system at the Fall Joint computer Conference [Engelbart and English 1968]. In that system, he demonstrated how files are organized in a hierarchal form and linked together. Moreover, Engelbart demonstrated the mouse and the chord key set.

During Engelbart's development of his system, Ted Nelson was developing his hypertext system Xanadu [Nelson 1967, Nelson 1980]. Xanadu system acts as a repository publishing system; once a document is written it is not deleted. Therefore, the original document remains, except if there is another new version(s) of it; that would reference to the original one.

1.1.3 Why adaptive educational hypermedia?

Although hypertext has provided many advantages over conventional paper documents, it has disadvantages as well. Conklin [Conklin 1987] has pointed to two major problems in hypermedia/hypertext: disorientation - where a user could be easily lost in the hyperspace (information network) - and cognitive overload - where a user could find masses of information that may or may not suit him/her.

From an educational prospective, Chen and Ford [Chen and Ford 1997] stressed that the disorientation problem has a negative impact on education: *“disorientation prevents learners from getting to where they want to go; it slows down the access of meaningful and relevant material; it confuses the learners reducing their patience and diminishing motivation to learn”* [Chen and Ford 1997].

Furthermore, they pointed out that the cognitive overhead also has a negative effect upon learning, as students spend much of their time navigating through the presented links; thereby, they have very little time to read and think about the information. Moreover, Chen and Ford contemplated other problems such as a) lack of comprehension – as hypermedia representing information in the form of a network of interconnected concepts; there is some doubt for a user with a little experience with the system to understand the interconnections and construct a “coherent overview”; b) problems with access – novice users will find difficulty in constructing information about a subject they are not familiar with; and c) inefficient learning strategies.

Diana Laurillard in her book “Rethinking University Teaching” indicated that in hypertext *“...there is no intrinsic feedback on the user's actions: the information in the system does not change as a consequence of the user's actions on it; it only changes if they change the system, by changing the information or the link directly”* [Laurillard 1993, p 121].

From the above quotation, it can be understood that traditional/classical educational hypermedia systems, which in turn include non-adaptive web-based systems, are generally static, in that once written their content cannot be changed without external intervention, thus providing only a single learning experience to each learner, regardless of their needs and requirements. Hence, a web application that is designed with a particular class of users in mind may not suit others [Eklund et al. 1997].

Because of the disadvantages found in hypermedia systems, adaptive hypermedia has been developed. These kinds of systems can address those problems in the classical hypertext systems by altering the information presented to each user depending upon a defined set of characteristics, such as background, knowledge, etc [Brusilovsky 1996, Quentin-Baxter 1999]. Therefore, the problems of disorientation and cognitive overload could be solved. As an example of such systems is ELM-ART system [Brusilovsky et al. 1996b, Weber and Specht 1997b], which is considered one of the earliest web-based adaptive educational hypermedia systems (explained in detail in Chapter IV). In Chapter III various techniques of adaptation on two levels: contents level and links level are explained.

In order for adaptive systems to be able to adapt their contents and links, information about users' different traits such as knowledge, background, goals, etc. need to be modelled and represented. Therefore, these systems could use such information to provide their users with an appropriate kind of adaptation that suit their different characteristics. Hence, user modelling is a prerequisite for any adaptive system. In Chapter II, different techniques of user modeling are described.

Adaptive educational hypermedia systems are considered to be the most widespread application of adaptive hypermedia [Brusilovsky 2000]. This popularity is probably, in part at least, due to the disadvantages found in the traditional educational hypermedia systems.

Adaptive educational hypermedia combine ITS and hypermedia together, as it inherited from ITS the usage of the knowledge about students, domain and teaching strategies, and from adaptive hypermedia systems the adaptation of contents and links of the hyperdocuments to users according to different forms of user models

[Brusilovsky 2001]. There have been many adaptive educational hypermedia systems developed in recent years, such as AHA! [De Bra et al. 2002b], Interbook [Eklund et al. 1997], Metadoc [Boyle and Encarnacion 1994], etc., the characteristics of each of these systems and more being explained in detail in Chapter IV and their analysis in Chapter IX.

1.2 Problems Under Study

The main goal of adaptation is to deliver information to users in a way that suits their different backgrounds and goals. To achieve that, a suitable user-model needs to be developed.

By analysing the user model employed by many adaptive educational systems such as those described in Chapter IV, it is found that they maintain the knowledge value of each individual user, in addition to other characteristics, with respect to concepts that belong to a single topic. According to De Bra [De Bra et al. 1999], concepts could be divided into three kinds: atomic concepts (smallest information units), pages that are composed of fragments, and abstract concepts that represent larger units of information. In addition, these concepts are connected to each other through different concept relationships, such as prerequisite relationships.

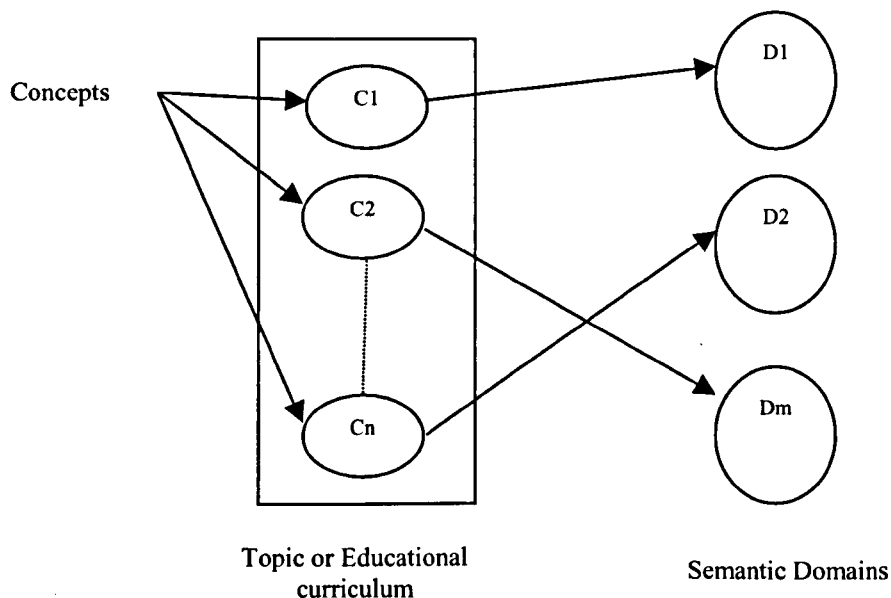


Figure 1.2 Domains and Topics - From this figure, it could be perceived that Topics or educational curricula are composed of different concepts. Moreover, these concepts belong to different semantic domains. For example, a concept might belong to the chemistry domain, and another one may belong to the biology domains. Hence, a topic or an educational curriculum may serve one or more semantic domain(s) according to the integrated concepts. C: Concept, D: Domain

One question that arises here is: ‘what if we want to have a framework that has the capability to run different topics or courses simultaneously for users with different educational levels, such as undergraduate students and postgraduate students?’ Furthermore, what if these topics/educational curricula have prerequisites that depend on semantic domains not on topics? For example, there may be a topic about the biochemistry domain and that topic may require certain knowledge in the biology and chemistry domains a user may acquire after learning different concepts from different educational curricula about each domain independently through different educational stages, e.g. in his/her first year and second year as an undergraduate. Thus, users who have that level of knowledge about biology and chemistry could understand the topic’s concepts, and thereby, they could reach the required level of knowledge from that topic with respect to the biochemistry domain. Therefore, to be able to manage users’ knowledge globally the presented concepts that are embedded in an educational curriculum should be mapped to comprehensive semantic domains and not to the material that presents them.

It is important to clarify that a topic or an educational curriculum refers to the teaching material to be taught and domains refer to semantic domains such as biology, chemistry, etc. Moreover, from Figure 1.2 it could be perceived that an educational curriculum/topic may contain concepts that belong to one or more domains, and as a result, it integrates one or more semantic domain(s) simultaneously.

During research about systems that considered that problem, a system called Metadoc [Boyle and Encarnacion 1994] is found, which is described in detail in Chapter IV. This system addresses these issues and solves them to a certain extent. Metadoc classifies users with respect to their domain specific knowledge into four classes: novice, beginner, intermediate, and expert. Likewise, AIX/Unix concepts and general computer concepts are classified into different concept levels using the same scale. Thus, users’ knowledge level about Unix/AIX and general computer concepts is independently stereotyped, i.e., a user may be a novice in Unix/AIX and a beginner in general computer concepts. The problem in the user model used in that system is that it cannot differentiate between users at different educational levels. For example, if there are two users and one is an undergraduate and another one is a

postgraduate, and both of them are intermediate in Unix/AIX domain. In reality, the intermediate level of the postgraduate may be very advanced with respect to the undergraduate one. Moreover, the learning style and rate of acquisition may be vary also

Therefore, the problem under study could be summarized as follows:

- How to create a user model that can be used with educational frameworks and how that model can manage users' knowledge levels globally (i.e. with respect to semantic domains) and not locally (i.e. with respect to topics). Furthermore, these topics may require users with a certain knowledge level in certain domain(s) to understand them. For example, if a user has a biology background and wants to study a topic about law, it may be required that that user has a certain knowledge level in the law basics domain. Furthermore, that domain, which is law basics, may include several topics.
- How to create a user model that can differentiate between its users that share the same knowledge classes. For example, if there are two users who share the same knowledge class, such as an intermediate in certain domains, and one of them is an undergraduate and the other is a postgraduate.

As a possible solution to these problems, a novel user model - known as the Hybrid Model (HM) has been developed [Zakaria et al. 2002, Zakaria and Brailsford 2002], and this is described in detail in Chapter VI.

1.3 Proposed Solution Overview

By observing the user modelling that is central to many adaptive educational hypermedia systems, in addition to the extensive research that has been done in the user modelling field, it is found that two measure techniques have been used: the overlay model [Carr and Goldstein 1977], and the stereotype model [Rich 1989]. Those two models will be explained in full detail in Chapter II. Briefly, the overlay model is a widely used technique, which is used to measure the knowledge level of students in certain topics. The knowledge level is represented in the form of a "Concept-Value" pair. The stereotype model classifies users into groups according to their knowledge, background or other selection criteria.

The Hybrid Model has been constructed using aspects of both these models, as follows:

- The Overlay model: in order to extend a user's knowledge from a topic's concept to domains, the classical definition of overlay as a "Concept-Value" pair needs to be modified to "Domain-Value" pair [Zakaria and Brailsford 2002], where domain means semantic domains such as chemistry or biology. Thus, educational materials may involve one or more domain, such as biochemistry, which may combine between the biology and chemistry domains. Based on that, users' knowledge levels are examined with respect to the involved domains through quizzes in addition to any other parameters that systems' authors may see convenient, such as the time each student spent in studying a certain topic. In Chapter VI, that modification in the overlay model and how it is carried out is explained in much more depth.
- The Stereotype model: in the Hybrid Model, the stereotype technique mainly depends on the knowledge level of users with respect to the involved domains. Moreover, classes in the stereotype are concerned with fixing certain weak points the students might have. Each class defines an article or set of articles, a link/links to external document/documents, or to lesson/lessons in other course/courses. If a student belongs to one of the advanced classes, he/she will be provided with advanced articles or links to help him/her find out more about the topic. The number of classes will be defined according to the authors of the system, as they can assign only three classes, as beginner, intermediate or advanced, or more. In addition to those classes, there are other classes that are concerned with the educational level of involved users. For example, a class for undergraduates, another one for postgraduates and so on. The full details about the number of stereotype sets used in the Hybrid Model and about their functions are described in Chapter VI.
- In addition to the overlay model and the stereotype model, the Hybrid Model embraces what is called an information pool, which holds educational materials to be adapted according to users' category and knowledge level.
- Each time a student passes from one lesson/topic to another his/her knowledge level will be updated according to the score the user get in the system assessment, in addition to any other parameters due to the policy of the system's authors.

According to the student's new knowledge level, his/her class might be changed to another stereotype class or to remain the same in case the knowledge level is not changed.

In fact, the overlay model and stereotype model together provide the main components used to create the Hybrid Model. Furthermore, the novelty comes in the way of modifying the overlay model and in mixing the two models together in a unique and genuine method to serve adaptive educational frameworks. In Chapter VI, the skeleton of the Hybrid Model is explained in detail.

In order to test this model, it was implemented using an experimental Integrated Learning Environment called WHURLE (Web-based Hierarchical Universal Reactive Learning Environment) [Brailsford et al. 2001, Brailsford et al. 2002]. This version of WHURLE is referred to in the rest of the thesis as WHURLE-HM¹ (WHURLE – Hybrid Model).

The WHURLE system was initially derived from an earlier system called the "Scholar's Desktop" (SD) that was developed by the TLTP (Teaching and Learning Technology Program) Biodiversity Consortium in the mid-1990s. This system is an Integrated Learning Environment, designed to deliver hypermedia courseware. The content for the SD was structured as a "Study Unit" consisting of a number of Nodes, each of which consisted of a single root/main page associated with a number of child-nods. Each study unit represented an interactive learning resource that was designed to promote specific learning objectives through the self-paced, interactive engagement with tasks, information, problems, or all three.

WHURLE is the next generation of SD, which implements features that worked well with SD along with a change in the pedagogy. Full details about the WHURLE structure are explained in Chapter VI under section 3.1. Furthermore, in Chapter VI and Chapter VII, the full details about how the system and the model are integrated together without any change in the main design of the WHURLE system are described. Also, in the implementation chapter (Chapter VII), a flow chart of the implementation algorithm is given.

¹ WHURLE-HM is called "Adaptive WHURLE" in [Zakaria and Brailsford 2002], which is available in Appendix F

1.4 Research Motivation and Objectives

The main motivation behind this research is to enable the creation of an adaptive educational hypermedia framework, where the users' knowledge should not be limited around topics' concepts, but be extended to semantic domains - as each domain may be involved through different topics that help to build users' knowledge about that domain. Moreover, multiple domains should be involved and integrated together in a single domain model, and there must be no limit to the number of involved domains. Starting from that motivation, the objectives can be summarized into:

- Creating a user model that mixes between two well-known and widely used techniques in a novel manner.
- Building a user model that has the capability to track students' performance, manage users' knowledge level with respect to the studied domains not topics, point out students' weak areas and provide them with the right kind of information that could fix these areas, and this all depends on the topics'/courses' authors.
- Testing how the Hybrid Model is flexible, by adapting it to integrate with the WHURLE system without any deviation from its main goals. In addition, applying different adaptation ideas to the resulting framework, such as those described in Chapter III.
- Testing the efficiency of the model by testing it with real students using real educational material

1.5 Thesis Structure

This thesis is divided into four parts:

PART I: Literature Review

Chapter I- Introduction: this chapter gives an introduction to the problem for which a solution is being sought, in addition to the motivations and objectives of that research.

Chapter II- User modelling and user modelling systems: a review about the user modelling techniques and systems

Chapter III- Adaptation in hypermedia: a review about various methods of adaptation used in different systems.

PART II: Technology Review

Chapter IV- Survey: describing different well-known systems from two angles: the user modelling angle and the adaptation techniques angle.

Chapter V- Technology: a review about various existing technologies such as XML, JAVA, Publishing frameworks, etc.

PART III: The Contribution

Chapter VI- The Hybrid Model: this chapter is considered to be the heart of the thesis, as it describes the Hybrid Model and how it is conceptually implemented inside WHURLE. Furthermore, this chapter has been published in the AH2002 conference proceedings and in the New Review of Hypermedia and Multimedia Journal (NRHM'02, Volume 8) - both of them are available in Appendix F.

Chapter VII- Implementation: through this chapter, the technical mechanism of the integration between the Hybrid Model and WHURLE is explained. In addition, the flow chart of the implementation is provided. Moreover, the explained system architecture in that chapter has been published in the Hypertext conference 2003 as a short paper, which is available in Appendix F.

PART IV: User Trial and Discussion

Chapter VIII: User Trial: experimenting the model through the WHURLE-HM system with real students using real teaching material is described here. The main goal of this trial is to prove the applicability of the Hybrid model to adaptive educational hypermedia systems and it is possible to be used in educational setting.

Chapter IX: Discussion: this chapter explains why the Hybrid Model is unique and what is the main difference between it and other used models in different systems. Moreover, an explanation is given as to how that system could be extended. This is followed by the limitations, further research and conclusion.

Chapter II: User Modelling and User Modelling Systems

2.1 Introduction

For adaptive hypermedia systems to be able to adapt their contents and links to their users, information about users' knowledge, background, goals, traits, etc. are needed to be modelled and represented through different techniques as will be described later in the chapter. Therefore, these systems could use this information to provide appropriate personalization for every individual user. Thus, user modelling is an important aspect of human-machine systems and without it any personalization in hypermedia systems will not be feasible.

According to Brusilovsky, adaptive hypermedia systems are:

"...the systems which can provide automatic adaptation on the basis of the user model" [Brusilovsky 1996].

Moreover, he distinguished three stages in the adaptation process, as in Figure 2.1: collecting data about the user, processing the data to build or update the user model, and then applying the user model to provide adaptation.

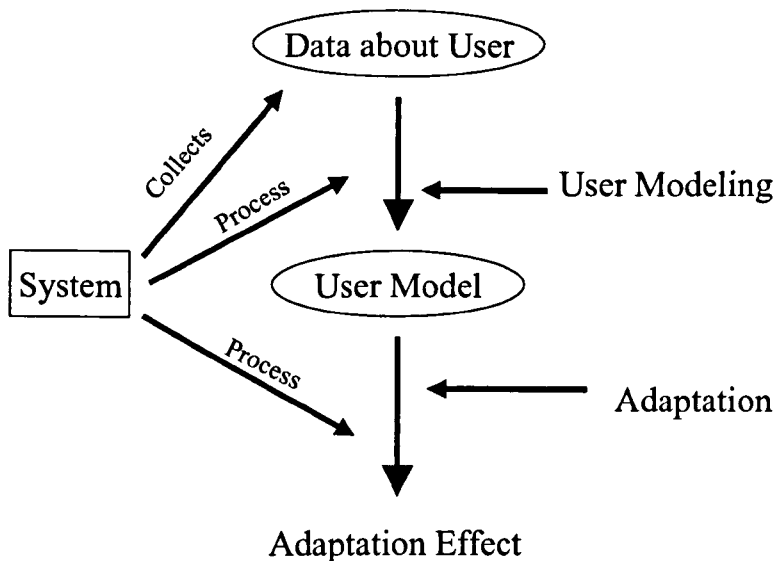


Figure 2.1 Adaptation Process - Stages of adaptation process in Adaptive Hypermedia [Brusilovsky 1996]

From the above figure, it could be perceived that the process of collecting data about users to construct a user model could be explicit (provided by the user) or implicit

(inferred by the system), and this will be explained later. As a result, adaptive hypermedia systems could perform adaptation based on this information for every user individually.

Kobsa [Kobsa et al. 2001] also has distinguished three tasks in the adaptation/“personalization” process: acquisition, representation and secondary inference, and production task. Furthermore, he has differentiated between data about user (user data), data about computer usage (usage data) and data about users’ environments (hardware and software). The term user data refers to information about personal characteristics of a user, while usage data refers to user’s interaction behaviour with the system.

There are common features that could be considered to construct a user model and related to the user data, and they are as follows: [Brusilovsky 1996, Brusilovsky 2001, Finin 1989, Kobsa et al. 2001]

- *Demographic data*: such data is related to the characteristics that identify users physically such as record data (name, address), geographical data (city, zip code), etc. Commercial applications, like e-commerce sites, rely on this feature to provide customers with support and promotion services. However, if consumers’/users’ data is not properly protected it could be subject to abuse (data privacy).
- *Knowledge*: Through this kind of information the adaptation could be individualized, as each user may see the information that suits his/her knowledge level without being perplexed if the presented information is much higher than his level, or bored because of the unnecessary explanations. Many adaptive hypermedia systems, especially educational ones, depend on this feature to provide users with the appropriate kind of personalization. such as those described in Chapter IV as CHEOPS [Ferrandino et al. 1997], AHA! [De Bra et al. 2002b], HYPADAPTER [Hohl et al. 1996] and others. From the researcher’s point of view, this feature plays an important and fundamental role in educational applications.
- *Skills and capabilities*: a user’s skills and capabilities can also play an important role in the adaptation of a system to a user’s needs. It is likely that a user knows how to perform a certain task but he/she cannot do it because of a physical

disability or any other obstacle. For example some systems that deal with tourist information, such as AVANTI [Fink et al. 1998], take into account the needs of disabled people and recommend actions that are within their capabilities. From the researcher's perspective, such a feature is very useful especially for systems that deal with users who have special needs.

- *Interests and preferences:* a user's interests and preferences are considered important features for many systems such as information retrieval (IR) systems, such as search engines, and recommender systems. Recommender systems recommend items to users such as products, services, news, etc. therefore, users can express their interest by rating the suggested items. Therefore, through these rating actions the system could adapt itself to the interest of each user with respect to the corresponding items. Moreover, a user's preferences, such as his/her preferred learning style or language, are used heavily in adaptive educational hypermedia systems such as those described in Chapter IV as TANGOW [Carro et al. 1999, Carro et al. 2000], Hypadapter [Hohl et al. 1996] and others. In WHURLE-HM this facility has been used so that students can choose the interface style they prefer.
- *Goal and plans:* a user's goal is the aim or the target the user wants to achieve, while a user's plan is a sequence of steps he/she takes to achieve that goal. However, each step in the plan may have its own sub-goal to realize. Furthermore, goals could be classified into either a high-level goal or a low-level goal. For example, in adaptive educational systems, a high-level goal is the learning of a subject, and a low-level goal may be solving a current problem. What is more, a user's goal may be directly specified by a user or deduced by the system. This feature has been applied through different adaptive educational hypermedia systems successfully such as Interbook [Brusilovsky et al. 1998a].
- *Background:* background refers to all the relevant information related to the user's previous experience outside the domain of the hypermedia system. This background may include a user's profession, education, work experience in a related area, etc. This feature has been utilised in Hybrid Model in the form of categories where users are categorised according to their educational status and background i.e. undergraduate, postgraduate, researcher, etc.
- *Experience:* experience refers to users' familiarity with the structure of the hyperspace. This has a positive effect on helping users to reach their goal swiftly

and obtain more in depth knowledge about the information the domain model is presenting.

Knowledge about users held by a user model differs from one application to another [Finin 1989]. For example, in e-commerce sites, a user’s interest in a certain product or a range of products is much more important than his/her knowledge level. On the other hand, in educational systems, users’ knowledge about concepts presented in the domain model is vital compared to users’ interest in a certain product. Furthermore, most of the presented features are common keys used in building a user model for adaptive educational hypermedia systems as described in Chapter IV. Nevertheless, the knowledge feature is the most essential aspect used in all user-adaptive web-based educational systems.

Throughout the rest of this chapter, different kinds of user modelling and presentation techniques are discussed. Moreover, a background describing generic user modelling systems is provided.

2.2 Types of User Modelling.

Rich [Rich 1999] has categorized the world of user modelling into three-dimensional space. These dimensions are:

- Standard user models and individual user models.
- Explicit (collaborative) models and implicit (automatic) models.
- Long-term models and short-term models.

2.2.1 Standard User Models and Individual User Models

Within that dimension, Rich has differentiated between two types of user model. One of them is a standard or “canonical” user model. This kind of user model is not useful for systems with heterogeneous users, as users are diverse in their traits such as interests, preferences, knowledge, background, etc. An example of such systems built using a standard model is WebCT [Piguet and Peraya 2000, Curtin 2002]. This is a hypermedia educational system originally developed in 1995 at the University of

British Columbia, and subsequently developed as a commercial product¹. It is an environment for integrating and delivering educational materials over the web, and for administering and supporting courses. The content model of WebCT presents a static and inflexible pedagogic experience, without any kind of adaptation at the user level. Hence web applications, such as those delivered via WebCT, tend to be designed with a particular class of users in mind, and may not suit those even marginally different from the original target [Eklund et al. 1997].

The other type is an individual user model. This kind of model enables systems to provide adaptation to each user in a way that suits his/her preference, interest, and knowledge, such as in CHEOPS [Ferrandino et al. 1997], AHA! [De Bra et al. 2002b] and NetCoach [Weber et al. 2001] and others described in Chapter IV. Those systems are adaptive educational hypermedia systems, where their users are diverse in their goals and background. Therefore, an individual user model for each user or learner is essential in such a wide-ranging population.

2.2.2 Automatic (Implicit) Models and Collaborative (Explicit) Models

In this dimension, two different approaches of user modelling are explored. The first one is called *automatic (implicit) user modelling*, where the system deduces a user's goal, knowledge, etc. by observing the user's behaviour while navigating the hyperspace and processes this data to build a user model. The second one is called *collaborative (explicit) user modelling*, where users have a role in the user modelling process.

2.2.2.1 Automatic (implicit) user modelling

In the previous section, three steps in the adaptation process were described. In classic or fully adaptive systems, the first two stages, collecting data and processing it to build the user model, are automatically performed by the system in addition to the adaptation process. Thus, the process of user modelling is totally implicit and the user has no direct role in the process, which is why it is called automatic user modelling.

¹ <http://www.webct.com>

Fully adaptive systems watch users' behaviour and through this monitoring, these systems could deduce different traits about each individual user such as: [Kobsa et al. 2001]

- *Interest*: this characteristic could be deduced through different actions a user is performing such as:
 - *Selection actions*: a user may select a product or an article he/she is interested at. Many systems use a user's action of selecting as an indication for interest. For example, in Amazon.com if a user chose to read an abstract of a book or navigate the table of contents the system suggests others books related to the same subject. However, this action does not give a strong indication of whether the user is interested in a certain article/product or not. For example, the user may choose a link because of curiosity. On the other hand, although this feature is not accurate, it could provide help to systems that do not have much information about their users.
 - *Viewing time*: viewing time is difficult to be used as a positive sign to deduce users' interest in the presented hyperdocument. This is because people vary in their speed of reading; also it is impossible to guarantee the time spent by the user in front of the computer screen looking at a specific item. Therefore, the time that a user spent viewing an article/item could be used as negative indication. For example, if a user spent less than a specific time in a certain page, this may indicate that the user is not interested in this page or familiar with its contents. In case of hypermedia streamed objects (video or audio objects), if a user terminated the streaming, this gives strong evidence that the user is not interested, but if he/she waited until the end of the presentation this may give a positive sign about the user interest. From the researcher's perspective, indicating user's interest through this action is more accurate than selecting objects.
 - *Evaluating*: evaluating an item could happen through rating this item. A user's rating for an object could indicate explicitly how this item suits his/her interest such as in the case in the previous

action (Viewing time). Therefore, this behaviour gives a strong indication for users' interests.

- *Purchasing actions*: in e-commerce sites such as Amazon.com, purchases made by users are regarded as strong evidence of the user's interest; therefore the systems react adaptively by suggesting related or similar products. Purchasing may not be an indication of a user's interest if this product were a gift to other people from the user. Amazon.com tried to solve that problem by discarding purchase orders with a shipment address different from that of the user, which is inadequate solution. For example, if a user in one place whose address is registered by the system ordered a product to be delivered to a holiday address in a different name, the system would assume that the product is a gift and therefore it does not fit in the preferences of that particular user. The researcher proposes that if the user can imply explicitly that such a purchase is a gift, such a problem could be solved more efficiently and the user's interest could be indicated more precisely.
- *Further processing actions*: if a user accessed certain document or visited certain web site and then he/she did another actions such as printing/saving/forwarding such an article (such as in my.yahoo.com) or bookmarking such a site, this behaviour could give an indication about user's interest in this article or web site. However, this behaviour cannot give a strong indication if a user is interested in such site or article, because he/she may want to forward/print/save such article/site for someone else such as a colleague, a friend, etc.
- *Unfamiliarity*: from a user's behaviour it could be inferred that he/she is not familiar with information/product/service a system is providing. For example, the MetaDoc [Boyle and Encarnacion 1994] system, which an adaptive educational hypermedia system, allows its users to get more explanation about the technical expressions they are not familiar with by means of the stretch text technology. The MetaDoc system uses this action as an indication of the unfamiliarity of the

user with that expression and consequently his/her user model is updated. However, such behaviour cannot be relied on as a sign of unfamiliarity with certain expressions, as a user may choose an explanatory link just for curiosity or to assure his/her information. Metadoc system handled this problem by considering a user as unfamiliar with an expression/concept if he/she chose to expand the hot word that provides an explanation to that expression/concept or jumped to the glossary more than once. From the researcher's perspective, this is an adequate solution to confirm a user's unfamiliarity. Moreover, viewing time action explained in the previous point could be used as an indication of a user's unfamiliarity with the material the system is presenting. For example, if a user waited till the end of a presentation then this action could indicate he is not familiar with the presented material.

- *Preferences*: selecting actions could be used to deduce users' preferences. For example, by presenting different objects to a user to select from, his/her selection may help the system to infer the user's preferences with respect to the selected object type. However, this feature could also be difficult to be precisely inferred by the system because a user might choose an object out of curiosity at that particular time. Therefore, it is preferable to be explicitly specified by the user.
- *Knowledge*: in adaptive educational hypermedia systems, such as those explained in Chapter IV, users' knowledge could be deduced by a system through monitoring their behaviour while navigating the presented material or through quizzes and tests.

Brusilovsky [Brusilovsky 1996] argues that systems that perform adaptation without a user's influence (automatic user modelling) cannot be completely relied upon. These systems are likely to deduce a mistaken user model and consequently provide mistaken adaptation. The researcher finds that this argument is a valid one, and therefore, automatic user modelling should be used with caution. Therefore, systems that implement such a kind of user modelling must have a way to be positive about the facts they collect to build the user model, in addition to a way to resolve conflicts between gathered facts about users [Rich 1999].

2.2.2.2 Collaborative (explicit) user models

In collaborative or cooperative user modelling [Barker et al. 2002, Kay 1995, Rich 1999], the user has a direct role in the user modelling process. Brusilovsky [Brusilovsky 1996] has suggested how to involve users in the process of user modelling, as in Figure 2.2.

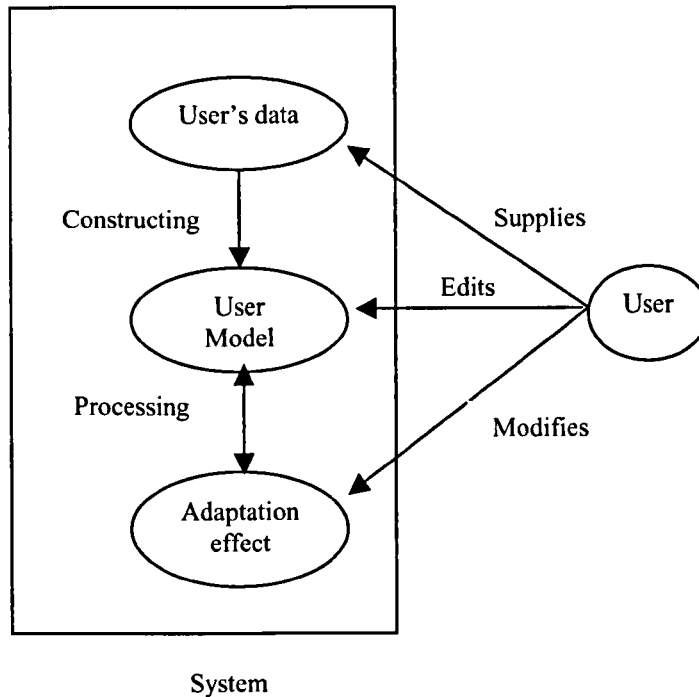


Figure 2.2 Collaborative user modelling - involves users in the user modelling process, where they have a role in each involved step. Modified from [Brusilovsky 1996]

From the above diagram, it is found that:

- Users can provide the user modelling with the required data such as specifying if a certain article is relevant to them through ratings.
- Users could change the amount of the presented information. For example, in Metadoc system [Boyle and Encarnacion 1994] users could change the detail of the presented information through the stretchtext technique. Therefore, they change the adaptation effect.
- Users could edit their user model. In many adaptive systems, this feature is used to get information about a user's background such as his/her profession and experience, which is difficult for the system to infer, such as in TANGOW [Carro et al. 2000]. In addition to this information, users can set other influential data like their current goal. To achieve that purpose, systems could provide a special interface for their users in order to perform these tasks. Moreover, such systems

that allow their users to perform changes in their user model are called Adaptable systems. However, some information in the user model is very critical and any change in it could end up with an incorrect adaptation. Hence, this data needs to be accessible only by an experienced user like the system administrator to set the model correctly. From the researcher's perspective, in case of adaptive educational systems, the only exception could be given to systems that hold pre-assumptions about the knowledge of their users. Therefore, if they (users) find any of the presented concepts they are familiar with, they can inform the system to update their model and future adaptation could be carried in the light of these changes such as in Hypadapter [Hohl et al. 1996].

2.2.3 Long-term Models and Short-term Models

For a system to interact reasonably with a user, the system has to access a wide variety of information about that user. This information ranges from short-term facts like the subject of the article that a user has just finished reading, to long-term facts like his/her experience/knowledge level in solving a particular problem.

Short-term models describe a user's specific goals and tasks in the current interaction with the system. Such a kind of user modelling is widely implemented in e-commerce applications. Long-term models express stable characteristics of a user such as interest or expertise, which are derived from a series of interactions between the system and the user [Rich 1989, Rich 1999].

From the above sections, it could be perceived that implicit user modelling is not dependable as it presumes assumptions about users, which are subject to different explanations, while they perform actions or navigate the hyperspace. However, this kind of modelling could be useful for systems that use short-term models, nevertheless, based on the fact that those system utilise different methods and techniques to reassure consistency of the assembled beliefs about their users. On the other hand, explicit user modelling shares users in collecting data from them. Therefore, it has a small room for error. However, explicit user modelling methods is not very practical for applications with which users interact for a short period of time, such as e-commerce applications, as users do not have much time to fill out forms. In different contexts, such as education where long-term models are used,

users react with the system many times and they could have time to feed the system with their traits explicitly. Furthermore, in educational applications, integration between explicit and implicit user modelling methods is useful as in the case of WHURLE-HM [Zakaria and Brailsford 2002], NetCoach [Weber et al. 2001], TANGOW [Carro et al. 2000], and others explained in Chapter IV. With this kind of combination, users could supply a system with their traits that are difficult to be inferred by the system, such as their preferences, background, etc., and the system could deduce their knowledge about the presented curriculum through different methods such as quizzes, tests, etc.

2.3 Representation Techniques

In general, a student model and a user model are quite similar, but they have characteristic differences in the way they are generally viewed. A user model is a model an adaptive system keeps for a user who is currently using it, while a student model is a model for an individual the ITS views as a student [Kass 1989]. Moreover, a student model considers how a student represents or reasons the knowledge he/she has. On the other hand, a user model is concerned with modeling a user's goals, plans and beliefs. Thus, a user model focuses on what a user wants or believes, and sometimes on how he/she plans to achieve his/her goal rather than modeling how a user's belief might change or why a user holds a certain belief [Kass 1989].

In the following subsections the presentation techniques will be divided into two categories: knowledge presentation techniques, and stereotyping. The reason for that classification is that stereotyping could be used for different purposes rather than representing a learner's knowledge, which is why it is preferable to classify it separately.

2.3.1 Knowledge Representation Techniques

Adaptive educational hypermedia systems are derived from Intelligent Tutoring Systems (ITS) and adaptive hypermedia systems. ITS use the knowledge about domain, student, and teaching strategies to provide each user with an individualized learning experience. On the other hand, adaptive hypermedia systems apply various

user models in order to adapt their contents and links to a user's needs, requirements and knowledge [Brusilovsky 1998].

In the following subsections, an overview will be given about some common knowledge representation techniques used by ITS and adaptive educational hypermedia systems.

2.3.1.1 The overlay model

The overlay model was developed in MIT (Massachusetts Institute of Technology) as a part of the COACH project [Carr and Goldstein 1977], whose aim was to develop an Artificial intelligence based Computer-aided Instruction (ICAI) program for tutoring the abilities needed for successfully playing various computer games. The idea of an overlay model is to represent a user's knowledge about any subject as an overlay of the domain model. In other words, an overlay model presumes that a user's knowledge is a subset of the domain knowledge, as shown in Figure 2.3. Thus, for each concept in the domain model there is an individual overlay model that measures how much a user knows about that concept. This measurement could be a binary value (known – not known), or a qualitative or quantitative measure [Brusilovsky 1996]. An overlay model of user knowledge could be represented in the form of pairs (Concept – value) for each concept in the domain model. Overlay modelling is easy to implement, as it does not require any information outside the domain model content.

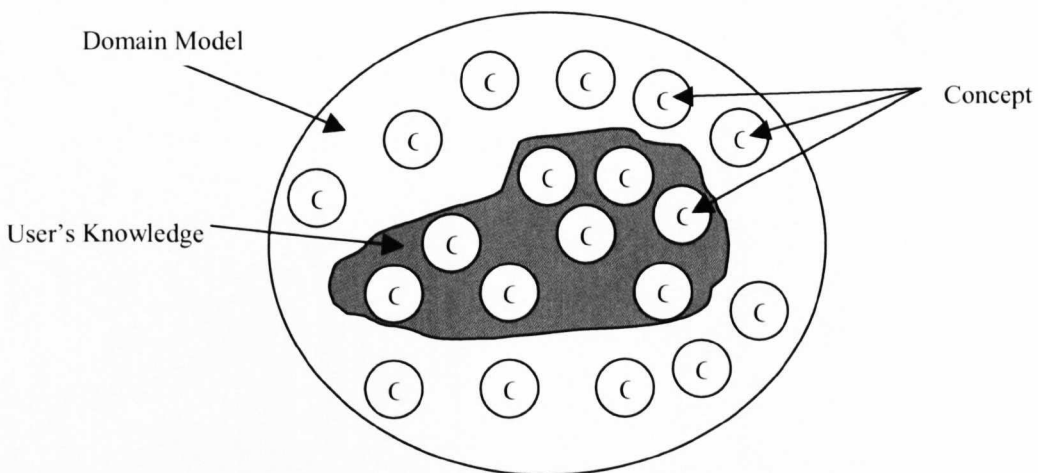


Figure 2.3 Overlay Model - user's knowledge in the overlay model is a subset of the domain knowledge. C: Concept

Kass [Kass 1989] argues that the overlay model has a fundamental drawback, in that it cannot deal with the knowledge a user has if it is different from the knowledge presented in the domain model. From the researcher's perspective, this point cannot be seen as a drawback because this is the nature of overlay models, as they measure a student's knowledge with respect to the concepts represented within the domain model. Moreover, the Overlay model is widely used among adaptive educational hypermedia systems such as those described in Chapter IV.

2.3.1.2 The differential model

The differential model is a different form of overlay model [Kass 1989]. However, instead of representing the user/student knowledge as an overlay of the domain model, as in the case of the overlay model, the differential model divides a learner's knowledge into two categories: knowledge that should be known by the user (such as prerequisite concepts), which is presented in the domain knowledge, and knowledge the learner is not expected to know [Kass 1989]. Figure 2.4 illustrates the differential model. In adaptive educational hypermedia systems, this concept is widely used, as a curriculum's authors expect students to be familiar with certain concepts (studied before) upon which the new concepts presented in the educational material are based.

Kass argues that the differential model suffers from the same problem that the overlay model suffers from, which is the inability to handle user knowledge if it differs from that presented in the domain model.

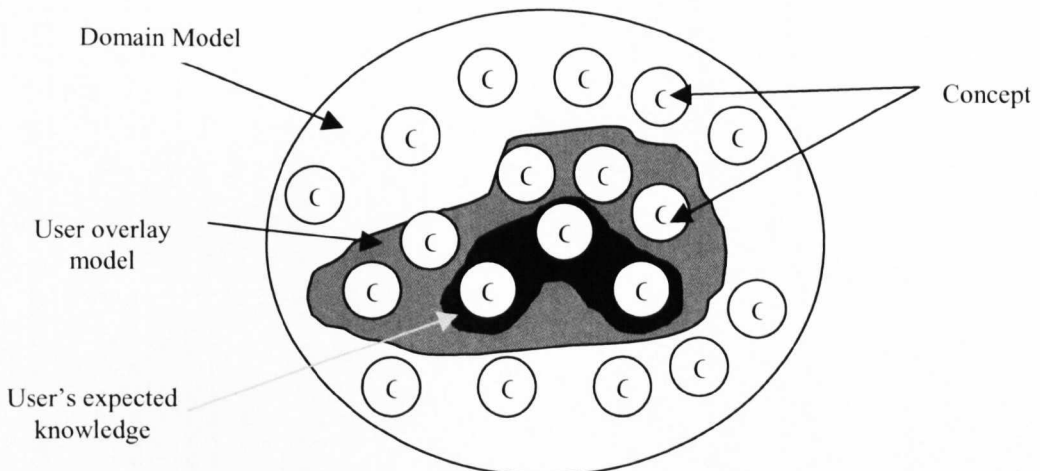


Figure 2.4 Differential Model - the black area shows the knowledge expected to be known by the user. C: Concept

2.3.1.3 Perturbation and Bug Models

Perturbation modelling does not represent learners in terms of correct knowledge only, as in case of overlay model, but also represents their faulty knowledge - as shown in Figure 2.5. Thus, it combines overlay modelling with a representation of faulty knowledge [Holt et al. 1991]. The most common technique used to represent the perturbation model, is to set the domain knowledge, and then extend that knowledge with the expected mistaken conceptions a user might have.

A bug library or a bug catalogue is a fixed collection of misconceptions that might be held by learners. As the learner progresses, the perturbation model can be updated with respect to the existence of known bugs in the bug library. Several approaches have been found for the development and the representation of bug libraries. One approach is to list or enumerate all the bugs based on observed analysis of learners' mistakes/errors – enumerative theories of bugs. Another approach is to generate bugs based on learners' misconceptions – generative theories of bugs [Holt et al. 1991].

A bug library is useful in recognizing causes of errors and misconceptions. A perturbation model adds an interpretation to a learner's misconceptions; also, it uses the bug library to define a range of likely misapprehensions in a student's learning at some point. For example, a student is adding two numbers but one is positive and the other is negative and he/she did it wrong. According to the bug library, the cause for this misconception may be due to the reason that the user is not aware about

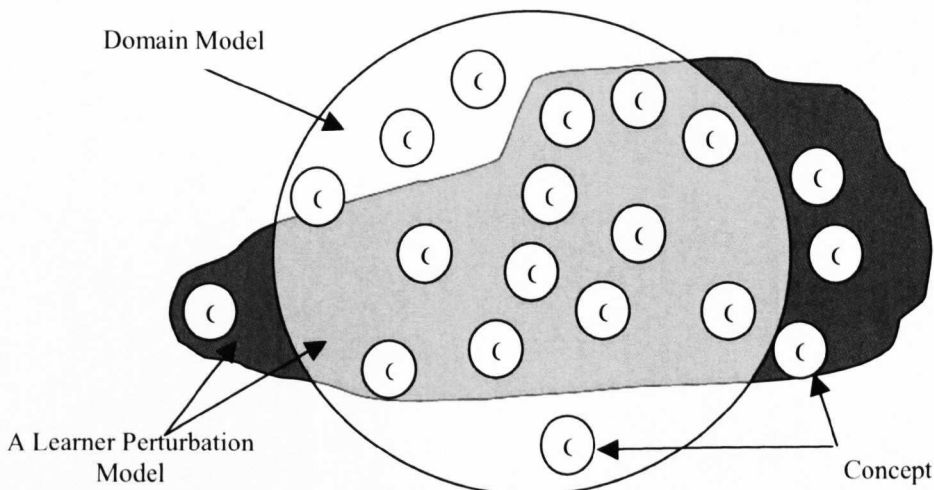


Figure 2.5 Perturbation and Bug Models – the dark grey area represents the misconceptions the user has about certain concepts, while the light grey one is the knowledge the user has and which is a subset of the knowledge represented through the domain model. C: Concept

positive and negative numbers, or may be he/she needs to know more about subtraction.

Kass [Kass 1989] claims that this model is a very reasonable method to model learners/users' knowledge. However, the implementation of such a model requires many empirical studies to obtain misconceptions that learners may have. Different ITS used such model. On the other hand, it is not commonly used among web-based educational hypermedia systems.

2.3.1.4 Constraint-based model

The constraint-based model represents learners' knowledge as constraints upon the correct knowledge representation [Holt et al. 1991, Ohlsson 1994]. That model extends the overlay approach by allowing more sophisticated reasoning about domain concepts further than whether they are known or not. A violation of those constraints indicates incorrect or incomplete knowledge of the student(s), shown in Figure 2.6. This kind of information could be used by ITS to guide student(s) in the learning process.

For example, a student has a problem of arithmetic that he/she is trying to solve, which is the addition of X/Y with Z/F , and there is a constraint that states if there are two fractions with different denominators, then these denominators have to be equalized before the fractions are added. If the student added the numerators before equalizing the denominators, this means that he/she violated the constraint,

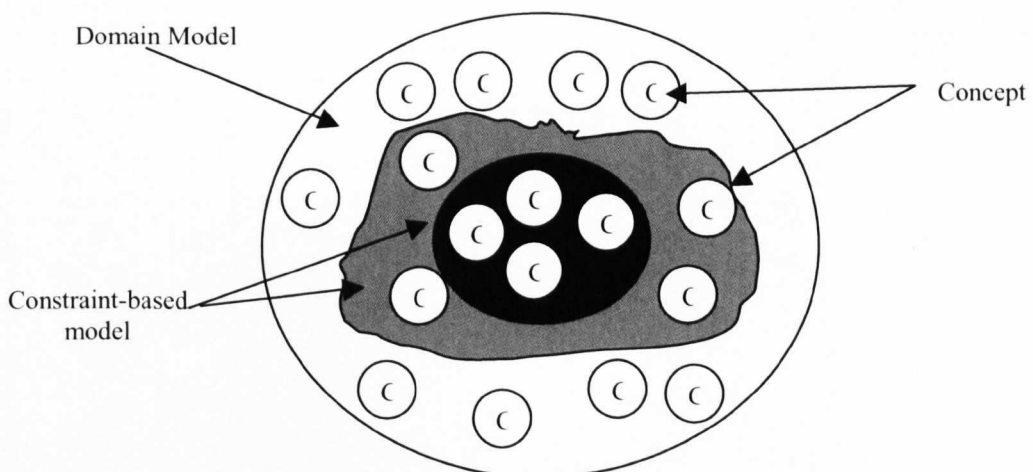


Figure 2.6 Constraint-based Model – the grey area represents the correct knowledge about concepts (subset of those presented in the domain model) a user has since he/she did not violate the associated constraints. On the other hand, the dark (black) area represents the user's misconceptions about some concepts because he/she violated the associated constraints.

and this action implies that he/she lacks the knowledge about fraction additions and needs to know about this type of addition.

From the researcher's perspective, the constraint-based model is close to the perturbation and bug models. However, the difference is that the constraint-based model refers users' misconceptions about the presented concepts to the violation of the associated constraints defined within the domain model and not to a bug library as in the perturbation model. The constraint model offers advantages over the perturbation mode. This model does not need wide empirical studies such as in the perturbation mode, as misconceptions about concepts presented in a problem could be directly compared to the constraints the student violated. Moreover, this model could identify user's creative answers as long as they do not violate any of the presented constraints [Ohlsson 1994]. Even with these advantages, this model has drawbacks as well. For example, any presented problem should be analysed efficiently to identify the constraints that if violated indicate which concepts are misinterpreted by the student [Ohlsson 1994]. Therefore, if the presented problem is not analysed enough to obtain the right constraints that can indicate users' misconceptions about which concepts, this model will not be effective.

2.3.2 Stereotyping

Stereotyping is a widely used technique in the world of user modelling. The idea behind stereotyping is to infer information about users through one or more observable event(s), trait(s) or action(s).

SPORTS-PERSON		
Triggers (used-description "athletic")		
Attribute	Value	Rating
Motivation	Excitement	600
Character-strengths	- Physical-strength	900
	- Perseverance	600
Interests	Sports	800
Thrill	5	700
Tolerate-violence	4	600
Generalizations		
ANY-PERSON		

Figure 2.7 a stereotype used in Grundy system [Rich 1989]

The stereotype represents a collection of traits that could be presented as a set of “attribute-value” pairs [Rich 1999]. For example, an attribute could be a user’s knowledge level, such as expertise in a certain topic and the value could be a scale value between 1 and 5.

Rich [Rich 1989] stated that a stereotype is a “knowledge structure” and composed of:

- Body: holds information that is true for all users who belong to the stereotype that hold this information.
- Triggers: a trigger is a condition that a user met, therefore all the information in the stereotype body is true for that particular user

Finin [Finin 1989] identifies stereotypes as a collection of both facts and rules and a way for classifying people according to the common traits they share.

The structure of knowledge that is contained in a set of stereotypes may vary. Rich has viewed the knowledge about users contained in these structures as a set of assertions regardless the form of those assertions. For example, Figure 2.7 shows a stereotype used in the Grundy system [Rich 1989], which is used to suggest novels to potential readers. Grundy uses two collections of data: one of them is an individual description for each book, each description being a set of facets filled with value, and the other contains facts related to readers’ taste in books.

The body of the SPORTS-PERSON stereotype (Figure 2.7), which is used within Grundy system, contains a set of attributes/facets associated with values. All values have an associated confidence measure called a rating. The rating is used to show the system’s confidence in each facet’s value before allowing it to influence the performance of the systems.

This stereotype is triggered when a user uses the term “athletic” in the self-description (Trigger). The generalization relationship relates this stereotype to another stereotype called ANY-PERSON. Thus, the class of people described by the SPORTS-PERSON stereotype is a subclass of the class of people described by ANY-PERSON.

Moreover, the stereotype model could be presented in a form of stereotype-value pairs, where the value could be Boolean (true or false), i.e. the value determines if the user belongs to that stereotype or not, or the value could be probabilistic, i.e. the value represents the probability that the user belongs to that stereotype.

Brusilovsky [Brusilovsky 1996] sees a problem with the stereotype model of knowledge, as adaptation techniques need a more detailed model like the overlay model. To overcome this problem, Brusilovsky suggested mapping from the stereotype to the overlay model by associating a fixed set of concept-value pairs with each stereotype. For example, Metadoc uses two knowledge classifications and two groups of stereotypes (novice, beginner, intermediate, expert); each group for each classification. One classification describes a user's knowledge about general computer concepts and the other one represents a user's knowledge about UNIX. Hence, a user may belong to the intermediate stereotype in a computer's general concepts and to the novice stereotype in UNIX. The combination between stereotype modelling and overlay modelling has shown good results [Brusilovsky 1996]. That combination could be achieved by applying stereotype modelling to classify new users to initiate values for the overlay model and subsequently a regular overlay model is used. It is important to notice that such mapping between overlay model and stereotype model is used in many different adaptive educational hypermedia systems such as WHURLE-HM [Zakaria and Brailsford 2002], CHEOPS [Ferrandino et al. 1997, Negro et al. 1998], and others described in Chapter IV, which proves that it is widely accepted.

2.4 Generic User Modelling Systems

In early adaptive systems, there was no apparent difference between systems' components that operate user-model services and other components that perform other operations [Kobsa 2001]. This kind of separation has been increasingly made since the mid-1980s onwards which led to the development of generic user modelling systems that are not dedicated to any specific application domain. In 1986, Finin introduced his GUMS 'General User Modelling System [Finin 1989], which is considered to be the earliest generic user modelling system [Kobsa 2001]. Moreover, that system sets the context for the basic functionality of generic user modelling

systems, as it provides selected user model services at run time that could be configured during the development time. Generic user modelling systems serve as a separate user modelling component in the relevant applications, as developers fill these systems with the user modelling knowledge that suit the application domain. In the following subsection, a brief overview about GUMS will be given.

2.4.1 GUMS

The General User Modelling System (GUMS) is a domain independent user-modelling tool that is designed for building long-term models [Rich 1999, Rich 1989] of individual users [Finin 1989]. That system allows for building models of individuals that are composed of a collection of facts that could be maintained. Users in GUMS are assigned to one or more stereotypes, from which additional facts are inferred using inference rules, which are associated with the stereotypes a user belongs to. The contents of the user model are left to the application developers to decide, as each application has its own needs. Any application using GUMS is responsible for initiating stereotypes for its users, acquiring facts about them (explicitly or implicitly) and providing services for model maintenance. The GUMS system does not interact directly with the user, but it interacts with the application and learns everything about the users through the application itself, as shown in Figure 2.8.

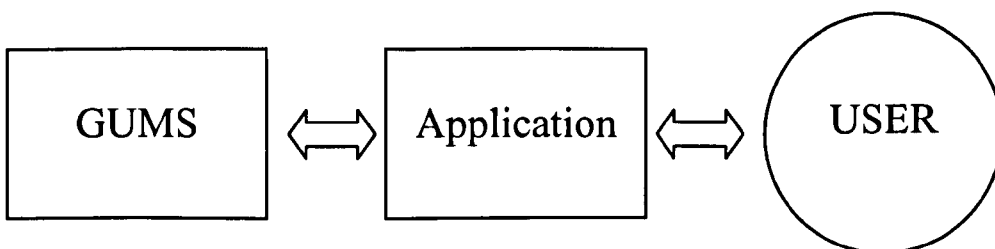


Figure 2.8 GUMS Interaction - GUMS interacts with the application and the application interacts with the user. Hence, everything the GUMS learns about the user comes through the application.

This user-modelling system has a separate knowledge base for each application it serves. Each knowledge base is composed of two parts:

- A collection of stereotypes
- A collection of individual models installed in the stereotype hierarchy

One service that GUMS offers the application is the ability to store new information about users, which the application has acquired through its interaction with those users. This information may trigger one of the following:

- Detection of any inconsistency in the user model and informing the application
- The user model may infer a new fact about the user
- The user model updates some previously deduced facts about the user.

Default reasoning of some kind or another is at the heart of all user modelling systems, and that is because in some situations the system wants to draw some reasonable and believable conclusions about a user from a small base of distinct knowledge about him/her [Finin 1989]. GUMS used three forms of default reasoning techniques: stereotypes, explicit default rules and failure as negation. In the following subsections, the role of each of them within the GUMS will be explained.

2.4.1.1 Stereotypes

In GUMS stereotypes are used to capture general traits among large classes of users. Stereotypes are composed of facts and rules that are believed to be true to all users who belong to them. Those stereotypes are organized in a hierarchal form, as shown in Figure 2.9. Within that hierarchy, a stereotype could include another stereotype(s) if it is believed to be more general. For example, postgraduate students stereotype subsumes both the research students stereotype and the master students stereotype, because postgraduate students is a more general stereotype. Moreover, everything that is true about postgraduate students is necessarily true for research students and master students, i.e. a stereotype inherits from its direct parent that in turn inherits from its direct parent also.

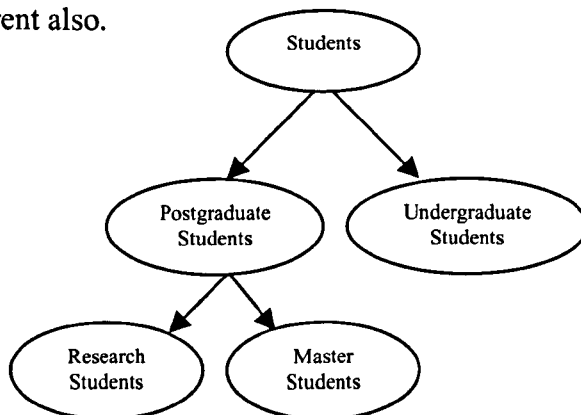


Figure 2.9 GUMS Stereotypes – Postgraduate Students stereotype includes under it research students stereotype and master students stereotype. Therefore, postgraduate students stereotype is more general than the research students and the master students stereotypes.

Each stereotype in GUMS has two types of facts and rules, one is definite and the other is default. The definite facts and rules determine what is true for anyone who belongs to that stereotype. If the knowledge about a user contradicts this information, then the user does not belong to that stereotype or any of its descendant stereotypes. On the other hand, default facts and rules set the initial beliefs about any user who belongs to that stereotype. These initial beliefs are changeable as long as new information about the user could be obtained through his/her interaction with the application. Positive facts about a user override default facts that are known or inferred from the stereotype to which the user belongs. These certain facts must be consistent with definite facts as, if there is any contradiction between these facts and the definite ones, then the user must be reclassified to another stereotype.

The strategy that GUMS follows to reclassify a user to a new stereotype is to search the ancestors of the current stereotype in order of specificity until it finds one in which there are no contradictions. In addition to the default and definite knowledge, stereotypes in GUMS have a third component called meta-level knowledge. That third component is a collection of meta-knowledge about predicates used in the definite and default knowledge base. Figure 2.10 shows the components of any stereotype in GUMS.

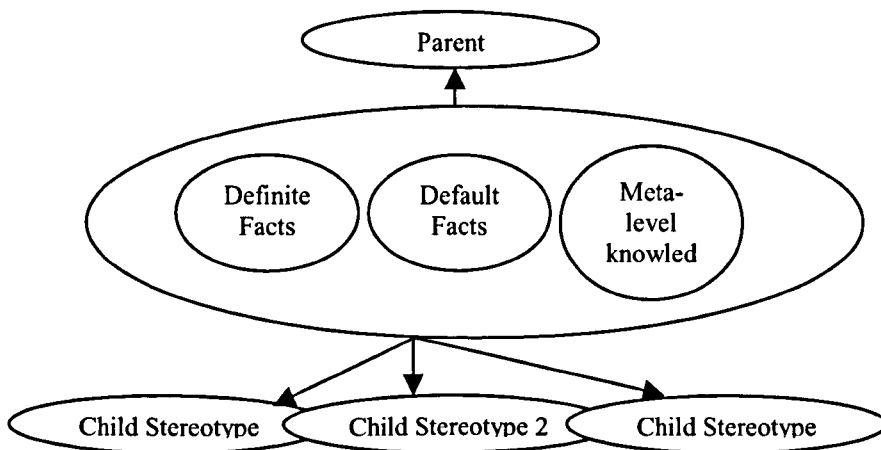


Figure 2.10 GUMS Stereotype components - each stereotype in GUMS is composed of definite facts and rules, default facts and rules, and meta-level knowledge. Stereotypes may have a parent stereotype, unless it is the more general one, i.e. root stereotype. In addition to that, any stereotype may have one or more children that inherit all its facts and rules.

2.4.1.2 Default reasoning with rules

In GUMS, each rule or fact in every stereotype, as described earlier, is either definite (necessary) or default. GUMS represents this kind of knowledge within stereotypes

through: a) the predicate certain: to introduce the definite fact or rule, b) the predicate default: to indicate a default fact or rule.

2.4.1.3 Failure as negation

Failure as negation is used as a general technique to collect weak evidence for assumptions that a system might make about a user when no stronger evidence is available. Hence, the default (rule or fact) is obtained from the lack of certain information (i.e. negation) as well as from the presence of certain information. For example, as in [Finin 1989], if there is a programmer stereotype, there can be a rule that can use the user's lack of knowledge about compilers as an indicator of what he/she probably knows about interpreters.

Kobsa [Kobsa 2001] has classified generic user modelling tools or systems to be: Academic user modelling shells and Commercial user modelling servers. In the next subsections, a brief background about each approach will be given.

2.4.2 Academic User Modelling Shells

In the early 1990s, researchers started constructing user-modelling shells that were provided with structures and processes, which were believed to be important for adaptive systems at that time. However, These structures or processes were based upon the experience of the developers of these user-modelling shells through their previous work on adaptive systems [Kobsa 2001]. The term “shell” was borrowed from the field of expert systems, and it was first used by Kobsa [Kobsa 1990] to refer to these generic user modelling systems.

Academic user modelling shells are characterised by being domain independent, therefore, they can serve as many application domains as possible. Moreover, these shells are capable of inferring many facts about their users by stereotyping them according to their different traits and by using their interaction history to deduce additional facts. Furthermore, these shells maintain consistency among gathered facts when contradictions are found by utilising several artificial intelligence techniques such as reasoning with uncertainty and conflict resolution. [Kobsa 2001].

According to Kobsa, academic user modelling shells were not used outside the institutes where they were developed, except the BGP-MS (Belief, Goal and Plan

Maintenance System) [Kobsa and Pohl 1995], which seems to be the only academic user modelling shell that has broken this barrier and has a few reports on external usage. Briefly, BGP-MS is an application-domain free user modelling shell. This shell communicates with applications through inter-process communication, where it can obtain observations regarding users' behaviour from applications and supply them with information based on the held assumptions about users.

2.4.3 Commercial User Modelling Servers

Unlike academic user modelling systems, commercial shells have been applied to real life applications through, for example, web-based applications such as e-commerce. The architecture used among these systems is the client-server architecture. That design gave the ability to these systems to communicate with more than one application. Moreover, client-server architecture provided commercial users' modelling systems with different advantages, such as users information could be stored centrally. Therefore, different applications could share this information. Moreover, that technique helps commercial user modelling systems to provide consistency regarding users traits easily because all the stored information about a particular user could be gathered through different applications. Furthermore, in a user modelling server different methods and tools that protect users' data such as encrypting data, authentication control, etc. could be used and all the applications that communicate with that central repository could benefit from them when they exchange data with that server [Kobsa 2001].

Nevertheless, systems based on client-server architecture have drawbacks as well. For example, these systems require the network to operate efficiently, as any failure in network transparency, such as delays or failures in server request and responses, may affect the performance of the system.

Due the nature of e-commerce applications that use this kind of user modelling systems new characteristics have been added to these systems which are not found in the academic use modelling shells. For example, these systems adopted different modelling methods to provide quick adaptations to users who interact with the e-commerce application for a short period of time. Moreover, different methods and techniques for exchanging information among different user modelling servers are

employed, such as interfaces and Application Programme Interface (API). In addition to other characteristics such as deploying different mechanism in case of a network break down and the ability to serve more applications and users through load distribution techniques [Kobsa 2001]. There are many different tools for web personalization with different capabilities on the market. For a comprehensive review about commercial user modelling servers see [Fink and Kobsa 2000].

2.5 Summary

Throughout this chapter an overview has been given about user modelling from different perspectives, considering its important characteristics, different techniques of representation, and the diverse types of user modelling that are in common use. Moreover, effort has been made to draw some borders, as user modelling commonly involves areas of artificial intelligence such as Natural Language Dialogue (NLD) systems, multiple-agent planning systems, text comprehension and generation, intelligent help systems, game playing and expert systems [Wahlster and Kobsa 1989]. Within these boundaries the characteristics of user models, and some of the representation techniques used in adaptive hypermedia systems and Intelligent Tutoring Systems (ITS) have been discussed. Special attention has been given to the representation techniques embodied in this chapter, as techniques that deal with the certainty and uncertainty issues in user modelling have been avoided, as these issues are out of the scope of this chapter and this thesis.

This chapter is deployed as follows:

1. **Introduction:** this section gives a general overview about what is meant by user modelling and its important characteristics used in many adaptive hypermedia systems and ITS.
2. **Types of user modelling:** in that section a discussion about different types of user modelling is given in addition to the advantage and disadvantage of each of these types.
3. **Representation techniques:** that section deals with representation techniques which involve knowledge representation, which is important for any ITS and adaptive educational hypermedia systems, and stereotyping, upon which most of the generic user modelling systems are based. That section does not deal with any certainty or uncertainty techniques.

4. **Generic user modelling systems:** characteristics and types of generic user modelling systems are described here. These systems act as an independent component/software with any application to provide user-modelling services.

Chapter III: Adaptation in Hypermedia

3.1 Introduction

In the previous chapter user modelling types and techniques, which are important for adaptive hypermedia systems to provide users with personalization, were explained. The question that arises now is how could those systems provide such adaptation? To be able to answer this question it is important to clarify that adaptive hypermedia systems are composed of different nodes/pages, which include different sorts of information, and those nodes/pages are connected together through links. Therefore, adaptive hypermedia systems provide adaptation on two levels: content level and links level. Content level adaptation means that the presented information is adapted to suit users' knowledge, goals, etc. by including extra explanations, removing some parts of the presented text, and so forth, as will be explained later. On the other hand, links level adaptation means adapting links between nodes in the hyperspace by annotating them, removing links to irrelevant nodes, etc. to provide users with links to nodes that suit their different traits. Thus, it could be understood that adaptive hypermedia augments the functionality of classical hypermedia from only presenting information that could not be presented through conventional papers to more effective role, which is personalizing this information to every individual's user characteristics. Hence, adaptive hypermedia systems are very useful when they are likely to be accessed by users who are diverse in their knowledge, background, preferences, etc.

In the following sections, methods and techniques used to adapt information on the content level and links level will be discussed.

3.2 Content Level Adaptation

The goal of content level adaptation is to provide each user with appropriate information that suits his/her traits such as knowledge, goals, preferences, etc. For example, users with less knowledge about presented concepts/materials can be supplied with extra explanations while those who know more can access advanced

level materials without the need for these extra basic explanations. Moreover, according to users' preferred learning styles, such as more examples than abstract descriptions, or the converse, the form of the information (examples, case studies, etc.) could be altered. Therefore, each user will access the form of information that efficiently helps him/her to understand the presented material. It is important to notice that this form of adaptation personalises text content through different techniques as will be described later.

However, the content of any hypermedia page may be a text or a set of various multimedia items. Therefore, multimedia items are also subject to personalization through what is called 'Adaptation of modality' [Kobsa et al. 2001]. Through that type of content adaptation, adaptive hypermedia systems could choose from different types of media, such as video, animation, etc., to present information to users. This kind of choice depends on users' different traits such as their preferences, learning style and so forth. Thus, content level adaptation provides each user with a kind and a form of information that suit his/her different characteristics. [Brusilovsky 1996, Brusilovsky 2001].

Adaptive hypermedia systems may utilise different methods to apply content level adaptation [Brusilovsky 1996], for example, providing extra explanations to users who have less knowledge about the presented material, while those who have a higher level of understanding may access more advanced information; this method is called 'Additional Explanation'. Moreover, in adaptive educational hypermedia systems such as WHURLE-HM [Zakaria and Brailsford 2002], this method is usually combined with stereotyping users' knowledge into different classes and each user can access information that suits his/her knowledge stereotype. Furthermore, systems can provide content adaptation according to users' preference or learning styles by storing different versions of the same hypermedia page or fragments of it, this method is called 'Explanation variant'. Therefore, the presented material could be presented in different format to different users who are diverse in their favoured learning styles, preferred presenting language (such as in TANGOW [Carro et al. 1999]), etc. Another method that could be used is to order fragments of information about a particular concept according to users' different traits - this method is called

“Sorting”. Thus, fragments of information that are most relevant to a user’s knowledge and background are placed in a more prominent position followed by the less relevant and so forth, such as in Hypadapter [Hohl et al. 1996].

Combining these methods could provide a useful means of adaptation. For example, the “additional explanation” method could be used to present users with information that suits their knowledge stereotype, and the “explanation variant” could be used to present this information in a way that users may prefer such as text only, pictures and text, etc.

Different techniques have been used by many adaptive hypermedia systems in different areas (education, e-commerce, etc) to implement these adaptation methods, such as: [Brusilovsky 1996, Brusilovsky 2001, Kobsa et al. 2001]

- *Conditional text*: This technique is widely used to apply the “additional explanation” method. The idea behind it is to break up the information about a particular concept(s) into fragments (as in AHA! [De Bra et al. 2002b]) or independent chunks (as in WHURLE-HM [Zakaria and Brailsford 2002]). Furthermore, each fragment or chunk is associated with a condition related to a user’s specific trait such as his/her knowledge level, preferences, languages, etc. thus, if a user satisfied the condition(s) coupled with a particular fragment or chunk, thereby, he/she can gain access to this piece of information.
- *Stretch text*: In some hypermedia pages, hot words could be found within the content of a page. These hot words acts like links. However, when a user activates any of these hot words the related text (description for example) is presented either in a new window or extends the content of the current page. Adaptive educational hypermedia systems, such as Metadoc [Boyle and Encarnacion 1994], employed this technique to provide content adaptation. In this system, the content of a page is adapted by un-expanding explanations for technical expressions/concepts and expanding those whose description is important to be accessed by a user - depending on his/her user model. Furthermore, users could participate in providing extra adaptation by expanding an explanation for a specific concept by activating the associated hot word. Thus, the system takes into account these changes for the presented concepts for future

adaptation. Such a technique could be utilised to implement the “additional explanation” method. Nevertheless, users have the option to access extra information if the need arises.

- *Page and Fragment variants*: These techniques could be used to apply the “explanation variants” method. The idea is to store different versions of the same page (page variants) or the same fragments (fragment variants) of that page. Each fragment or page version is presented to users based on their knowledge, language, preferences, etc. The only difference between page variants and fragment variants is that the latter is a fine-grained version for the former one. However, fragment variants provide more flexibility than page variants. For example, fragments could contain different language versions combined with versions targeting users with different knowledge levels. Therefore, a page that is composed of dynamic fragments could provide users with effective adaptation because it could simultaneously merge different user-traits. On the other hand, a page variant could be adapted to only one aspect of users’ traits at a time and it is difficult to combine different aspects of users’ traits, such as learning style, language, knowledge level, etc. at the same time unless a system that uses this technique has different patterns for the same page, such as certain language merged with a specific knowledge level for all the presented concepts and a particular learning style, which is not feasible as in fragment variants.
- *Fragment colouring*: in the fragment colouring approach, the content of the hypermedia remains unchanged for all users. For each individual user, parts of the presented hypermedia pages are marked out as being important, or not important, etc. depending upon the users’ characteristics. Therefore, this technique acts like a marker that highlights sections in a “textbook”, so that all users can see all available information. Fragment colouring technique is not practical in certain contexts such as in education where the offered information cannot be presented in the same manner to all users. Fragment colouring has the same function as linking annotation technology in links level adaptation (as explained later). Thus, it is suggested calling this technique fragment annotation. Moreover, it is proposed that such a technique could be used to apply the sorting method more efficiently rather than any of the mentioned methods. For example,

the more relevant fragments are in the top with a specific colour, and the next relevant one will have another colour, and so on.

- *Frame-based technique*: this technique presents all information about a particular concept in a frame. Slots of the frame may hold examples, extra explanations, etc. The decision on allocating slots to a certain user is based on his/her knowledge level, learning style and other characteristics such as in Hypadapter [Hohl et al. 1996]. In this system, there are rules that calculate the priority of each slot depending on a user's traits such as his/her knowledge level about a presented concept, learning style, etc. According to these rules, the user receives these slots ordered from the most relevant to the least relevant. This technique could be used to apply the sorting method.

3.3 Links Level Adaptation

Links level adaptation is employed by different adaptive hypermedia systems to help users find their paths in hyperspace by adapting the presented links according to their goal, knowledge and other features. This is achieved using different techniques such as [Brusilovsky 1996, Brusilovsky 2001, Kobsa et al. 2001]:

- *Direct guidance*: using direct guidance, a hypermedia system guides users to the next page to visit according to their characteristics that are presented in the user model. Such a technique has been used by different adaptive educational hypermedia systems such as Interbook [Brusilovsky et al. 1998] and ELM-ART [Brusilovsky et al. 1996b]. These systems apply the direct guidance technique by means of providing a "TEACH ME" button that is dynamically connected to the best node that suit users' knowledge levels. According to Brusilovsky [Brusilovsky 1996], such a technique does not provide users with the opportunity to navigate the hyperspace without automatic guidance. On the other hand, direct guidance could be useful especially for users who do not have experience with the system's hyperspace structure and cannot choose the best node that suits their knowledge to visit. Therefore, such a technique could help users to achieve their goal more rapidly.
- *Sorting*: using this technology, links within a particular page are sorted according to the user model with the most relevant links located at the top. The

sorting technique is widely used among information retrieval systems such as web search engines (Google.com, infoseek.com, etc.) where links are sorted according to their relevancy to a user's goal. Moreover, it has been used by adaptive educational hypermedia systems such as Hypadapter [Hohl et al. 1996]

- *Link deactivation*: the idea behind this technique is to conceal from users links that guide them to pages that present information not relevant to their knowledge, goals, etc. Link deactivation could be achieved through either: a) disabling links by removing their functionality without any change in their appearances, such as in TANGOW systems [Carro et al. 1999], b) hiding links by changing their appearances to look like a normal text such as in AHA! [De Bra and Calvi 1998b], or c) removing links such as in CHEOPS [Ferrandino et al. 1997]. De Bra and Calvi [De Bra and Calvi 1998a] have illustrated that when they used link disabling in an early version of AHA! combined with some kind of hiding, users were not satisfied as they could easily guess the presence of many links. Therefore, the link hiding technique requires that hidden links should be difficult to spot by users. On the other hand, such problem is not found in link disabling or link removal.
- *Adaptive annotation*: using the adaptive annotation technique, links to pages are annotated according to the relevance of these pages to users' different traits such as knowledge, goal, etc. therefore, through this technique, users become aware about the status of information held by pages that these links point to. Annotations could be provided in different forms, such as textual form or visual form (colour, fonts, icons). The simplest form of annotation is found in web browsers where they differentiate between visited and unvisited links by changing links' colours. This technique has also been used widely in many adaptive educational systems such as NetCoach [Weber et al. 2001] and AHA! [De Bra et al. 2002b], and others, which suggests that it is a successful technique. Moreover, this technique has been used with WHURLE-HM [Zakaria and Brailsford 2002] to inform users about their lessons' status: a) not finished lessons, b) finished lessons and c) new lessons.
- *Collateral structure adaptation*: content adaptations in some way could implicitly embrace link adaptation. For example, a fragment of a page may contain link(s) that will not be presented to users because the fragment itself will not be

presented, such as in WHURLE-HM [Zakaria and Brailsford 2002]. Such a technique is very efficient for systems whose data model is composed of independent chunks where each chunk might hold text, picture, etc. in addition to an external/internal link(s) such as in WHURLE [Brailsford et al. 2001].

Integration of different techniques could provide a flexible and efficient method of link adaptation such as such as Interbook [Brusilovsky et al. 1998], which uses adaptive annotation and direct guidance, Hypadapter [Hohl et al. 1996] that utilizes link sorting, hiding and annotation, and AHA! [De Bra and Ruiters 2001] that uses links removal and annotation, and others.

The following chapter (Chapter IV) demonstrates how content level adaptation and link level adaptation are implemented in different popular adaptive educational hypermedia systems. Moreover, a description about how such techniques are merged together in some systems is given. Furthermore, Chapter VI illustrates how WHURLE-HM implemented collateral structure adaptation to provide personalisation to its users on content level and link level. Also, Chapter VI shows how WHURLE-HM utilised links annotation technique using different font colours and abbreviated words (for browsers that cannot render java scripts) to inform users about the status of their lessons (new lessons, opened lessons, and finished lessons).

3.4 Summary

Throughout this chapter, an attempt has been made to draw a general overview of adaptation techniques and methods on the level of content adaptation and link adaptation. Moreover, examples have been given of systems that use these techniques, which are described in detail throughout the survey chapter (Chapter IV).

This chapter is organized as follows:

- **Introduction:** this section gives an idea about the general goal of the chapter.
- **Content level adaptation:** throughout this section, different methods and techniques used in adapting contents (adaptive presentation in [Brusilovsky 1996]) of hypermedia pages are discussed.

- **Links level adaptation:** in this section, different links level adaptation techniques (adaptive navigational support in [Brusilovsky 1996]) have been explained.

Chapter IV: Survey

4.1 Introduction:

Research in adaptive hypermedia dates back to the beginning of the 1990s, as at that time the hypertext and user modelling fields, from which adaptive hypermedia came, were active and mature. However, 1996 is considered to be a turning point in this area of research, and since this time adaptive hypermedia has gone through a period of rapid growth [Brusilovsky 2001]. This rapid development was a result of two main factors. The first of these is the rapid growth in the WWW that occurred around 1996, which led to massive amounts of information being available to users who had different goals and background knowledge, etc. This situation caused a demand for personalization that boosted research into techniques of adaptation in hypermedia and on the WWW. The second factor is the accumulation of research experiences from field studies, as most earlier systems were experimental, and were only implemented to explore new ideas in laboratories. However, it is important to note that much of the research that has been published since 1996 is based upon these early investigations. Nevertheless, systems that have been developed since that date are dedicated to real world usage [Brusilovsky 2001].

Brusilovsky [Brusilovsky 1996, Brusilovsky 2001] has categorized the present systems into six distinct classes:

- *On-line information systems*: The goal of this group of systems is to provide users with diverse backgrounds access to different sorts of information such as products, history, support, etc. for example, e-commerce systems, such as Amazon.com, and electronic encyclopaedia such as Microsoft Encarta.
- *Online help systems*: this group of systems provides information about computer applications, such as spreadsheets or programming environments. They are similar to the former group but, unlike them, they are not self-contained. Rather, they are attached to their parent application, and they have a relatively small hyperdocument that is limited to that domain.
- *Information retrieval hypermedia systems*: the goal of these systems is to retrieve information or documents, and to provide hypertext links by the means of index terms. The docuverse in such systems is potentially very large and thus difficult

to be manually structured. For this reason links are automatically computed by the system. As an examples of such systems are search engines such as Google.com, infoseek.com, and others.

- *Institutional hypermedia systems*: these systems serve the information required by an institution to support the daily work online. Thus, these systems are considered to be the medium for everyday work of many employees. Therefore, according to their employee position in an enterprise, they may use only a specific area of the hyperspace and, according to their current working goal, they may need access to a very small subset of it. This kind of system was developed as a loosely related database, but in recent systems these databases are gathered into a single hyperspace that could be relatively large.
- *Systems for managing personalized views in information spaces*: this kind of system helps users to build their own personalized views of the entire hyperspace, in order to protect them from its complexity. This group of system is similar in some way to institutional hypermedia and other kinds of information systems where users need a suitable access to a subset of information space for daily work. Personalized views in the WWW require permanent management, such as searching for new and relevant items and identifying expired or changed items.
- *Adaptive educational hypermedia systems*: these will be discussed in detail in the following section.

Adaptive educational hypermedia systems, information retrieval hypermedia and on-line information systems are the most widespread applications of adaptive hypertext technology [Brusilovsky 2001]. Furthermore, adaptive educational hypermedia systems are the most commonplace within these. The reason comes from the fact that there is no clear-cut border between those systems, as on-line information systems and information retrieval could also be used in the educational context.

4.2 Adaptive Educational Hypermedia Systems

In traditional web-based educational systems, the contents are generally static, in so far as, once written, their contents cannot be changed without external intervention. This provides a uniform learning experience, regardless of the needs and requirements of the learners. One example of such a system that is in widespread use

is WebCT [Curtin 2002, Piguet and Peraya 2000]. This is a hypermedia educational system developed in 1995 at the University of British Columbia. It is an environment for authoring and delivering educational materials over the web. WebCT presents a static and inflexible pedagogic experience, without any kind of adaptation at the user level. Hence web applications, such as WebCT, Blackboard¹, COSE-k12² and others, which tend to be designed with a particular class of users in mind, may not be suitable for those who are even marginally different from the original target [Eklund et al. 1997].

Adaptive educational hypermedia systems can address these issues by altering the information presented to each user depending upon a defined set of characteristics [Brusilovsky 1996]. For adaptive systems to adapt their contents, a model for interacting with users to deliver them the information that suits their knowledge, goals and background is needed, as user-modelling is a prerequisite for any adaptive system, including educational systems.

According to Brusilovsky [Brusilovsky 2000], most early Intelligent Tutoring Systems (ITS), such as ELM-PE [Weber and Möllenberg 1994], provide almost no material, as the main task for these systems is to help students in the problem solving process, as described in Chapter I. In addition, it was believed that the required knowledge is acquired from outside the system, such as classrooms or reading books. On the other hand, hypertext or hypermedia provides the ability to organize educational materials over the web.

Adaptive educational hypermedia, as explained in Chapter I, combined ITS and hypermedia together, as it inherited from intelligent tutoring systems (ITS) the usage of the knowledge about students, domain and teaching strategies, and from adaptive hypermedia systems, the adaptation of contents and links of the hyperdocuments to users according to different forms of user model [Brusilovsky 2001].

In the following subsections, examples of successful and popular systems that employ many of the user modelling and adaptation techniques (discussed in Chapter

¹ www.blackboard.com

² www7.nationalacademies.org/cose/

II and III) are described in detail. For example, the AHA! system that is a well known system and extensively developed since it was introduced in 1996 is investigated. Moreover, ELM-ART system, which is considered as one of the earliest web-based educational hypermedia systems, in addition to NetCoach and Interbook systems, which are derived from it and followed its approach, are explained. The MetaDoc system that helped the researcher in developing the rationale behind the hybrid model through its ability to handle two knowledge domains is also illustrated. Moreover, the CHEOPS system was chosen because of the unique approach it used in representing its knowledge model as a knowledge pyramid. Furthermore, The Hypadapter system shows an example of the frame-based technique described in contents level adaptation, which was explained in Chapter III. In addition to this, the TANGOW system was selected to demonstrate an example of the link disabling technique in order to apply links level adaptation, which was discussed in Chapter III.

4.2.1 AHA!

AHA! (Adaptive Hypermedia Architecture) system has been developed from a static hyperdocument system that was used to deliver a hypertext course “2L670: Hypermedia Structure and Systems” in the Eindhoven University of Technology, in the department of computer science. That course did not have any hierarchical structure, and it aimed to teach students how to create hyperdocuments that are easy to use, and how to build hypermedia systems that are rich in terms of navigational capabilities and tools. In this system, there was no adaptation, as all students had to read the course content, which was the same for all students, and then answer a multiple-choice quiz to prove that they had absorbed the course concepts well. Students’ grades were dependent on assignment work, as students were only allowed to progress to an assignment if they answered all of the 20 multiple-choice questions correctly [De Bra 1996]. In 1996 the AHA! System was introduced, where adaptation techniques are employed. The adaptive hypermedia engine within AHA! is responsible for maintaining the user model and performing adaptation for information contents as well as link structure.

The user model in an early version of AHA! [De Bra and Calvi 1998a] was composed of a set of Boolean values. Each of these values represents users’

knowledge about each concept in the domain; also each page is associated with a concept, as discussed later. Knowledge about concepts is generated when a user reads a page or takes a test. For example, if a user accessed a page holding a concept called ‘introduction to hypermedia’, the Boolean value for that concept with respect to that user changes from false to true (i.e., known by the use).

Through that system, two levels of adaptation are applied: adaptive presentation (contents level adaptation) and adaptive navigation support (links level adaptation). Adaptive presentation techniques [Brusilovsky 1996] are used to adapt a document’s content, either textual or multimedia, according to the user model. Brusilovsky in [Brusilovsky 1996, Brusilovsky 2001] has discussed many techniques used in adaptive presentation, which are described in Chapter III. AHA! applies the fragments variants technique through utilizing conditional texts. In AHA! the conditional texts means pieces of HTML, which may include objects like applets or pictures, that may be included or excluded from the presented page/node depending on whether or not a condition of it is met. Conditions are written in the form of HTML comments in the HTML files and the adaptive engine process these conditions. For example, as stated in [De Bra and Calvi 1998b]:

```
<!-- if definition and history -->
this part appears if the two concepts (definition and
history) are both known according to the user model.
<!-- else -->
if this is not the case then this alternative is presented instead
<!-- endif -->
```

On the other hand, adaptive navigation support changes the link structure between pages/nodes that together perform the hyperdocument [Brusilovsky 1996]. That change mainly depends on a user model. AHA! supports link hiding and link annotation. In link hiding, the user is guided by the system by not showing links to pages/nodes the system believes are not relevant to the user. That kind of link adaptation is implemented in AHA! by changing the colour of links to irrelevant pages/nodes to that of the normal text (Black). Thus, links are still there, but a user cannot distinguish between them and the rest of the text. Moreover, AHA! supports link annotation by colouring links with respect to their relevance and access (if visited or not) to the user. Thus, the links to nodes/pages that are relevant to a user are annotated by a blue colour, visited links are in purple, while irrelevant links are in

a dark grey colour. In AHA! users can choose between link hiding and link annotation depending on the way they configure their web browsers (i.e., if a web browser is configured to not underline links, therefore link hiding will be applied, and if not, link annotation will be activated) [De Bra and Calvi 1998a]. Users in AHA! can configure the colour scheme through a setup form, which will be stored in their user model. In addition to link hiding and link annotation, link removal could be realized as well, where link removal is a technique that links to nodes that the system finds irrelevant to a user's knowledge level at that time are removed. In AHA!, this kind of link adaptation is achieved by wrapping the HTML anchor tag with a conditional text. For example, as in [De Bra and Calvi 1998b]

```
<!-- if desired -->
<a href="..."> here is the link anchor text </a>
<!-- else -->
  here is the link anchor text
<!-- endif -->
```

Apart from link adaptation, AHA! supports non-adaptive links, which enable an author to create static links that always present in addition to the adaptive links that are dependant on the user model.

In that version of AHA! [De Bra and Calvi 1998b], each page consisted of four parts:

- Header: contains the definition of the style-sheet with the link colouring scheme.
- Body: starts with an author-defined header that is automatically included in every page.
- Page content.
- Footer.

Pages that are delivered to users are assembled from these four parts by means of a CGI-script. Each page has prerequisite concepts that have to be known before accessing it as well as an outcome concept. These prerequisites and outcomes are defined by the pages' authors as comments in the beginning of each page. For example, a prerequisite could be defined as [De Bra and Calvi 1998b]:

```
<!-- requires readme and history but not (intro or definition) -->
```

And the outcome concept could be as in [De Bra and Calvi 1998b]:

```
<!-- generates history -->
```

From these two lines in every page the system creates: a dependency file, which contains pairs of page names and their prerequisites, and a concept list, which contains all the concepts in the domain, where users have the ability to change the user model by altering the value for each concept to true or false.

In AHA!, links are ranked into three levels:

- Good/desired: links to unvisited relevant pages, which always hold the blue colour (border colour in case of image).
- Neutral: links to pages that have been visited previously, and for which the prerequisites are satisfied, hold the purple colour (border colour in case of an image).
- Bad/undesired: links to pages where a user hasn't fulfilled their prerequisites hold the black colour (border colour in case of image).

Also the AHA! script defines three types of linking:

- External: these are links to pages outside the AHA! hyperdocument. These links are in a red colour and, if followed, change to dark grey.
- Conditional: links that point to pages with prerequisites. If a user satisfied the requirements to which a link points, this link changes to "good" level and its colour becomes blue, but if the page is visited before its level changes to "neutral", then its colour changes to purple. On the other hand, if the user did not match the requirements of that page the link changes to level "bad" and its colour turns to black.
- Unconditional: links points to pages with no prerequisites, thereby they belong to "good" or "neutral" level depending on the access state of the destination page(s) (i.e., if a page has been visited before or not).

In a newer version of AHA! [De Bra et al. 2000], the way the user model is handling users' knowledge has changed. Instead of representing knowledge about concepts as a set of Boolean variables, a scale of integer numbers between 0 and 100 is used. Moreover, in addition to fragments and concepts that are associated to pages, a new term is added, abstract concepts, referring to concepts that are not pages [De Bra et al. 2000]. Pages in that version of AHA! are written in XML files instead of HTML. That version distinguishes three types of concepts relationship:

- Link relationships: AHA! realizes HTML anchor tags that represent links between pages.
- Generate relationships: in the new AHA! the access to a page may change several elements in the user model (i.e., update the knowledge value of other concepts with respect to a user). The generate relationship specifies these updates and they are stored in a special XML file. In the early version of AHA! the generate relationships could be considered as the outcome from accessing a page, i.e., from which students have learned. In addition, it was used to define HTML comment inside that page.
- Requirement relationships: the desirability for a fragment to be included in a page or for a certain page to be linked to depends on conditions specified by requirement relationships. These requirement relationships for pages and abstract concepts are stored in a special XML file, while those for fragments are included on inside pages.

That version of AHA! consists of three parts: domain model, user model and adaptation model. The adaptation model consists of generate rules and requirement rules, in addition to some system-defined adaptation rules that define the behaviour of AHA!

The requirement rules could be defined as [De Bra et al. 2000]:

```
<concept>
  <conceptname>theconcept</conceptname>
  <relationexpression>req1 > 30 and req2 < 80 </relationexpression>
</concept>
```

In the above example, the requirement for concept “theconcept” is that the knowledge value of concept “req1” is bigger than 30 and that of concept “req2” is less than 80.

The generate rule also could be expressed as [De Bra et al. 2000]:

```
<genitem>
  <name>concept1</name>
  <genlist>concept2:+40 concept3:-30 concept4:50</genlist>
</genitem>
```

From the above example, it could be observed that the outcome of concept1 is an increase in the knowledge value of concept2 by 40% of the knowledge value of

concept1 (X), decrease concept3 by 30% of X and give an absolute value 50 to concept4. In the early version of AHA! the outcome was just altering users' knowledge with respect to a page's concept to true.

On the other hand, conditions for including fragments could be defined as [De Bra et al. 2000]:

```
<if expr="req1 > 30 and req2 < 80" >
  <block> here is the conditionally included fragment</block>
</if>
```

In the above example, the XML fragment is included if the knowledge value of concept "req1" is bigger than 30 and that of concept "req2" is less than 80.

According to the new scale for knowledge values, if a user has followed a link to a desirable/relevant page his/her knowledge level about the concept held by that page increases to 100. On the other hand, if a user followed a link to an irrelevant page, his/her knowledge value increases to 35, in a case of 0, or remains as it is if it is 35 or more [De Bra and Ruiter 2001]. In the early version of AHA!, if a user followed an irrelevant link, his knowledge value about that undesirable page would not change.

As shown above, the generate rules change the knowledge value of concepts associated with the updated concept with respect to a user that results from him/her visiting a page or taking a test. Also these concepts change the knowledge values of their associated concepts with respect to the same user and so on. Thus, the updating process becomes recursive. Recursive updating could end with some problems like infinite loops, continuous updating of the knowledge value of concepts which results from visiting a page more than once, or if a concept updates two other concepts and each of these concepts results in updating the same concept [De Bra et al. 2000].

In that version of AHA! [De Bra et al. 2000], four cases of updating have been distinguished:

- Monotonic update: a monotonic clause "+" in generate rule is only allowed to update abstract concept and not page concept. Moreover, the updating algorithm continues recursively for that concept.

- Non-monotonic update: a monotonic clause “-” in generate rule is only allowed to update abstract concept and not page concept. Moreover, the update algorithm performs the update but does not continue recursively for that concept.
- Absolute update: page concepts only are allowed to have a generate rule with absolute update, and the update algorithm performs the update but does not continue recursively for those concepts.
- Self-update: this kind of update is useful for concepts to update themselves. Page concepts are only allowed to have a generate rule with self-update. The update algorithm does not allow the rule for those concepts to be executed again.

As an example of these four types of updates, if a user visited a page that presents a particular concept about milk products (concept1), and this concept provokes or generates other three concepts: cheese types (concept2), chocolate types (concept3) and machines used in milk industry (concept4). Moreover, the generate rule for concept1 is as follows:

```
<genitem>
<name>concept1</name>
<genlist>concept1:0 concept2:+60 concept3:-20 concept4:70</genlist>
</genitem>
```

By analysing this code, it could be observed that if the page concept (concept1) is augmented by value Y, then concept2 will be incremented by 60% of Y and the generate rule for concept2 will be executed. Futhermore, concept3 will be decremented by 30% of Y and concept4 will have an absolute value of 70. However, the generate rules for concept3 and concept4 will not be executed. Finally, concept1 will reset itself to 0, and as a result, when a user visits the same page again the same effect will be carried on.

In AHA!, concepts of the domain are in a hierarchal form. For example, in generate relationships, concepts are arranged automatically in top-down form as shown in Figure 4.2, modified from [De Bra and Ruiter 2001]:



Figure 4.1 AHA! Domain Concepts - concepts in generate relationships are in a hierarchal form.

From Figure 4.1, it could be perceived that reading about “godiva” shows interest in chocolate and also means the user has learned something about chocolate [De Bra and Ruiter 2001]. Moreover, learning something about chocolate means the user has learned something about Belgian products. And learning about Belgian products means the user has learned something about Belgium. AHA! provides an authoring graphical interface for entering the generate relationships for which the given concept is the source. Each time the author adds or updates generate relationships the whole structure is updated accordingly.

In previous version of AHA!, users have the ability to change the Boolean value for each concept from true to false or vice versa (i.e., unknown to known). In the newer version, authors have the ability to determine which concept could be changed by users by indicating if it is “changeable” or not via a graphical interface.

AHA! contains a module for multiple-choice tests. That module randomises tests to make cheating among students somewhat difficult. These tests can be created using plain HTML and are linked to pages that explain whether an answer is wrong or not. When a user chooses a wrong answer, the explain-page reduces the knowledge value for associated topics and resets the value of associated pages to 0. After the user reads the material again, his knowledge value about associated topics and pages increases once more.

In the most recent version of AHA! [De Bra et al. 2002a], multiple attributes per page are used, where authors can choose attributes’ names that match the usage of those attributes. Via this new approach, it became possible to indicate interest in a certain concept as well as knowledge about that concept. Those attributes could have

a string, numeric or Boolean variables. For example, the requirement for a certain fragment to be included in a page, or a link to a certain page to be desirable, could be defined as follows:

`Chocolate.interest > 60 and French_fries.interest < 30`

Moreover, unlike previous versions of AHA!, each rule in the generate rules associated with every page and concept has a condition to be met before a certain action may be performed. It is also possible to specify an alternative action to be performed in case the condition is not met. For example, the interest in painting should be high to understand everything on the page about Picasso.

Concept: `painting.interest > 70`

Action: `Picasso.knowledge: = 100` (if interest level in painting more than 70)

Alternative Action: `Picasso.knowledge: =35` (if interest level in painting less than 70)

All that has been described up to now belongs to AHA! version 1 and its extensions until AHA! version 2 (AHA!2) [De Bra et al. 2002b] is reached. New features in AHA!2 are based on the AHAM (Adaptive Hypermedia Application Model) reference model [De Bra et al. 1999a], which is an extension to the Dexter [Halasz and Schwartz 1990] hypertext reference model.

The Dexter reference model is an outcome from two small workshops on hypertext, the first one organized by John Leggett and John Walker that brought experienced hypertext system designers together in October 1988 at the Dexter Inn in Sunapee, New Hampshire, and from which the model had its name [Halasz and Schwartz 1990].

Briefly, that reference model serves as a starting point for the design of hypertext systems. The Dexter model divides a hypertext system into three layers: storage layer – describes different components that form a hypertext system, such as links, chunks of text, pictures, etc.; within-component layer – concerned with the contents and the structure within these components; and the run-time layer – deals with the user interface aspects in a hypertext system. Between the storage layer and the within-component layer, there is an interface called anchoring, which is a mechanism that refers to an item or location within the content of an individual component. For example, linking to a certain paragraph in a text page. In addition, between the

storage layer and the run-time layer there is another interface called specification presentation, which holds information on how a component could be presented to a user. For full information about the Dexter reference model refer to [Halasz and Schwartz 1990].

In AHAM, as an extension to the Dexter, the storage layer is divided into three areas: domain model, user model and teaching strategies. The teaching strategies hold pedagogical rules that are used by an adaptive engine to create the specifications presentation [De Bra et al. 1999a].

AHA!2 deviates from AHAM by not separating the domain model from the adaptation model - concepts are defined along with their prerequisites and generate rules. Generate rules in AHA!2 are condition-action based as illustrated in the above example. Moreover, each user aspect, such as his knowledge, goal, interest, etc., could be represented through concepts or different attributes of a concept or concepts. This means the combination between related adaptation aspects such as knowledge about a concept and interest in a concept is possible. The attributes, as in the previous version, can have Boolean values, string values, or numeric values. AHA!2 supports storing the concept structure in XML files or in a mySQL database. The problem of recursive updating in the user model in the previous version of AHA! and the proposed solution for it is no more exist in AHA!2. In AHA!2, rules are allowed to trigger each other provided that the “isPropagating” attribute is turned to true. For example, a part from the code in [De Bra et al. 2002b] is listed below:

```
<generateListItem isPropagating="true">
<req>chocolate.interest<50and
chocolate.interest>4</req>
<trueActions>
<action>
<concept>chocolate</concept>
<attribute>interest</attribute>
<expr>chocolate.interest-5</expr>
</action>
</trueActions>
</generateListItem>
```

In the above example, the “isPropagating” attribute is turned to “true”. This means that the updating action in the “interest” attribute of the “chocolate” concept will affect other concepts.

When a user logs on for the first time a special concept called “personal” is created for him/her. In the concept, user related information such as name, address, e-mail, etc., is stored, and therefore this information could be used in adaptation. In AHA! version 1 login information was not used for adaptation purposes [De Bra et al. 2002b]. For example, a part from the second code in [De Bra et al. 2002b] is listed below

```

<profile>
<record>
  <key>personal.id</key>
  <type>string</type>
  <persistent>true</persistent>
  <value>debra</value>
</record>
<record>
  <key>personal.email</key>
  <type>string</type>
  <persistent>true</persistent>
  <value>debra@win.tue.nl</value>
</record>
<record>
  <key>personal.goodlink</key>
  <type>string</type>
  <persistent>true</persistent>
  <value>0000ff</value>
</record>
.....
</profile>

```

AHA!2 provides JAVA-based authoring tools for authors to enter the concept structure as well as generate rules to update the user model [De Bra et al. 2002b]. Moreover, whenever a user clicks on a link, the adaptive engine that is composed of JAVA SERVLETS filters the requested page with respect to the user model, also the user model is updated according to the generate rules that come from that page.

4.2.2 ELM-ART

ELM-ART (Episodic Learning Model-Adaptive Remote Tutor) [Brusilovsky et al. 1996b, Weber and Specht 1997b] is a web-based introductory LISP course. That system is based on ELM-PE (Episodic Learning Model-Programming Environment) system [Weber and Möllenberg 1994], which is an intelligent learning environment that supports example-based programming, intelligent analysis of problem solving, and advanced testing and debugging facilities. ELM-PE is designed to support beginners learning the programming language LISP, and was used for a long time in

introductory LISP courses at the University of Trier [Brusilovsky et al. 1996b]. Course materials (printed) were presented to students in regular classes as well as to self-studying students, and then they used the ELM-PE system to practise their knowledge by working on exercises. The system is passive as long as help is not requested. Immediate feedback from the system is given for syntactical error, and if a student wants to work on further exercises while the solution to the previous one is still incorrect [Brusilovsky et al. 1996b]. Moreover, ELM-PE is based on ELM (Episodic Learning Model) model [Weber 1996] that stores knowledge about users in the form of episodes or stages. ELM-ART is a web-based version of ELM-PE that is used not only to solve exercises but to provide online adaptive materials as well. In ELM-ART, the materials are written in HTML format, and de-composed into small subsections and text pages that are associated with concepts to be taught. Relationship between concepts is presented through a conceptual network where concepts' prerequisites and outcomes are described. The user model in ELM-ART records every interaction of the user with the system (implicit user modelling – explained in Chapter II), as for every visited page the corresponding concept is marked [Brusilovsky et al. 1996b]. ELM-ART uses the link annotation technique (explained Chapter III) for the links level adaptation, where links are annotated according to the information found in the user model. Next to each link there is a coloured ball that indicates the status of a link as follows [Brusilovsky et al. 1996b]:

- Red Ball: means the page or the section the link is pointing to not ready for the user to visit it, as he/she hasn't fulfilled its prerequisites.
- Green Ball: means the text page or the section the link is pointing to ready for the user to visit as he/she has satisfied its requirements.
- Yellow Ball: means the test page or the section the link is pointing to ready to be visited but not recommended by the system.

In addition to the link annotation, ELM-ART also uses links sorting technique (explained Chapter III) [Brusilovsky et al. 1996b], as links are sorted according to their relevance to users.

Evaluation and diagnosis of problems' solutions are performed in the same way as in ELM-PE. Moreover, the same feedback messages used in ELM-PE are used in ELM-ART.

ELM-ART had different shortcomings in different areas. For example, a user's knowledge level is only updated when a user visits a page. Moreover, the annotation technique adopted by the system does not give users enough information about the state of concepts they visited and concepts they are going to visit. Because of these problems ELM-ART II [Weber and Specht 1997] has been developed. In ELM-ART II knowledge about units to be taught is represented in terms of conceptual network, where concepts' prerequisites and outcomes are described as in the previous version. Units are deployed hierarchically into lessons, sections, subsections, and terminal pages that are used to introduce either new concepts or problems to be solved. Each unit is an object that is composed of different parts. For example, a part represents related concepts to that unit, outcome concepts and prerequisite concepts. Another one for test items users have to perform. Moreover, information used to annotate links with respect to a user's model is presented in another part.

The user model in ELM-ART II is updated with each interaction with the system. For each page visited the corresponding unit is marked in the user model, and after a user solves the test items in a test-group or a programming problem correctly, the outcome concepts for that unit are marked as known to that particular user (implicit user modeling).

Test-groups in ELM-ART are a collection of test items that are associated with page units. ELM-ART II supports four kinds of test items [Weber and Specht 1997]:

- Yes-No test items: user answers by clicking on yes or no to each answer.
- Forced-choice test items: user answers by selecting one answer among the alternative ones.
- Multiple-choice test items: user answers by selecting all correct answers.
- Free-Form test items: user types an answer to the asked question into a form.

Test items for a test-group are presented to a user as long as not enough correct answers are given. ELM-ART II provides a feedback on the number of incorrectly answered test items in the last page; also it presents those test items with the user's incorrect answers and the system's correct answers with an explanation for those correct answers. In the user model, all the correctly answered test items of a test-

group are stored. Therefore, when enough test items are answered correctly the outcome concepts of the corresponding unit are marked as known.

ELM-ART II provides two adaptation techniques for the adaptive navigation support: adaptive annotation and direct guidance through the NEXT button (explained in Chapter III).

As in the previous version, the coloured ball next to each link determines its state, but instead of three colours ELM-ART II uses four colours [Weber and Specht 1997]:

- Green Ball: this annotation means that the destination page or section the user has satisfied its requirement and it is ready to be visited.
- Red Ball: means that the destination node is not ready for the user to visit, as he/she has not fulfilled its requirements. But if the user followed the link to that page and he/she succeeded in solving the corresponding test or programming problem, the system infers backwards that all prerequisites for that page are known to that user.
- Yellow Ball: it has a different meaning depending on the kind of destination page:
 - Terminal page with a test or problem page: means the problem or the test has been solved correctly.
 - Other terminal page: means the page has been visited before.
 - Lesson, section or subsection: means the subordinated pages have been learned or visited.
- Orange Ball: it also has a different meaning depending on the kind of destination node:
 - Terminal page: means that the system infers from other successfully learned pages that the user will be familiar with the content of this page.
 - Lesson, section or subsection: means that some of the subordinated pages are not visited or not worked at correctly.

On the other hand, ELM-ART II offers direct guidance by providing the NEXT button, where the system suggests the next best step a user can follow depending on the user model for that particular user.

The direct guidance algorithm in ELM-ART II starts with searching for the next page ready to be visited; that page may include a test or problem to be solved. If the system did not find any page with fulfilled prerequisites, all pages from the beginning of the course are checked to see if any have not been visited. Thus, the first one found is annotated as ready to be visited and the user accesses it. In the case where there is no next page to follow, this means that the user has successfully completed the course.

In ELM-ART II users are allowed to use the code of pre-analysed examples when solving a new problem. Similarity links between examples help users to find relevant examples from their previous experience. In addition, the system can predict users' ways of solving a problem and find the most relevant examples from the individual learning history, which is an important feature for users who cannot find relevant examples by themselves [Weber and Specht 1997].

Another feature that ELM-ART II provides, which is very useful for distance learning students, where students and tutors are in different locations, is the code-level suggestions. If a student failed to complete a solution of a problem or he/she cannot find an error reported when the code is evaluated, the user can ask the system to diagnose the code in its current state. Thus, the system provides a feedback by providing a sequence of help messages of increasingly detailed explanations of an error or non-optimal path [Weber and Specht 1997].

4.2.3 NetCoach

NetCoach [Weber et al. 2001] is an authoring system, which is derived from ELM-ART. That system adapts to learners' knowledge, goal and preferences. The knowledge base in NetCoach is composed of concepts, which are the internal representation of pages that will be presented to learners, and test items. Those concepts define two kinds of relationship between each other:

- Prerequisite relationship: concepts required to be learned before the current one, as a result of which the system guides users through these prerequisites before suggesting the current concept.
- Inference relationship: This relationship specifies which concepts will be affected, inferred as known, if a user has finished other concepts successfully.

The user model in NetCoach is a multi-layer overlay model that consists of four layers:

- Layer 1: indicates whether a user visited a page corresponding to a concept.
- Layer 2: contains information on which exercises and tests the user worked at related to that concept; also whether he/she has successfully worked on them up to a certain level.
- Layer 3: describes whether the concept is known through inference links from more advanced concepts the user has already worked on successfully.
- Layer 4: describes if the user has marked the concept as already known (Overlay model).

In addition, the layers' information is updated independently and each layer does not override any other.

A concept is assumed to be learned by a user if it is either tested to be known, inferred from other learned concepts, marked by the user, or visited in case it is not associated with a test group, which is a group of test items. Therefore, the system integrates explicit and implicit user modelling techniques.

A user's learning state in the user model is computed according to the user's success on working on test items that are collected in test groups and assigned to concepts.

NetCoach employs an adaptive navigation support technique:

- Direct guidance: depending on the user's general learning goal and his/her learning state about concepts, the system suggests which is the next best page to visit.
- Adaptive Annotation: links that are in a table of contents or in an overview of each page are visually annotated depending on users' learning states.

Moreover, users get a warning if they access a page with missing prerequisites. Thus, accessing unsuitable pages is not restricted; also, warnings could be switched off.

Users differ in their goals; some of them may need to explore the introductory part of a course, others may want to go to the advanced part of that course. Thus, learners who decide not to work on a complete course but to fulfil a sub-goal will receive

recommendations about which concept to visit to complete their sub-goal. Thus, authors of courses specify the sub-goals in each course and the necessary concepts that fulfil these sub-goals. Therefore, NetCoach supports global guidance (described in Chapter III)

NetCoach provides tests and exercises in different formats, such as multiple-choice questions, gap filling tests, open questions and email questions. Learners can evaluate their answers with open questions, as this type of question has question-answer examples, while tutors evaluate email questions. The remaining items have hints that show the right answers in addition to the explanation for answers.

Test items are collected in test groups that are assigned to concepts. Test groups could be used as an introductory and final questionnaire. Thus, users' learning state in the user model depends on how they successfully worked on these test items.

NetCoach provides authoring tools that facilitate authoring learning materials, composing tests, defining learning goals, and adapting the layout and behaviour of interface. Moreover, NetCoach offers a chat module and discussion lists that help learners to communicate together and to discuss topics and ask questions. In addition, users could adjust many features in the system for their preferences, such as warning and recommendations being disabled.

4.2.4 Interbook

Interbook [Brusilovsky et al. 1996a, Eklund et al. 1997, Brusilovsky et al. 1998] is a subject-independent authoring tool that simplifies the process of creating adaptive electronic textbooks over the web. Interbook follows the same approach that was described in the ELM-ART system [Brusilovsky et al. 1996a], but with a generalized style – not dedicated to a knowledge domain as LISP programming as in ELM-ART. Interbook uses knowledge about domain, which is represented in the form of a domain model, and about users, which is represented in the form of a user model, to provide adaptation. That system distinguishes two parts in adaptive electronic textbooks: a glossary and a textbook. A glossary is considered as a visualized domain network, as each glossary entry corresponds to one of the domain concepts. Moreover, each glossary entry provides links to all available textbooks'

sections that introduce or require the concept [Brusilovsky et al. 1996a]. Educational material is represented in Interbook as a set of textbooks. Each textbook is hierarchically structured into units of different levels: chapters, sections and subsections. Moreover, each terminal level unit is an atomic presentation, example, problem or test.

To connect textbooks to glossary, units in a textbook are indexed with domain model concepts. Furthermore, textbooks that describe or explain the same subject form a bookshelf [Brusilovsky et al. 1998]; so all books from the same bookshelf are indexed with the same set of domain concepts.

For each unit there is an attached spectrum, which is a list of prerequisite concepts, related concepts and outcome concepts. A concept is included as a prerequisite in the spectrum if it is essential for the understanding of the content of that unit. On the other hand, a concept is included as an outcome concept, if some part of the unit presents a piece of knowledge assigned by the concept [Brusilovsky et al. 1996a].

For every concept in the domain model, there is an individual user knowledge model that stores a value that represents an estimation of the user knowledge level about the concept (i.e., Overlay Model). Therefore, every interaction a user makes with the system, such as visiting a page, solving a problem, or answering a quiz, is tracked and used to change the knowledge value of that user with respect to the involved concept(s) [Brusilovsky et al. 1996a, Brusilovsky et al. 1998] – Implicit user modeling as described in Chapter II.

Interbook sustains adaptive navigation support through utilizing adaptive annotation technique and direct guidance (explained in Chapter III). Each link is annotated with a colored ball next to it and with certain fonts with respect to users' knowledge level about the unit the link is pointing to:

- Red: content of a unit is not ready to be learned, as the user has not fulfilled all its prerequisites.
- Green: content of the unit is ready to be learned.
- White: content of a unit does not present any new information.

In addition, there is a checkmark, which marks the visited pages.

Following the same approach, Interbook distinguishes four different knowledge levels about presented concepts: unknown, known (learning started), learned, and well learned [Brusilovsky et al. 1998]. Each of these levels annotate the presented concepts in the domain model with respect to each individual user model as follows:

- No annotation: means unknown concept.
- Small checkmark: means known concept.
- Medium checkmark: means learned concept.
- Big checkmark: means well-learned concept.

On the other hand, direct guidance is provided through the TEACHME button, to be used by users who cannot make a choice for the next link to follow. Thus, the system employs several heuristics to select the most suitable node for a user to learn [Brusilovsky et al. 1996a].

Interbook provides a kind of help to users called prerequisite-based help. Within that kind of assistance, students who have problems with understanding some explanations, examples, or how to solve a problem, a list of links to all sections that present information about the prerequisite concepts of the current section is generated by the system when asked for. The links are sorted according to users' knowledge represented in the user model. The higher the link the more informative it is, i.e., links to sections that present information about an unknown concept are more informative than links to sections that present information about a known concept, and links to sections that present information about two unknown concepts are more informative than links to sections that present information about one unknown concept and so forth [Brusilovsky et al. 1996a]. Prerequisite-based help also supports the backward learning mode for users who have certain interests in a course and jump far ahead in the course until they reach the goal part of the material. In other words, provide them with the minimal knowledge required to understand the goal part [Brusilovsky et al. 1996a].

Interbook uses multiple windows and frames in its interface. The two main windows used by the system are the glossary window and the textbook window. In the glossary window, the upper part is a list of glossary concepts and the lower part is

used to show the glossary entry for each concept. On the other hand, the textbook window is the most important window in the Interbook interface, where the main content of a textbook is viewed. The textbook window is divided into frames, each performing a different task, such as Text Window that shows a particular section of the text book, and navigation centre which shows the position of the current section in the textbook.

Interbook helps authors to convert electronic textbooks to adaptive electronic textbooks. That kind of conversion happens through a series of steps [Brusilovsky et al. 1996a]:

- Creating textbook in a structured MS Word file, i.e., titles of high-level sections should have a pre-defined text style “Header1”, and titles of subsections are in a “Header2” text style, and so forth.
- Concept-based annotation of electronic textbook: this step helps Interbook to know which concepts are behind each section. The result of this step is an annotated structured MS Word file. An annotation is a piece of text with a special style and format inserted at the beginning of each section. Moreover, for each unit in the file the author has to define a set of prerequisite concepts and outcome concepts. The format of the outcome concepts is: (out: concept-name 1, concept-name 2, etc.), and the format of the prerequisite concept is: (pre: concept-name 1, concept-name 2, etc.).
- Translation to HTML. After annotation is finished the file is saved in RTF format, and then translated to HTML by means of the RTF2HTML program with special settings. The resulting file is an annotated HTML file, where the extension is altered manually to “.inter” so it can be recognized by the Interbook system.
- When the Interbook server starts it parses all Interbook files and builds a list of section frames. Each section holds the name and the type of the unit, its spectrum as well as its position in the HTML file.

The content that is presented to users is generated on the fly using knowledge about textbook, user model, and the extracted HTML fragments from the original HTML file.

Advanced authors who have some experience with HTML and LISP programming can skip steps 1 and 2 by preparing the textbook directly using the HTML format with annotations provided as especially formatted comments.

The web implementation of Interbook is based on the COMMON LISP Hypermedia Server, which is an HTTP server completely implemented in LISP.

4.2.5 Metadoc

The Metadoc system provides hypertext documents to potential readers in an adaptive way that suits their knowledge level. It is so called because it has knowledge about documents it presents [Boyle and Encarnacion 1994]. Metadoc provides its adaptive capability through the use of an interactive agent that stores knowledge about users in a user model, and that knowledge is used to vary the level of detail presented in the document for each user individually. Moreover, if a user decided explicitly to modify the level of the presented details into more or less, the user model is informed and future presentation of information may change. Thus, Metadoc provides automatic and manual control of the amount of the presented information [Boyle and Encarnacion 1994].

The system is used to present the content of two chapters of the technical manual 'Managing the AIX operating system' [Boyle and Encarnacion 1994]. According to that manual, Metadoc classified users with respect to their knowledge of Unix/AIX and general computer concepts into four classes: novice, beginner, intermediate, and experts. Furthermore, Unix/AIX concepts and general computer concepts are classified into different concept levels using the same scale. Thus, users' knowledge level about Unix/AIX and general computer concepts is independently stereotyped, i.e., a user may be a novice in Unix/AIX and a beginner in general computer concepts.

To present the correct level of information, the system varies the amount of explanation through the stretchtext technique as a way for providing contents level adaptation (explained in Chapter II). Thus, users who belong to a classification whose level is less than the difficulty of a given concept are assumed to be unfamiliar with that concept and extra explanation via stretchtext is provided. On the other hand, expert users/readers would rather want more in-depth detail than

explanations. Boyle [Boyle and Encarnacion 1994] has defined four types of stretchtext with respect to the positioning of the new text relative to the original one:

- Prefix: new text appears at the beginning of the original one.
- Embedded: new text becomes embedded within the original one.
- Appended: new text appears at the end.
- Replacement: new text completely replaces the old text.

The embedded type and the appended type are used with Metadoc. In Metadoc concepts are associated with stretchtext buttons to facilitate the comparison between the amount of information needed by a user and that presented by the system. Moreover, recursive stretchtext buttons are allowed, i.e., an embedded stretchtext button may contain an append button, and the append button may contain an embed and an append button, etc.; in addition, a stretchtext button may contain a link to a glossary node. Metadoc has default rules that control the depth of information presented to users, which are [Boyle and Encarnacion 1994]:

- Explanations to concepts associated with higher knowledge level are always presented to lower knowledge level users.
- Explanations to concepts associated with lower knowledge level are concealed from higher knowledge level users.
- Higher level details not necessary for understanding certain concepts are concealed from users with lower knowledge level.
- Details of equal or lower knowledge level concepts are presented to higher knowledge level users.

A user's actions in Metadoc are not overridden. For example, if a user clicked on a button for more detail, the detailed information associated with the button will always be shown to the user. In addition, other buttons associated with the same concept will follow the same behaviour, provided they have not been controlled by the user.

One of the components of the MetaDoc system is the intelligent agent component. This component functions as an assistant to the user in determining the correct level of information provided, as it matches the depth of the presented information to the

user model. In addition, it keeps track of a user's actions during the session to determine their correct knowledge level. The utilized user model is composed of:

- Long-term model: which keeps track of a user's knowledge state between sessions depending on short-term model and user's initial knowledge level.
- Short-term model: which maintains a user's immediate actions within a session.

In Metadoc two forms of user modelling are involved: explicit user modelling and implicit user modelling. In explicit user modelling, users are required to indicate their experience with computers before their first use, or they can use the system's default assumption about their experience. Based on that, two expertise levels, which correspond to a user's presumed knowledge of Unix/AIX and general computer concepts, are sustained for each user. In addition, users have the option to explicitly change the user model within sessions by specifying which concepts should be explained and which should be shown with more detail through stretchtext operations.

On the other hand, implicit user modelling is used throughout the session to upgrade the user model. Therefore, a request for an explanation or more detail about a concept indicates the lack of familiarity with that concept. In addition, jumping to the glossary for a definition gives the same indication as requesting further explanation in the case of stretchtext. In Metadoc, the first request for further explanation is not considered to indicate a lack of understanding about a particular concept, but any subsequent requests are taken as a sign that the user lacks familiarity with this area of study.

4.2.6 Hypadapter

Hypadapter [Hohl et al. 1996] is an adaptive hypertext system that supports exploratory learning and programming activities in the domain of COMMON LISP. Thus, typical users of that system are programmers with different programming skills in COMMON LISP. Through exploratory learning, users independently explore the hyperspace to discover information that is necessary for the solution to their current problem and acquire new domain knowledge by extending their understanding of new known subdomains. The system analyses users' navigational behaviour to deduce users' learning progress and to adapt the presentation of topics and links

while the user is navigating the hyperspace of topics. Users' programming skills in COMMON LISP provide the basis for systems' initiated adaptation.

Hypadapter classifies users according to their knowledge level about COMMON LISP into four stereotypes: novice, beginner, intermediate and expert, where each of those stereotypes masters topics to a certain difficulty level. Each user is assigned to only one stereotype based on the information already presented in the user model, which is based on a general application independent user modelling framework called MODUS [Schwab 1989]. For initial classification, users have to fill in personal questionnaires that reflect their expertise and their personal preferences concerning the presentation of learning material such as examples, additional notes, and links to related topics. Questionnaires' entries could be modified dynamically at any time to express changing needs regarding future adaptation. Moreover, the system provides defaults for entries that can be changed later by users (system is adaptable). For example, the system assigns users by default into novice stereotype unless they change that assumption, as some users may consider themselves intermediate rather than novice with COMMON LISP programming.

In Hypadapter, the domain model/knowledge base is composed of a network of topics, which are small chunks of information that indicate an understanding by a user and with independent self-meaning.

Topics are semi-structured entities defined with a set of attribute-value pairs. Every topic has general attributes that describe a topic's name, level of difficulty and links to prerequisite topics (which must be known before the current one), in addition to descriptive attributes such as definitions, examples, notes and summaries. Moreover, semantic relationships between topics are presented through link attributes that may refer to for example, superior topics, content-related topics, and sister topics. Furthermore, the knowledge base is composed of different types of topics such as concept-topics that represent programming concepts.

In Hypadapter, a topic's knowledge value ranges numerically between 0 (unknown) to 10 (well-known). Each time a user accesses a topic its knowledge value about it increments. Thus, the user is re-classified to a higher level of expertise after reaching

a certain knowledge level in a certain number of topics that differs from stereotypical level to another, i.e., the system increases the knowledge value of any visited topic. Moreover, Hypadapter gives users the option to customize certain aspects in their models through user model inspector by invoking a personal questionnaire through it. Therefore, the system is integrating implicit and explicit methods of user modeling as explained in Chapter II.

One of the components of the architecture of the Hypadapter system, in addition to the user model component and the knowledge base component, is the evaluation component. That component identifies topic attributes and links that are relevant to a user by applying selection rules. Selection rules enclose a teaching model that is based on a scoring mechanism to determine those attributes and links that are mostly relevant to the user's needs and interest. The selection rules for evaluating descriptive attributes are:

- Contents of attributes should correspond to the user's stereotypical level. For example, explanations should be tailored to the user's current classification, ranging from novice to expert.
- A user's learning style and behaviour are considered. For example, in presentation, curious users need more information.
- A user's preferences regarding certain attributes such as examples and notes are taken into consideration.
- Different levels of detail such as in-depth description or short summaries should be provided according to the user's preferences.

On the other hands, selection rules for link attributes to other topics are:

- Avoiding mastered topics.
- Avoiding topics that require a pre-knowledge a user does not yet have.
- Topics that are needed immediately are preferred.
- Topics that are related to known topics are preferred.
- A topic's level of difficulty should match the user's qualification. Thus, for example, beginners should not access topics with a difficulty level that matches expert users.
- Topics a user declared as bookmarks are preferred as they might be useful in the future.

The goal of these rules is to protect users from information that is not relevant to their expertise level, and to support the incremental expansion of their knowledge by topics that are suitable to them. Hence, according to these rules, links level adaptation is provided in two forms: a) hiding - hiding links to topics that are not relevant at all to the users' current knowledge state; b) sorting - relevant links are sorted according to their priority with respect to users' knowledge in descending order (from most relevant to less relevant); c) annotation – the most relevant sorted link is annotated in a different font size than the next relative link, which in turn will have a different font size than the following one and so forth.

Furthermore, on the contents level adaptation, the Hypadapter system uses a frame-based technique (explained in Chapter III), where slots of a frame that hold different information related to a certain concept are sorted according to users' knowledge level and characteristics, such as preferred learning strategy and preferences.

Therefore, when a topic is presented to a user the system dynamically determines an appropriate subset of attributes to be displayed depending on the user's characteristics. In addition, it suggests links to topics that might be relevant or interesting for users according to their current expertise level. For example, there are two users, A, beginner in Lisp, and user B, advanced in Lisp, both of whom are accessing the same topic. Thus, for user A, the contents of the descriptive attributes such as notes and examples, will be for beginner users. On the other hand, user B will be restricted to a short summary, expert level examples, and links to external information resources. In both cases, non-relevant information will be hidden behind an icon that can be expanded on demand.

The Hypadapter system provides a graphical user interface to help users to access and explore the knowledge base, where navigation by link-based browsing is the main way to access this information. Through this interface users can bookmark topics that are already visited or think might become important in the near future. In addition, topics are provided by alphabetically-sorted indices that facilitate direct access to them. Moreover, users can revisit previously visited topics through topic histories that record presentations of topics visited before. Furthermore, structure-

oriented access to topics is presented via knowledge maps that provide graphical overviews of parts of the underlying knowledge base structure. In addition, access to topics known rather by name is provided by a command line interface

4.2.7 CHEOPS

The CHEOPS [Ferrandino et al. 1997, Negro et al. 1998] system is developed at Dipartimento di informatica ed Applicazioni of the University of Salerno, Italy. That system is a server-based implementation of a session-based interaction model.

The CHEOPS system represents its knowledge model as a knowledge pyramid. That pyramid is composed of two polygons with different sizes, where the smaller one is at the top, which represents the minimum knowledge level, while the bigger one, which represents the maximum knowledge level, is at the bottom of the pyramid. Users' knowledge levels are classified into three stereotypes: novice, amateur, and expert. The edges of the pyramid represent these knowledge levels with respect to each category, as hyperdocuments in CHEOPS are divided into categories to help users in navigation though the presented information, where vertices of the knowledge pyramid represent these categories. Thus, the experience gained by a user could be represented as the surface obtained cut in the intersection of the current experience level on each category. Moreover, categories could be dependent on each other, which means, according to a user's knowledge level in some categories, another category/categories could be accessible to that user. Therefore, the system provides links level adaptation through using link removal technique (explained in Chapter III). Furthermore, users' expertise level for each category depends on their previous interaction within each category; in addition, they have the ability to alter their own knowledge level for each category, i.e., the system is adaptable – users could change the adaptation effect by altering their knowledge state about the presented concepts. Therefore, the system integrates between implicit and explicit user modelling techniques (explained in Chapter II) to obtain users' knowledge level with respect to the presented material.

In this system, the adaptation mechanism takes the required document as an input, and creates on the fly a document that has all the links changed according to the user's profile, which is stored in a database and contains information about the

current levels of expertise the user has reached during his/her interaction with each category. Thus, a user's profile is modified each time the user chooses a link. Furthermore, this system allows its users to annotate every visited page, by evaluating it as "Interesting", "Not Interesting", or "Very Interesting", in addition to a one-line comment, which could be considered as a user defined annotation to the visited page.

In CHEOPS, to provide adaptivity to hyperdocuments, the author/authors of every hyperdocument have to classify hyperdocument files into categories with knowledge levels, and to fix the visibility of some categories. Also, he/she has to provide the system with all information related to the hyperdocument, such as its directory, names of the categories, and so on. In addition, the author has to build the category summaries that contain links to all documents at the same level for a given category. Afterwards, he/she has to specify an entry point to the hyperdocument.

4.2.8 TANGOW

TANGOW (Task-based Adaptive learNer Guidance On the WWW) is a tool for creating internet-based courses, where courses are described as a set of teaching tasks and rules [Carro et al. 1999, Carro et al. 2000]. A teaching task is the basic unit that appears in the learning process. In other words, teaching tasks correspond to concepts presented within a course. For example, if an HTML course holds three major concepts: "introduction", "creating HTML pages", and "advanced HTML issues", thus, each teaching task corresponds to one of these concepts. Moreover, any of these concepts may hold subconcepts. For example, the "creating HTML pages" concept may hold five other subconcepts. Thus, each teaching task corresponds to each concept and each subtask of that task corresponds to each subconcept of that concept. According to that description, each teaching task may be atomic, in case it does not have subtasks, or composed in case it has subtasks. Furthermore, each teaching task has a type, as it may be theoretical, practical or a set of examples. Also, it may have a set of associated media elements. On the other hand, rules play an important role in curriculum sequencing and adaptation. These rules describe how teaching tasks are de-composed, in case they are of a composed type. There may be several rules for the same teaching task, each of them representing a specific way of

de-composing the teaching task into subtasks. It may be necessary to perform these subtasks in a fixed order, in any order, or it may be enough to perform some of them to accomplish the composed task itself. Moreover, each rule specifies the prerequisites for the associated task, which depends on a user's profile and information about achieved tasks as well as the learning strategy in use. Hence, knowledge in TANGOW is represented by means of teaching tasks that need to be accomplished.

For each student there is a task manager that takes control over the learning process during the whole session. In addition, if a user is following more than one course there will be a task manager for every course the user is involved in. The task manager provides guidance to students by deciding the next set of tasks to be accomplished depending on the chosen leaning strategies (such as theory presentation first or practical exercises first), student personal data, and previous performed actions. Moreover, actions performed by a user, such as the number of visited pages, the number of finished exercises, and the number of exercises solved correctly, are stored by means of the task manager.

When a user accesses the system, he/she is asked to identify him/herself through supplying a user name and password, and then selects the course that he/she intends to access. If the user is accessing the system for the first time he/she will be asked to supply their preferred learning strategy, for example, presenting the theoretical part of the topic first and then examples, his/her preferred language and personal data, such as age. However, if he/she is not a new user, the system will restore the user's profile from the database. It is important to clarify that the same scenario is used in WHURLE-HM.

During the learning session, the task manager constructs a list of tasks to be accomplished that presented to the student that fit his/her educational state from a menu page. Thus, tasks for which a user has not fulfilled prerequisites, such as accomplishing other tasks before it, will not be accessible. Two other adaptation functionalities the TANGOW system performs:

- HTML pages are built on the fly, as they are composed of different media, such as text, animations, etc., that differ from one user to another depending on his/her profile, through what is called page generator.

- According to the selected learning strategy, for example, such as choosing examples first and then the theoretical parts of the course has an effect on the decomposition of tasks. Thus, decomposition of tasks depends on the user profile. For example, decomposition may include subtasks related to theory and exercises while another one may include additional subtasks with examples.

It could be figured that the system is providing adaptive navigation support through:

- a) link disabling – a link to a certain task will be disabled if a user has not fulfilled its prerequisites, such as a minimum number of exercise about particular task should be achieved before accessing the current one, or if the user has accomplished the task;
- b) sorting – tasks that have not been finished by the user come first and those that the user has not fulfilled their prerequisites or has accomplished (disabled links) come to the end.

Furthermore, the system provides adaptive presentation support through explanation variant, as according to the preferred language, learning style, etc. the material will be presented adaptively. Moreover, the systems implicitly deduce users' knowledge through accomplished tasks (implicit user modelling).

The general description of all teaching tasks that have been defined by a course author are stored in the teaching tasks repository database. In addition, media elements that appear in HTML pages and classified according to characteristics defining students' profiles such as language and age are stored in the course content database. In addition, users' profiles that include information about their age, preferred language, and preferred learning strategy are stored in the user database.

4.3 Systems analysis

By analysing the above systems from two angles: adaptation techniques and user modelling methods, the following could be obtained:

- Figure 4.2 classifies the systems according to the used adaptation techniques as follows:

TANGOW	Hypadapter	CHEOPS	MetaDoc	Interbook	NetCoach	ELM-ART	AHA!	Frame Based Technique
-	X	-	-	-	-	-	-	Conditional Text Technique
-	-	-	-	-	-	-	-	Explanation Variant
X	-	-	-	-	-	-	-	Direct guidance
-	-	-	-	X	X	X	-	Annotation
-	X	-	-	X	X	X	X	Hiding
-	X	-	-	-	-	-	-	Removing
-	-	X	-	-	-	-	-	Disabling
X	-	-	-	-	-	-	-	Sorting
X	X	-	-	-	-	X	-	Stretch text
-	-	-	X	-	-	-	-	

Figure 4.2 used adaptation technique - from the above table it could be perceived that AHA! system used the conditional text technique to provide contents level adaptation. Moreover, it used Annotation, hiding, removing to provide adaptive navigational support (links level adaptation). ELM-ART, NetCoach, Interbook systems used Direct guidance and annotation to provide adaptation on the links level, also sorting has been used by a previous version of ELM-ART as explained before. Furthermore, MetaDoc system used stretch text technique to provide adaptation on the links level as well. CHEOPS system used link removal. Hypadapter used frame-based technique to provide adaptation on the contents level and annotation, hiding and sorting for adaptive navigational support. On the other hand, TANGOW implemented disabling, sorting and explanation variant to provide suitable adaptation for its users.

From the above table it could be perceived that the explained systems integrated different adaptation techniques and methods to provide their users with a kind of adaptation that suit their knowledge level, learning style, goals, etc. for example, AHA! implemented link hiding, annotation and removing to supply its users with personalization on the links level, while conditional text technique is used to provide adaptation of contents. On the other hand, systems like ELM-ART, Interbook and NetCoach used direct guidance technique beside annotation.

Moreover, ELM-ART used sorting techniques. Furthermore, MetaDoc system used stretch text techniques to provide its users with adaptation on contents level. Hypadapter system used sorting, annotation, hiding and frame-based techniques. Moreover, TANGOW system used disabling and sorting techniques for link adaptation, while explanation variant method is used to provide its users with a kind of personalization that suit their preferred language and learning style. What is more, CHEOPS system used link removing to hide from its users links to information that does not suit their knowledge at that time.

- By analysing the mentioned systems from the user modelling perspective (using explicit and implicit user modelling techniques) the following results could be obtained as shown in Figure 4.3:

	Explicit	Implicit
AHA!	X	X
ELAM-ART	-	X
NetCoach	X	X
InterBook	-	X
Metadoc	X	X
CHEOPS	X	X
Hypadapter	X	X
TANGOW	X	X

Figure 4.3 Used user modelling methods - From the above table it could be seen AHA!, NetCoach, etc. combined implicit and explicit user modelling methods. On the other hand, ELM-ART and InterBook utilised implicit user modelling methods.

From the above table it could be perceived that the majority of the explained systems integrated explicit and implicit methods to obtain information about their users. By summarising the observed results the following could be obtained:

- a) AHA!: the system monitors the navigation of users through the course material and uses this information to infer their knowledge levels implicitly. Furthermore, users could change their knowledge state about specific concepts explicitly, if the course authors make the decision to allow this. In addition, users can choose between link hiding and link annotations.
- b) NetCoach: in this system the educational state of presented concepts is considered to be learned by users if it is implicitly deduced by the system through tests, if it is inferred from other concepts, or if it is specified by the user explicitly.

- c) Metadoc: the system implicitly infers users' knowledge about presented concepts if they subsequently ask for more explanation regarding a particular concept by means of either stretch text or by jumping to the glossary for a definition. Users can also explicitly specify their experience with respect to the knowledge domain the first time they use the system.
- d) Hypadapter: users in that system can change the assumptions that the system makes about their knowledge level with respect to the presented information. Moreover, users can decide their preferred learning strategy explicitly. From the implicit perspective, the system builds its assumptions about users' knowledge state by monitoring their navigational behaviour.
- e) CHEOPS: users can change the system's assumptions regarding their knowledge about the presented categories, as the educational material is divided into different categories (as described in Chapter IV). In addition to this, the system can infer their knowledge about the presented categories through monitoring their navigational behaviour.
- f) TANGOW: users in that system can explicitly set their preferences for both language and learning strategy. Furthermore, the system implicitly infers users' knowledge state through the accomplished teaching tasks (as described in Chapter IV).

With the other two systems, their user model is implicit. These are:

- a) ELM-ART: the system decides for a particular concept(s) to be known by the user if he/she solved the associated quiz or a programming problem correctly.
- b) InterBook: follows the same approach as ELM-ART by inferring users' knowledge about concepts through monitoring pages that users visited and by means of the quizzes and problems that they solved.

4.4 Summary

Through this chapter, an overview has been given about different adaptive hypermedia systems. Moreover, focus has been placed on adaptive educational

hypermedia systems by illustrating different popular examples and discussing each of them from two perspectives: adaptation techniques and user modelling.

The chapter is structured as follows:

- **Introduction:** an attempt was given at a brief overview about the history of research in adaptive hypermedia. Moreover, different categories of adaptive hypermedia systems are explained.
- **Adaptive educational hypermedia:** that section is the main focus of the chapter, through which different kinds of adaptive educational hypermedia systems have been illustrated, and the improvements that happened in each have been discussed. A comparison between the described systems and between the WHURLE-HM in addition to the shortage that these systems have in terms of measuring users' knowledge is discussed in Chapter IX
- **Systems analysis:** in this section a full analysis about the utilised adaptation and user modelling techniques employed by the systems explained in the former section is given.

Chapter V: Technology

5.1 Introduction

In the former chapters an overview about user modelling and adaptation techniques is given. Furthermore, a survey concerning adaptive educational hypermedia systems in addition to their analysis is provided. This chapter gives an idea about the technology used to implement the Hybrid model in the WHURLE system to produce WHURLE-HM, such as XML (eXtensible Mark-up Language), XSP (eXtensible Server Page), the Cocoon publishing framework, etc. Moreover, the chapter will clarify to the reader the technical expressions used in the implementation chapter (Chapter VII), in addition to the system's code in Appendix A.

5.2 XML and HTML

Hypertext Markup Language (HTML) [Berners-Lee 1992] provides a standardized way to create pages of formatted information that can be delivered to a global audience via the Internet. HTML is mainly designed for formatting documents for the web, as it consists of a set of elements that define how the data is to be displayed.

The following example represents a phone diary document written in HTML.

```
<H1>Mohamed Ramzy</H1>
<br>
<l>mrz@cs.nott.ac.uk</l>
<br>
<l>0998876</l>
```

It can be seen from the above example that the elements do not describe the data they enclose. All the text in the document can be changed and can lose the fact that this was originally a phone diary document. This is done because HTML is only concerned about describing text formatting and layout. Authors of HTML are limited to a particular set of elements and if this set does not meet a need, they have to either find a way to work around or live with the inadequacy. HTML documents might not be portable to other applications, because its only purpose is to be displayed over the web, and that creates extra work for authors who must use several languages to accommodate different applications.

On other hand, eXtensible Markup Language (XML)¹ is an open, text-based markup language that provides structural and semantic information about hyperdocument rather than the content itself [Pardi 1999, Gilchrist 2002]. XML is a powerful and standard-based complement to HTML that could be as important to the future of information delivery on the web as HTML was to its beginning. XML is a subset of a popular Standard Generalized Markup Language (SGML), from which HTML is created.

XML is a *meta-language*, i.e. a language used to describe another language. Thus, XML authors can create elements that describe the text they contain as in the following example:

```
<Phone>
  <first-name> Mohamed </first-name>
  <last-name> Ramzy </last-name>
  <email> mrz@cs.nott.ac.uk </email>
  <ph-num> 07947552</ph-num>
</Phone>
```

In the above example, a phone diary written in XML, the elements describe the texts they include, not as in HTML, which just describes the layout. XML is more a tool for creating documents structures than for applying those structures to a particular interface.

The primary factor of designing of XML is to work well on the web, but XML is intended to work in many environments outside the web as well, including data interchange, commercial applications and publishing. In XML, the formatting of data on a page is achieved by adding a style sheet to the document (as described later). This leads to the significant advantage of XML over HTML; it distinguishes between the contents and the style. XML is ideal for large complex documents, such as a thesis, because that data is structured. Moreover, it is not only specifying a vocabulary that defines the elements in the documents, it also establishes the relation between elements; this is because at its heart is a hierarchical system. In the hierarchical system, the elements are in the form of a tree, the nodes are in relation to each other, as parents and children. For example, by considering the following XML document about a phone diary:

¹ [http:// www.w3c.org/XML](http://www.w3c.org/XML)


```

<? xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="Phone.xsl"?>
<Phone-diary>
  <Friend>
    <Name>John</Name>
    <Phone>12345</Phone>
    <Location>UK</Location>
  </Friend>
  <Colleague>
    <Name>Sabstian</Name>
    <Phone>54321</Phone>
    <Location>France</Location>
  </Colleague>
</Phone-diary>

```

By representing the above code as a tree of node the following diagram in Figure 5.1 is obtained:

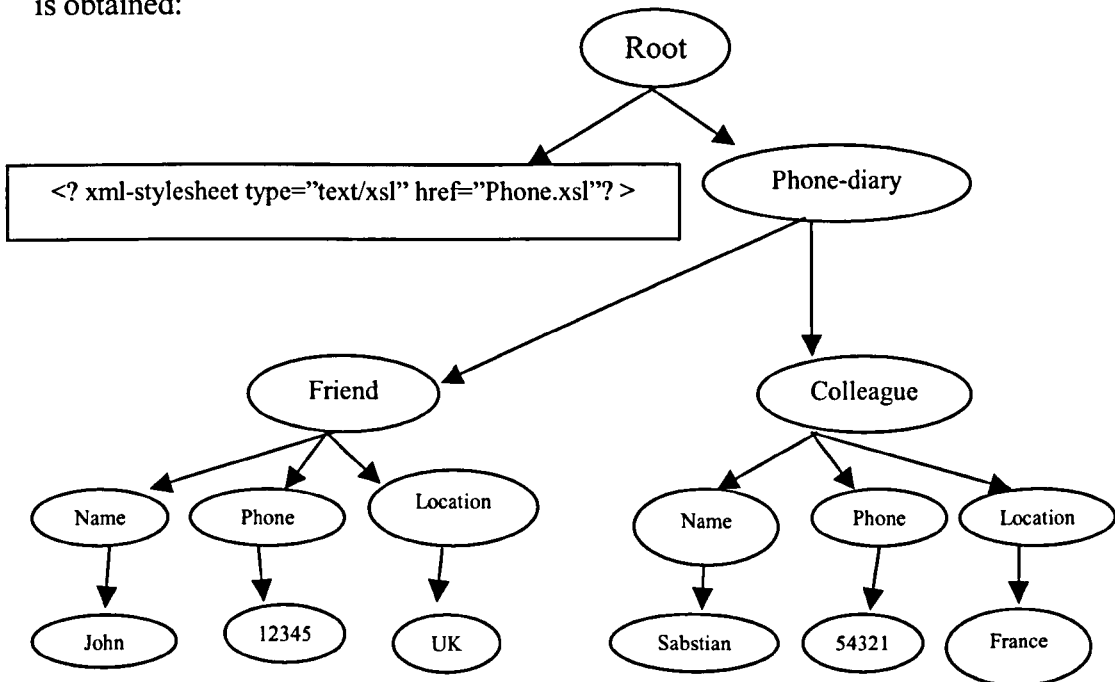


Figure 5.1: Tree Diagram

To understand the XML structure well it is necessary to know what is happening to any XML document since it is written until it reaches users browsers, i.e. XML document's life cycle. The life cycle of XML document, as shown in Figure 5.2, starts when an editor creates it. After that, the XML parser (XML processor) reads the document and converts it into a tree of elements. The parser passes the tree to the browser, which displays it to the end user

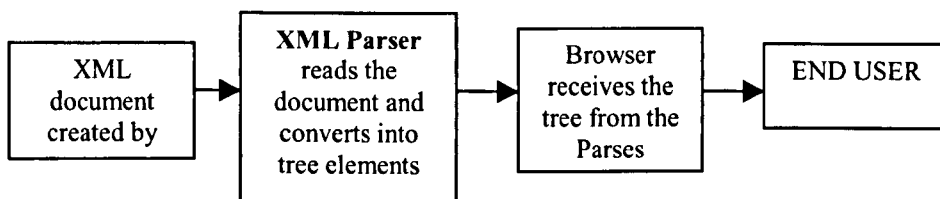


Figure 5.2- XML Document life cycle

As XML describes the structure of a document, this structure may have rules. A DTD² (Document Type Definition) specifies a set of rules for the structure of a document. For example, a DTD determines that a book element has exactly one ISBN child and exactly one title as well as one or more author children, and it may or may not contain a single subtitle. The DTD accomplishes this with a list of markup declarations for particular element, attributes, entities and notations. DTD could be included in the file that contains the document it describes, or it can be linked from an external URL. Different documents and web sites can share such an external DTD. Furthermore, a DTD shows how different elements of a page are arranged without actually providing their data. The following example shows how DTD could be written:

```

<!ELEMENT INVOICENO (#PCDATA)>
<!ELEMENT PRODUCTID (#PCDATA)>
  
```

The above example means that the two elements INOVICENO and PRODUCTID will hold just normal text data, as they are not attributes or entities. In fact, the use of a DTD has some drawbacks. Although DTD served SGML developers for a long time, it has some severe restrictions. DTDs call for elements to consist of one of three things: a text string, a text string with other child element mixed together or a set of child elements. DTD does not have XML syntax and offers only limited support for types or name space. For these reasons XML schema³ is considered, which has XML syntax and defines a set of new names such as the names of elements, types, attributes and attribute group. For more information about XML Schema refer to <http://www.w3.org/XML/Schema>.

² <http://www.w3c.org/TR/REC-xml#dt-doctype>

³ <http://www.w3c.org/XML/Schema>

5.3 Related Technology

XML does not operate by itself; to use it as more than a data format it requires interaction with a number of related technologies. An overview about these technologies is given in the following subsections.

5.3.1 Cascading Style Sheets (CSS)

CSS⁴ were introduced in 1996 as a standard means of adding information about style properties such as fonts and borders to HTML documents [Harold 1999]. CSS actually works better with XML than with HTML and that is because HTML is loaded with backward compatibility between the CSS rules and the HTML elements. Because XML elements do not have any predefined formatting, they do not restrict which CSS styles can be applied to which elements. A CSS style sheet is a list of rules. Each rule defines an element(s) and the style(s) that control the appearance of that element(s). Multiple style sheets can be applied to a single document, and multiple styles can be applied to a single element. An example of a CSS is:

```
POEM {DISPLAY: BLOCK}
TITLE {display: block; font-size: 16 pt; font-weight: bold}
POET {display: block; margin-bottom: 10 px}
```

This style sheet has three rules. Each rule has a selector - the name of the element, to which it applies - and a list of properties to apply to the instance of that element. The first rule says that the contents of the POEM should be displayed in a block by itself. The second rule, says that the contents of the TITLE also should be displayed in a block by itself with a font size 16 pt (points) and font weight bold. The third rule says that the contents of the POET should be displayed in a block by itself and should be set off from what follows it by 10 px (pixels).

In 1998, the W3C (World Wide Web Consortium)⁵ published a revised and expanded specification for CSS called CSS Level 2 (CSS2). At the same time, they renamed the original CSS to CSS Level 1 (CSS1). CSS2 is mostly a superset of CSS1 with a few minor exceptions; so CSS2 is CSS1 plus aural style sheets, media types, attribute selectors and other new features [Harold 1999].

⁴ <http://www.w3c.org/Style/CSS/>

⁵ <http://www.w3c.org>

5.3.2 eXtensible Style Sheet (XSL)

XSL⁶ is itself an XML application. XSL is intended to define the formatting and presentation of XML documents for display on screen, on paper, or in the spoken word. With the development of XSL, it became clear that this was usually a two-stage process. The first stage is a structural transformation in which elements are selected, grouped and reordered. The second stage is a formatting process, in which resulting elements are rendered. It was recognized that these two stages were quite independent, so XSL was split into two parts, *XSLT (eXtensible Style Sheet: Transformation)* and *XSL-FO (eXtensible Style Sheet: Formatting Object)* [Harold 1999].

5.3.2.1 eXtensible Style Sheet: Transformation (XSLT)

XSLT⁷ is primarily designed for transforming one XML document into another. It is more than capable of transforming XML to HTML and many other text-based formats, so in more general definition:

“XSLT is a language for transforming the structure of an XML document” [Kay 2000].

More precisely, an XSLT accepts as input a tree represented as an XML document and produces as output a new tree, also represented as an XML document. XSLT cannot be used to transform to or from non-XML formats like PDF, TeX, Microsoft Word or others, as XSLT uses the structure of the XML node tree. The transformation part of XSL is also called the tree construction part, as both the input and the output must be XML documents.

HTML and SGML are borderline cases because they are so close to XML, so XSLT can be used to transform to or from HTML and SGML if they meet XML's well-formedness rules (rules in the Document Type Definition (DTD), will be explained later). An XSLT document contains a list of template rules and other rules. A template rule has a pattern specifying the trees it applies to and a template to be output when the pattern is matched. When an XSL processor formats an XML document using an XSL style sheet, it scans the XML document tree looking through

⁶ <http://www.w3c.org/Style/XSL>

⁷ <http://www.w3c.org/TR/xslt>

each sub-tree in turn. As each tree in the XML document is read, the processor compares it with the pattern of each template rule in the style sheet. When the processor finds a tree that matches a template rule's pattern, it outputs the rule's template. This template generally includes some mark up, some new data and some data copied out of the tree from the original XML document. For example, by applying the following XSLT (Phone.xsl) to Phone-diary XML document.

The following is the XSLT style sheet, which specifies the look of the output tree.

```
<? xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl=http://www.w3.org/XSL/Transform/1.0>
<xsl:template match ="Phone-diary">
  <HTML>
  <body>
    <xsl:apply-templates/>
  </HTML>
</xsl:template>
<xsl:template match="Friend">
  <P>
    <xsl:apply-templates/>
  </P>
</xsl:template>
<xsl:template match="Colleague">
  <P>
    <xsl:apply-templates/>
  </P>
</xsl:template>
</body>
</HTML>
</xsl:stylesheet>
```

After applying that former stylesheet, the output of the XML document will be as follows in HTML format:

```
<HTML>
  <body>
    <p>John 12345 UK</p>
    <p>Sabstian 54321 France</p>
  </body>
</HTML>
```

It can be observed in the second line that the XML document (*href="Phone.xsl"*) calls the XSL style sheet. This XSL style sheet is very simple with two template rules. The first one matches the root element Phone-diary. It replaces this element with an HTML element. The contents of the HTML are the results of applying the other templates in the document to the contents of the Phone-diary element.

5.3.2.2 eXtensible Style Sheet: Formatting Object (XSL-FO)

XSL formatting objects provides a more sophisticated visual layout than HTML + CSS, even CSS2. Formatting supported by XSL formatting objects but not supported by HTML + CSS includes a non-western layout, footnotes, margin notes, page numbers in cross-references and more. In particular, while CSS is primarily intended for use on the web, XSL formatting objects are designed for more general usage. XSL formatting objects are a complete XML vocabulary used to arrange elements on a page. A document that uses XSL formatting objects is simply a well-formed XML document. That means it has an XML declaration, a root element, child element and so forth. It must stick to all the well-formedness rules of any XML document or formatters will not accept it. As an example of XSL formatting objects, the following XSL style sheet is an XSL formatting object document for the previous XML document to display it in a PDF form.

```
<fo:root xmlns:fo="http://www.w3.org/xsl/format/1.0">

<fo:layout-master-set>
<fo:simple-page-master page-master-name="only">
  <fo:reigon-body/>
</fo:simple-page-master>
</fo:layout-master-set>

<fo:page-sequence>
<fo:sequence-specification>
  <fo:sequence-specifier-single page-master-name="only">
</fo:sequence-specification>
<fo:flow>
  <fo:block font-size="20 pt" font-family="serief">
    hydrogen
  </fo:block>
  <fo:block font-size="20 pt" font-family="serief">
    helium
  </fo:block>
</fo:flow>
</fo:page-sequence>
</fo:root>
```

The above XSL document is a simple document using XSL formatting objects. The root of the document is a FO:ROOT. This element contains a FO:LAYOUT-MASTER-SET and a FO:PAGE-SEQUENCE. The FO:LAYOUT-MASTER-SET element contains the FO:SIMPLE-PAGE-MASTER child element. Each FO:SIMPLE-PAGE-MASTER describes a kind of page on which the content will be placed, the content is placed on copies of the master page using a FO:PAGE-SEQUENCE. The FO:PAGE-SEQUENCE contains a FO:SEQUENCE-

SPECIFICATION specifying the order in which the different master pages should be used. Next, it contains a FO:FLOW child that holds the actual content to be placed on master pages in the specified sequence. The content here is given a font-size and a font type by specifying the FONT-SIZE attribute and the FONT-FAMILY attribute in the two FO:BLOCK child elements. This is a very simple XSL formatting object document, however, documents that are more complex can have different master pages for first page, right and left, body pages, etc.

5.3.3 XML languages

In the following subsections, a brief overview about XML languages such as XML Path language (XPath), XML Linking language (XLink), and XML Pointer language (XPointer) will be given. Each of these languages gives XML more power and flexibility. For example, XML goes further than HTML link elements can go, as it uses XLink for linking to documents and XPointers for addressing individual parts of a document.

5.3.3.1 XPath

XPath⁸ was designed to be utilized by XSLT⁸ and XPointer⁹. The primary purpose of XPath is to locate different parts of an XML document. Furthermore, it facilitates manipulation of different data types such as strings, numbers and Booleans. Moreover, it uses non-XML syntax to ease its use within URIs (Uniform Resource Identifier) and XML attributes. XPath uses path notation to navigate through the hierarchical structure of an XML document. Thus, it is used to direct an XSLT processor when navigating through the hierarchal node tree of an XML source document to produce desired nodes in the output tree [Cagle et al. 2001].

Any XPath expression in its simplest form consists of:

- An axis: which specifies how to find the target element(s) by moving along one of the thirteen axes that XPath has (child, parent, descendant, ancestor, descendant-self, ancestor-self, following-siblings, preceding-siblings, following, preceding, attribute, namespace, self). Each of those axes specifies the relative location of the context node (current node) with respect to other node(s) in the node tree.

⁸ <http://www.w3c.org/TR/xpath>

- A node-test: it is used to check that the selected node is of the type specified. For example, if a child node is to be selected, the node-test checks the selected node to ascertain if it is of a child type with respect to the context node or not.
- A predicate: this operation is optional, and it is used to refine the selected nodes on the basis of their relative position or on the basis of some features they have such as a certain attribute or value.

Recently W3C has released a specifications draft for XPath V2, which works with XSLT V2. For more information about changes in XPath V2, refer to:

<http://www.w3.org/TR/2002/WD-xslt20-20021115/Overview-diff.html#XPATH20>.

5.3.3.2 XLINK

XLink¹⁰ enables any element to become a link, not just an element as in HTML, as the links could be bi-directional, multidirectional or even point to multiple mirror sites from which the nearest is selected. Elements that include links are called *Linking Elements* (anchors in HTML). Linking elements are identified by an xlink:type attribute with either the value *simple* or *extended*.

Simple XLinks are similar to standard HTML links and are likely to be supported by application software. On the other hand, Extended links go beyond what HTML can do, as it is possible with extended links to include multidirectional links between many documents. One of the capabilities of extended links is to point to more than one target; such a feature cannot be executed with HTML links.

XLink has the capability to store the links in a separate linking document. This might be useful to maintain a slide show where each slide requires next and previous links. By changing the order of the slides in the linking document, the targets of the previous and next links on each page could be changed without having to edit the slides themselves. XLink allows much more sophisticated connections between documents with XPointer.

⁹ <http://www.w3c.org/XML/Linking>

¹⁰ <http://www.w3c.org/TR/xlink/>

5.3.3 XPOINTER

XPointer is designed to be used as the basis for a fragment identifier for any URI reference (which is not possible with XPath) that locates a resource with any of the following media :

- text/xml.
- application/xml.
- text/xml-external-parsed-entity.
- application/xml-external-parsed-entity.

XPointer is based on XPath. Therefore, it could scan the hierarchical structure of an XML document to choose some of its internal parts based on different characteristics such as attribute values and relative position . Thus, XPointer can refer to a particular element of a document such as the first, second, or any such element and so on.

5.3.4 XInclude

By XInclude¹¹ a number of XML documents could be merged into a single composite XML document. XInclude differs from the linking features in the XML Linking Language (XLink), as XLink does not specify a specific processing model, but simply facilitate the detection of links and recognition of associated metadata by higher-level application. On the other hand, XInclude specifies a media-type specific transformation. It defines a specific processing model for merging information sets, as the processing occurs at a low level, often by a generic XInclude processor which makes the resulting information set available to higher level applications. The following is a simple example of XInclude:

```
<x xmlns:xinclude="http://www.apache.org/1999/XML/Xinclude">
  <xinclude:include href="1stpg.xml"/>
  <xinclude:include href="2ndpg.xml"/>
</x>
```

In the above example, the XML document 1stpg.xml will be replaced by its contents as well as the second one (2ndpg.xml).

WHURLE and WHURLE-HM used such technology to include chunks involved in a lesson plan when requested

5.4 Cocoon Publishing Framework

Having hundreds of XML documents on a site does no good if there is no mechanism to apply transformations on them when requested. The problem is that an engine must exist to handle this generation, particularly in a dynamic sense. Just as a web-server is responsible for responding to a URL request for a file, a web-publishing framework is responsible for responding to a similar request; however, instead of responding with a file, it often will respond with a published version of the file. In case of XML, a published file refers to a file that may have been transformed with XSL, massaged at an application level or converted into another format such as PDF. The requester does not see the raw data that may underlie the published result, but also does not have to explicitly request that the publication occur. Often a URI base signifies that a publishing engine that sits on top of a web-server should handle requests. To choose a publishing framework, many points should be in consideration as it has to serve clients on any platform and it must not be tied to a specific parser or processor. The Java language in particular, offers such an easy interface into XML/XSL transformation, and the other needed APIs. In addition, Java servlets offer such a simple means of handling web requests and responses.

A Java servlet is a generic server extension, which means that a Java class can be loaded dynamically to expand the functionality of a server [Hunter and Crawford 1998].

Servlets are commonly used with web servers, where they can take the place of CGI (Common Gateway Interface) scripts. A servlet is similar to a proprietary server extension, except that it runs inside a Java Virtual Machine (JVM) on a server, so it is safe and portable. Unlike CGI, which uses multiple processes to handle separate programs and/or separate requests, servlets are handled by separate threads within the web server process. Because servlets run within the web server, they can interact very closely with the server to do things that are not possible with CGI scripts. As mentioned before, as servlets are portable, they are used both across operating systems and across web servers.

¹¹ <http://www.w3c.org/TR/XInclude/>

According to the advantages of servlets and the two main points in choosing a publishing web server, the Cocoon¹² publishing framework from APACHE software foundation will be considered as an example. Cocoon is a Servlet-based open-source engine currently under heavy development and with a rapidly maturing feature set. This includes database drivers and support for many of the developing associated XML standards such as XLink and XPointer. Figure 5.3 shows how an interaction between a web server and the Cocoon publishing framework is happening:

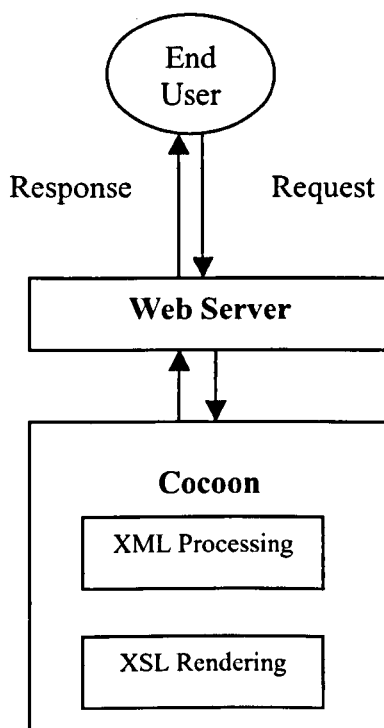


Figure 5.3 A Web server and Cocoon interaction – when an end user sends a request to a Web Server and this request is an XML file (for example), the Web server directs the request to the Cocoon publishing framework. Therefore, the requested XML document will be processed - if it contains logic sheets (XSP, ESQ, etc.). After that, the document will be formatted to a specific type (HTML, PDF, etc.) by applying an XSL style sheet to it – XSL Rendering. After the XML processing and the XSL rendering, the published version of the file will be directed to the Web server which will return to the end user.

From the above figure it could be perceived that the Cocoon framework after receiving a request from the web server does two steps: a) processing any logic sheet in the document such as XSP and ESQ (described below), b) rendering the document to a specific type for publishing such as PDF, HTML, etc. by applying an XSL stylesheet to it. After these processes, Cocoon responds to the web server with

¹² <http://cocoon.apache.org/1.x/>

the published version of the XML document and consequently the web server sends it back to the end user.

Cocoon is now in its second generation (Cocoon 2)¹³ as an XML publishing framework and it is completely based on JAVA, also it allows any conformant XML parser to be used. Moreover, Cocoon 2 relies on a pipeline model, this model acts as the servlets chain concept. In this concept the output of one servlet class could be the input for the another. There are other alternatives for Cocoon such as Websphere from IBM. This publishing framework is mainly commercial and not such an open source as Cocoon; in addition, Cocoon supports more technologies than Websphere has supported recently. The following sections give an overview about technology supported by Cocoon. Moreover, Cocoon is used by WHURLE and in turn by WHURLE-HM.

5.4.1 eXtensible Server Page (XSP)

XSP¹⁴ is one of the important developments coming out of the Cocoon project. XSP solves two problems found in JSP (JSP extends for Java Server Page, as it allows elements and inline Java code to be inserted into a normal HTML page, and when a JSP page is requested, the resulting code is executed and the results are inserted into the output HTML). The first problem is that JSP does not provide a separation of content and presentation. Secondly, there is no ability to transform the document into any other format, or use it across applications. This is because the JSP specification is designed primarily for the delivery of output only. The first problem is easily handled by XSP, as at its heart, it is simply XML, so presentation is separated from the contents. This allows developers to handle content generation, while XML and XSL authors can handle presentation and styling through modification of the XSL style sheet applied to an XSP page. As an example of XSP:

¹³ <http://cocoon.apache.org/2.1/>

¹⁴ [http:// http://cocoon.apache.org/1.x/xsp.html](http://http://cocoon.apache.org/1.x/xsp.html)

```

<?xml version="1.0"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<?xml-stylesheet href="style.xsl" type="text/xsl"?>

<xsp:page language="java"
  xmlns:xsp="http://www.apache.org/1999/xsp/core">
  <xsp:logic>
    private static int num=0;
    private synchronized int getnum() {
      return ++num; }
  </xsp:logic>

  <page>
    <title> hit counter </title>
    <p> i've been requested <xsp:expr> getnum()</xsp:expr> times
  </p>
  </page>
</xsp:page>

```

It can be observed from the above example that the Java code part is included between two elements `<xsp:logic>` `</xsp:logic>`, and the content between `<page>` `</page>`, so a separation found here; also, in the Java code part, a function was created and used inside the contents between the two elements `<xsp:expr>` `</xsp:expr>`.

The second problem is also easily solved, as XSP processing occurs before any style sheets are applied; the resultant XML document can be transformed into any other format. For example, in the previous code, the Java code in the XSP page would be processed and then the whole document would be transformed after that.

It is important to clarify that XSP technology is used extensively in producing different components of WHURLE-HM, such as the authentication component, the quiz engine, and importing custom JAVA classes. These components are explained technically in detail in Chapter VII.

Cocoon also has produced other technologies to connect an XML document with databases, such as ESQL tag library to do queries over the data in these databases as insertion, deletion, updating and searching.

5.4.2 ESQL tag library (ESQL logicsheet)

ESQL¹⁵ logicsheet is an XSP logicsheet and is the latest technology supported by Cocoon. This technology has a number of important advantages over its predecessors, such as SQL tag library and SQL processor that are not supported any more by Apache, as it allows mixing with other logicsheets. It also supports prepared statements, which gives automatic parameter escaping, and it allows encoding in a single document. The ESQL namespace uses the same parameters as the SQL namespace for getting a connection and a query. All of the formatting parameters have been dropped. In ESQL the result data from a query is contained in <ESQL:RESULTS> element. As an example of that:

```
<ESQL:EXECUTE-QUERY>
<ESQL:DRIVER>postgresql.Driver</ESQL:DRIVER>
<ESQL:DBURL>jdbc.postgresql://localhost/tst</ESQL:DBURL>
<ESQL:USERNAME>user</ESQL:USERNAME>
<ESQL:PASSWORD>password</ESQL:PASSWORD>
<ESQL:QUERY> SELECT * FROM CUSTOMER</ESQL:QUERY>
<ESQL:RESULTS>
  <ID><ESQL:GET-STRING COLUMN="id"/></ID>
  <NAME><SQL:GET-STRING COLUMN="name"/></NAME>
</ESQL:RESULTS>
</ESQL:EXECUTE-QUERY>
```

An important advantage of ESQL is that it allows nested queries, which was not possible before. ESQL technology has been used heavily in WHURLE-HM to establish connections with the database and exchange queries, as will be explained in Chapter VII.

5.5 Summary

Throughout this Chapter, an overview about the fundamental differences between XML and HTML is explained. In addition, it has been explained why XML is more powerful and why it provides a lot of flexibility. Moreover, different technologies, used alongside XML such as XLink, XPointer, and XPath have been discussed. Furthermore, the Cocoon publishing framework has been described with its related technologies.

The chapter has been deployed as follows:

¹⁵ <http://cocoon.apache.org/1.x/esql.html>

- **Introduction:** the introduction section sets the reasons for presenting this chapter.
- **XML and HTML:** differences between HTML and XML are described. In addition, a brief description for DTD and Schema is given.
- **Related Technology:** that section briefly explains XSL, CSS, XInclude and other XML languages such as XPath, XPointer, and XLink.
- **Cocoon publishing framework:** Cocoon and its related technologies are explained in this section.

For full information regarding XML and its related technology recommendation drafts, in addition to CSS refer to <http://www.w3.org>. Also, for detailed information about Cocoon development, refer to <http://xml.apache.org>.

Chapter VI: The Hybrid Model

6.1 Introduction

The Hybrid Model is an abstract generic user model developed for use by adaptive educational hypermedia frameworks, and initially tested in a Higher Education context. Thus, it handles users' knowledge levels in different knowledge domains concurrently. By adaptive educational hypermedia frameworks it is meant systems that: a) have the capability to deal with users with diverse knowledge, goals and background, b) have the capability to handle different knowledge domains simultaneously, c) can trace users' performance through different educational states, such as moving from first year undergraduate to second year undergraduate. The Hybrid Model is designed to be used by educational institutes to implement different online courses concurrently, taking into account the knowledge status of each individual student.

Different adaptive educational hypermedia systems, such as those described in Chapter IV, work on the level of concepts, i.e. they store information about the knowledge of each individual user, in addition to other characteristics, such as his/her preferences, with respect to concepts that belong to a single domain – knowledge domain-oriented. According to De Bra [De Bra et al. 1999], concepts may be divided into three categories: atomic concepts that are the smallest information units, pages that are composed of fragments, and abstract concepts that represent larger units of information. In addition, these concepts are connected to each other through different concept relationships, such as prerequisite relationships. Adaptive hypermedia systems maintain a model of users' knowledge about each involved domain's concept based upon the systems' observation, while a user is navigating through the adaptive hyperdocuments. For example, in the Metadoc system [Boyle and Encarnacion 1994], if a user changes the amount of detail in a requested page by means of stretch text operations, the system uses these changes as a basis for future adaptation according to the new inferred knowledge level.

The question that then arises is: what if we want to involve more than one knowledge domain, such as mathematics, biology, chemistry, and so forth, at the same time? In

addition, how do we maintain users' knowledge with respect to each involved individual domain? (This is important because there are interdependencies between concepts from different domains.) For example, in biochemistry, two domains might be used - biology and chemistry. The system then has to be capable of handling the prerequisites of any single involved domain in that subject, such as a certain knowledge level in chemistry and in biology, with respect to each individual user.

Metadoc [Boyle and Encarnacion 1994] to a certain extent tried to solve this issue, by classifying users with respect to their knowledge level into four stereotypes for every involved domain. Thus, for an n number of involved domains, the system should initiate an n number of four stereotype classes - each stereotype holds four knowledge classes for each domain. This solution is not adequate because the system cannot differentiate between users who have diverse educational state and share the same knowledge stereotype. For example, let's consider if first year and second year undergraduate students used the system simultaneously to teach them UNIX concepts. The problem comes when two or more students (one from the first year and the second from the second year) share the same knowledge stereotype. As a result, the system will provide both students with the educational material adapted in a same manner that suits the knowledge stereotype they belong to regardless their educational state (first year or second year). Another scenario may happen, if a second year undergraduate student accessing a certain curriculum/topic about UNIX, and that topic requires the student to have a certain knowledge level from the first year, such perquisites the system cannot handle as it cannot trace users' knowledge through different educational states. Therefore, according to these problems Metadoc system does not have the ability to manage users' knowledge globally through different educational stages.

In order to create an adaptive educational hypermedia framework, the users' knowledge should not be limited around a topic's concepts, but be extended to domains - as each domain contains topics that help to build users' knowledge about that domain [Zakaria and Brailsford 2002] as clarified in Chapter I. Moreover, multiple domains should be involved and integrated together in a single domain model, and there must be no limit to the number of involved domains. Furthermore, the user model should be capable of maintaining users' knowledge about each

involved domain, differentiate between users in different educational levels, and trace users' performance in different stages of their learning experience. This thus forms the fundamental rationale behind the Hybrid Model.

6.2 The Hybrid Model Architecture

The Hybrid Model [Zakaria and Brailsford 2002, Zakaria et al. 2002] is a generic abstract user model for adaptive educational hypermedia frameworks that simultaneously run different courses involving multiple knowledge domains for users of different goals, knowledge and backgrounds. Thus, it is based on measuring and classifying users' performance within knowledge domains, such as the biology domain, the mathematics domain, or any other domains involved in an educational curriculum. The Hybrid Model is composed of an overlay model, two sets of stereotypes: level stereotypes and category stereotypes, and the information pool, and as such represents a hybrid of the major techniques of user modelling. Figure 6.1 illustrates the component of this abstract model.

6.2.1 Overlay Model

The overlay model is currently perhaps the most widely used technique of user modelling. This technique involves the measurement of the knowledge level of users in any given topic or domain. A user's knowledge according to this model is considered to be an overlay of the total knowledge representing that domain. This knowledge level is represented in the form of "Concept-Value" pairs [Valley 1997, Carr and Goldstein 1977], but in the case of the Hybrid Model it is in the form of "Domain-Value" pairs [Zakaria and Brailsford 2002]. Moreover, this type of model has been utilised by the majority of adaptive educational hypermedia systems.

In the Hybrid Model, the overlay measures the knowledge level of each learner within certain subject domains. This knowledge level might represent the score achieved in the system assessment at the end of each lesson, course, or topic, although any other parameters the system authors may choose may also be used. For example, the score achieved in self-assessment quizzes is a widely used and well-accepted metric of the comprehension of information [Zakaria et al. 2002].

6.2.2 Stereotype Model

This model assumes that knowledge is customised for specific groups, with each user being assigned to one, and only one, group at any given time. Thus, users who share the same background or knowledge should be assigned to the same group. Users are not moved from one group (or class) to another until they trigger specific conditions that denote the new group [Rich 1999]. The Hybrid Model utilizes two different sets of stereotypes: level stereotypes and category stereotypes.

- *Level stereotypes*: level stereotypes mainly depend on the knowledge level of users. For example, they may simply be defined as Beginner, Intermediate and Advanced, but any classes may be used as appropriate to each system. According to the users' knowledge level, they are assigned to a single class of the level stereotypes within any given domain that they study. For example, a user studying biomechanics might be assigned simultaneously to the novice class in biology and to the advanced class in mathematics. Classes in the level stereotypes are concerned with providing assistance that is appropriate, and adapting the contents of any lesson to suit the learner. Each class may define an article or set of articles, links to external documents, or to lessons in other courses. For example, if a user belongs to one of the advanced classes he/she may be provided with advanced articles or links to help the user to find more about the topic or domain he studies. Level stereotypes have been used by many different systems such as CHEOPS [Ferrandino et al. 1996], Metadoc [Boyle and Encarnacion 1994], and many others.
- *Category stereotype*: the Hybrid Model has been designed for systems that simultaneously run multiple courses for different levels of users. For example, the system may be running courses for first year undergraduates as well as postgraduate users. Thus, users need to be categorized, as the knowledge level of undergraduate users in a certain stereotype level of a certain domain may not be the same as that of postgraduate users in the same stereotype level of the same domain. For example, consider two users: one of them a first year undergraduate and the second one studying for a higher degree. Both of these students are classified in the intermediate level stereotype for the biology domain. In addition, both of them are in the same level stereotype, however, the intermediate level of postgraduates will be

much more advanced than that of first year undergraduates. Category stereotypes solve that problem by assigning users according to their type of study or occupation into different categories. For example, undergraduate students belong to the undergraduate category and postgraduate students belong to the postgraduate category. Thus, members of each category are provided with information that suits their knowledge level with respect to their category.

6.2.3 Information Pool

The information pool is categorised by the domain model, and consists of a pool of articles, links, and other items that constitute the resources available to an adaptive system for any given domains. Thus the information pool embraces all the items that describe an educational curriculum and are subject for adaptation. It is likely that the information pool will differ in both form and content from one system that implements the model to another.

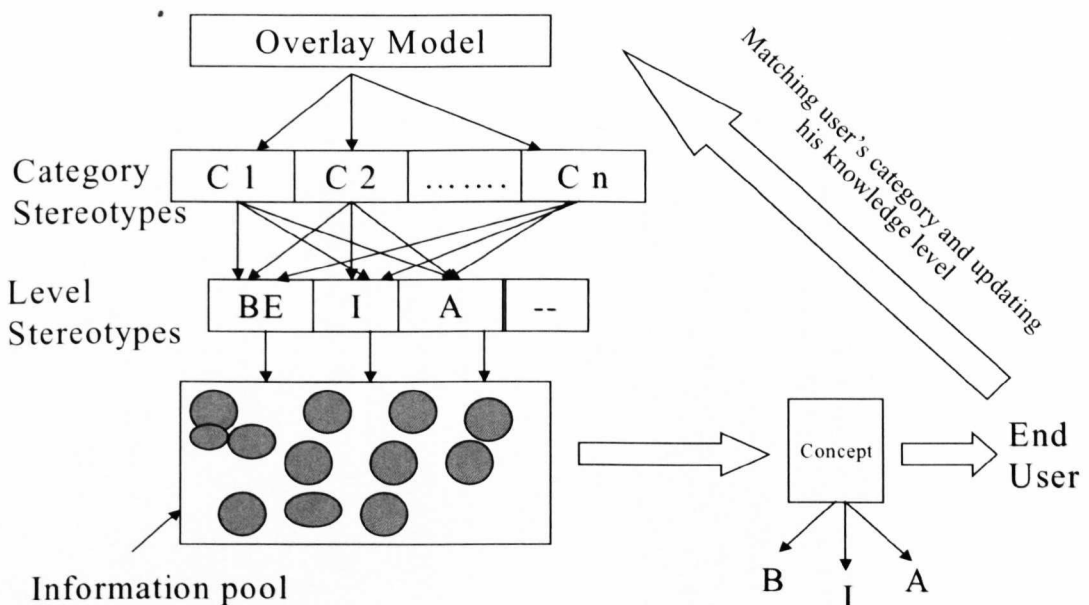


Figure 6.1 The components of the Hybrid Model – In this abstract model the overlay model combines with category stereotypes and level stereotypes to retrieve appropriate content from the information pool to convey appropriate adapted materials to users. Users interact with the system to inform and update the user model. BE – Beginner, B – Basic, I – Intermediate, A – Advanced, C - Category.

The overlay model, level stereotypes and category stereotypes combine to pick from the information pool the most convenient articles and links that suit each user's level, knowledge and background. Thus, according to a user's knowledge level and category, the most appropriate materials will be chosen from the information pool.

It is important to clarify, that the information pool is different than the knowledge domain of a framework that implement the model. This component embraces all the concepts that an educational curriculum/topic is composed of and has to be taught to potential students/users. On the other hand, the domain model of the framework that implement that model may contain different educational curricula, however, all the concepts that compose these curricula should be mapped to comprehensive semantic domains such as biology, chemistry, computer science, etc.

To give an example about the advantages the Hybrid model could offer, by considering a framework that implements the Hybrid model and has been applied to first year and second year undergraduate students, the following scenarios could happen:

- Students from both educational levels (first year and second year) could use the system at the same time and the system could provide each of them with appropriate adaptation that suit their educational state, and this is because of the category stereotypes.
- Students who moved to the second year their knowledge level about the studied curricula in the first year will be maintained and this is because the concepts that these curricula include are mapped to semantic domains (such as programming, mathematics, etc.), which are common among different educational stages. This is because the Hybrid model is stereotyping users' knowledge with respects to domains and uses the category stereotypes to differentiate between students' educational levels.
- By considering a student who is studying two curricula (A and B), and both curricula involve common domains. Moreover, if this student performed badly in one curriculum with respect to any of the presented domain(s)' concepts, the framework will be able to reflect that weakness automatically on the other curriculum, which will provide this student with additional explanations/examples that the course/curriculum author suggests for users who

belong to a knowledge level less than that of this particular student in this particular domain(s). Therefore, the Hybrid model could trace students' performance along the learning stages because it maps the involved concepts in any educational curriculum to comprehensive semantic domains.

6.3 The Hybrid Model and WHURLE

To exploit the functionality of the Hybrid Model, it has to be implemented through a strong educational hypermedia framework such as WHURLE (Web-based Hierarchal Universal Reactive Learning Environment) [Brailsford et al. 2002, Brailsford et al. 2001], which is an XML-based integrated learning environment. In this section, a conceptual view about the implementation of the Hybrid Model through WHURLE will be given. It is important to clarify that the implementation of the Hybrid model in the WHURLE system led to change the design of the system to produce WHURLE-HM. This change is explained in the implementation chapter (Chapter VII) particularly under the Adaptation filter subsection. Moreover, this new design is published in [Zakaria et al. 2003].

6.3.1 WHURLE Overview

The WHURLE system was initially developed as a successor to a system called the "Scholar's Desktop" (SD) [Davies 1994] that was developed by the TLTP (Teaching and Learning Technology Program) Biodiversity Consortium in the early 1990s. This system is an interactive learning environment designed to deliver hypermedia content. The content is structured as a "Study Unit" consisting of a number of nodes, which usually consist of a single root page surrounded with a number of children. Each study unit represents an interactive learning resource that is designed to promote specific learning objectives through the self-paced, interactive engagement with tasks, information, problems, or all three.

The SD software is a content-free shell, designed as a model of distributed development and delivery. WHURLE is the next generation of SD, where features that worked well with SD, with a change in the pedagogy, are implemented.

WHURLE is a server-based system-delivering HTML (or possibly in the future XHTML) dynamically generated from XML content by the use of XSLT (Extensible

Style Language: Transformation). In WHURLE the content consists of atomic chunks, each of which consists of the smallest conceptually self-contained unit of information that the author can envisage, where these chunks are totally transparent to the user/learner. All of the chunks available to any given instance of WHURLE are enclosed in what is called the melange. This melange acts as a pool where all of the chunks of all involved domains in the system are contained. What an end-user will see is a lesson, which is an apparent docuverse created by the WHURLE system. This contains the contents of any number of chunks together with navigational links and an overlaid environment that is generated by the system. The lesson is defined by another XML file that is called a Lesson Plan, which consists of WLPML (WHURLE Lesson Plan Markup Language).

The lesson plan contains a hypermedia pathway through the melange that is created by teachers using WHURLE (although default lesson plans are provided with a melange distribution). In its simplest conceptual form, a lesson plan consists of a hierarchy of levels, each containing one or more pages. Pages consist of chunks transcluded by means of XInclude¹. Moreover, levels could be nested inside each other. Figure 6.2 shows the architecture of WHURLE system

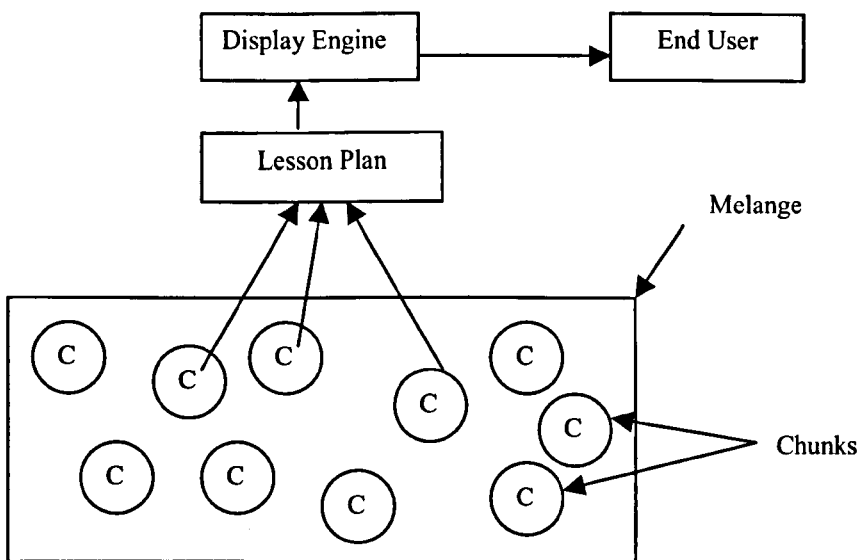


Figure 6.2 WHURLE components' architecture - from the above figure it could be seen that the WHURLE system is composed of: a) Melange: embraces all the chunks in the system, b) Lesson plan: composed of references to a subset of chunks (using XInclude technology) in the melange in addition to other information, c) the display engine: responsible for displaying a virtual document (Lesson plan), which is composed of different chunks, to the end user.

¹ <http://www.w3.org/TR/xinclude>

The processing of XInclude is orthogonal to both parsing and validation, and thus chunks are retrieved as required, rather than during the parse phase. Thus, there is a relatively modest processing overhead at parse time, and the server load is spread evenly during use. Figure 6.3 shows a simple extract from a lesson plan about HTML.

```

<lesson plan>
.....
<level name="intro-web-basics" title="The Basics of HTML">
  <page><chunk> introweb008</chunk></page>
  <level name="intro-web-tech" title="Important computing technical details">
    <page><chunk >introweb009</chunk></page>
    <page><chunk>introweb010</chunk></page>
  </level>
  <page><chunk >introweb011</chunk></page>
  <page><chunk>introweb012</chunk></page>
  <page>
    <chunk >introweb015</chunk>
    <chunk>introweb015a</chunk>
  </page>
</level>

<level name="intro-web-tags" title="Simple HTML Elements">
  <page><chunk >introweb016</chunk></page>
  <page><chunk >introweb017</chunk></page>
  <page><chunk >introweb018</chunk></page>
</level>

<level name="intro-web-link" title="Linking in HTML">
  <page><chunk >introweb027</chunk></page>
  <page><chunk >introweb028</chunk></page>
  <page><chunk >introweb029</chunk></page>
</level>
.....
</lesson plan>

```

Figure 6.3 An example of WLPML - This is an extract from a simple lesson plan about HTML basics. It could be observed that levels include one or more pages. And each page include one or more chunks. Also, levels could be nested inside each other.

If the code in Figure 6.4 is represented graphically as in Figure 6.3, it will be:

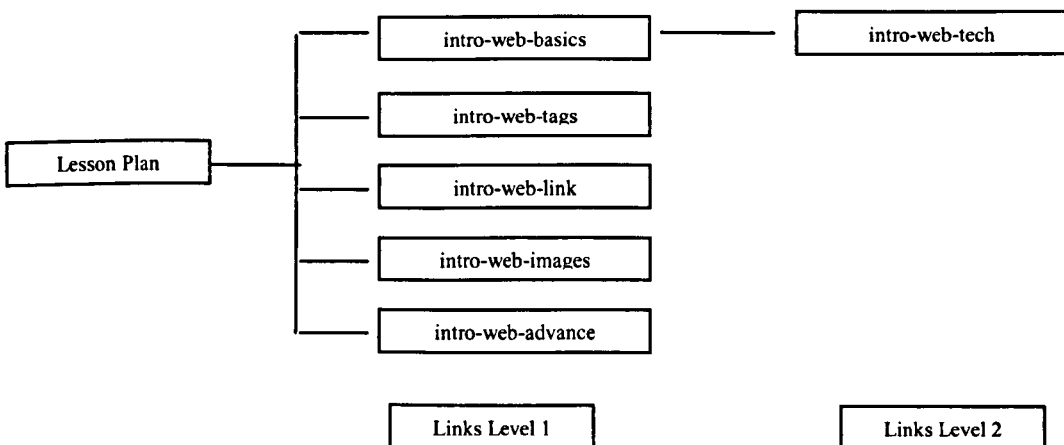


Figure 6.4 Graphical representation for the code in Figure 6.3 - it could be observed from that graph that level intro-web-basics (Links Level 1) include another link's level called intro-web-tech (Links Level 2).

6.3.2 The Hybrid Model Information Pool

Any lesson plan in the WHURLE system contains a number of chunks' references in addition to other information (in WHURLE terms called default narratives). Thus, the information pool, in the case of WHURLE-HM, embraces every chunk reference (in addition to other related information such as the author(s) of the lesson) the lesson plan has, as shown in Figure 6.5. On the other hand, the melange holds all chunks available in the system, i.e. a lesson plan includes a subset of chunks contained in the melange. Thus, the contents of the information pool differ from one lesson to another.

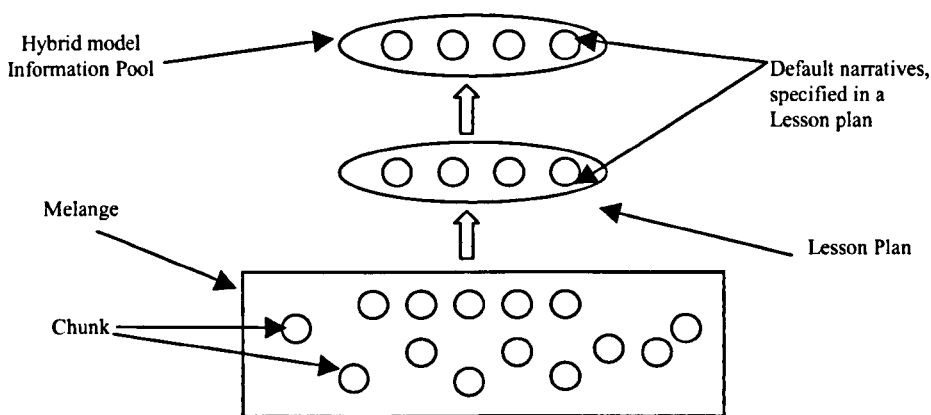


Figure 6.5 The Hybrid Model Information Pool in WHURLE-HM - A lesson plan holds references to a subset of chunks contained in the Melange in addition to other information (default narratives). The Hybrid Model information pool includes all the default narratives of the lesson plan

6.3.3 Knowledge Domains

The Hybrid Model depends on measuring and classifying users' knowledge with respect to involved domains in an educational curriculum. Thus, it is necessary to find a way to classify domains into sub-domains and sub-sub-domains, i.e. hierarchal classification. Because of this need, it was found that the best solution is to follow the same concept behind ontology systems such as Dewey Decimal Classification² (DDC).

Ontology is a fundamental branch of metaphysics, which is a branch of philosophy. That branch is concerned with the study of existence and its basic categories. Therefore, ontology has been defined by Guarino as "*a particular system of*

² http://www.oclc.org/dewey/about/about_the_ddc.htm#history

categories accounting for a certain vision of the world” [Guarino 1998]. An example is the categorization of human knowledge that is fundamental to the DDC as described below.

From the computer science perspective, especially AI (artificial intelligence) and knowledge representation, ontology refers a specific vocabulary presented in the form of concepts and relations’ names utilised to describe certain facts [Guarino 1998]. Thus, ontology in its simplest form describes hierarchal relationships between concepts [Guarino 1998]. Guarino called the AI definition of ontology conceptualisation. Therefore, according to Gruber [Gruber 1993] ontology is an “*explicit specification of a conceptualisation*”, where conceptualisation is an abstract overview about the domain to be presented. For example, in the DDC example, although Mathematics and Chemistry are different branches of science, both of them belong to the natural science and mathematics category but not to literature.

DDC is a general knowledge organization tool that is continuously revised to keep pace with knowledge. DDC coordinates materials on the same subject and on related subjects. That system has ten main classes which are [ANON 2003]:

- 000 Generalities
- 100 Philosophy and Psychology
- 200 Religion
- 300 Social Science
- 400 Language
- 500 Natural Science and Mathematics
- 600 Technology (Applied Sciences)
- 700 Arts
- 800 Literature
- 900 Geography and History

Each of these classes has its subclasses. For example, the Natural Science and Mathematics category embraces eight other subclasses, which are:

- 510 Mathematics
- 520 Astronomy
- 530 Physics
- 540 Chemistry
- 550 Geology
- 570 Biological Sciences
- 580 Botany
- 590 Zoology

Moreover, each of these subclasses is divided into others and so forth. Thus, the Dewey classification system works in a hierarchical form (from more general classes of knowledge to more specific in a hierarchal way). The DDC numbers are featured in the national bibliographies of 60 countries. The system was established by Melvil Dewey in 1873 and first published in 1876. Furthermore, many libraries around the world to classify their collections have used this classification system.

In WHURLE-HM, domains and subdomains are referred to in the same numbering concept such as in DDC example. However, the classification of the utilised domains in the system is up to the vision of the system's authors. Details about how this process is performed are explained in the implementation chapter (Chapter VII) particularly under the Database design section.

Although the main scope of this thesis about the Hybrid Model, and that model could use ontology, it is worth mentioning a new technology that relies heavily on ontology as well that is called the Semantic web. Ontology is an important feature that the vision behind semantic web [Berners-Lee 1998, Fensel and Musen 2001] relies. According to Timothy Berners-Lee, the head of W3C (World Wide Web Consortium), semantic web is "*a web of data, in some ways like a global database*" [Berners-Lee 1998].

The fundamental idea behind semantic web is to add meaningful descriptions (metadata) for web sites' pages that users visit by users themselves or by the creators of websites and could be shared among others. Such metadata have an ontology that facilitates computer software programs to understand and can infer new data from it. Therefore, such metadata will help machines to understand the context of the hyperdocuments. For example, if there is a web page of a certain company stating that all of their managers hold a PhD degree. Moreover, there is a home page

belonging to MR. X, who is a manager in this company. With the presence of such metadata, special software programs (i.e. agents) could infer that MR.X holds a Ph.D degree, although this information is not explicitly stated. In Chapter IX, a suggestion about how the Hybrid Model could be integrated with the semantic web technology will be given.

6.3.4 Knowledge Levels

Users' knowledge level, in the current implementation of the Hybrid Model, that is used in WHURLE, is represented through three stereotypes: novice, intermediate, and advanced. This kind of stereotyping is used by various other adaptive educational hypermedia systems, such as CHEOPS [Ferrandino et al. 1996], Hypadapter [Hohl et al. 1996] and many others.

The rationale behind stereotyping users' knowledge level is not only to provide advanced users with advanced information, or novice users with basics, but also to provide a kind of assistant. Thus, novice users could access other chunks from different domains or the same domain, or access links to other resources over the web, in addition to the presented concepts that suit their knowledge level. Likewise, advanced users could find more interesting advanced information about the topic they study. However, linking to other resources either within the system or outside it relies on the lessons' authors.

Users' knowledge level about involved domains is updated through answering quizzes, and taking tests at the end and the beginning of each lesson (depending on the lesson's author). Moreover, each question in a test or a quiz represents either one or more of the involved domain(s). According to a user's correct answers with respect to every involved domain question(s), his/her knowledge level about each involved domain is determined through a knowledge scale, which is a numerical value that ranges from 1 to 10. Additionally, for each range of scores, there is a corresponding knowledge value. For example, the knowledge value 1 corresponds to quiz scores that range from 0 to 10 (out of 100), and the knowledge value 2 corresponds to quiz scores that range from 11 to 20, and so forth, as shown in Figure 6.6.

Level stereotypes rely on the knowledge value of each user with respect to each individual domain to assign him/her to the appropriate expertise level. For example, the novice level might embrace users with knowledge values ranging from 1 to 4, the beginner level ranges from 5 to 8, and the advanced level ranges from 9 to 10. Furthermore, according to users' knowledge value, their level of expertise changes either positively or negatively, i.e. upgrading or downgrading. Thus, the system is tracking the performance of each user in each domain, and provides help whenever it is needed. Figure 6.6 shows how users' knowledge level is measured and stereotyped.

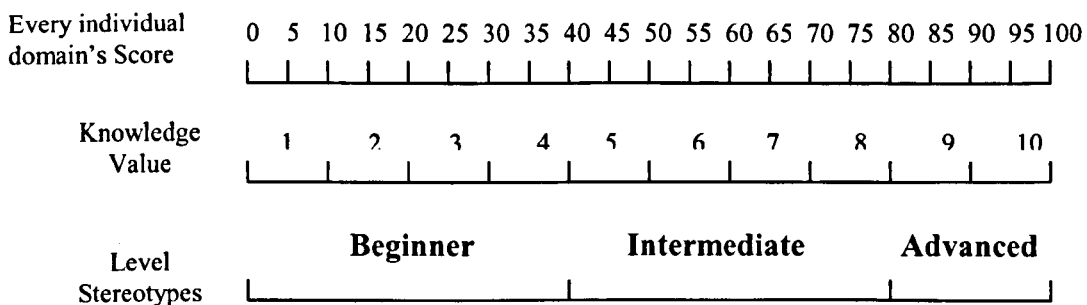


Figure 6.6 Knowledge Levels – Beginner level stereotype embraces knowledge values from 0 to 4, Intermediate level stereotype contains knowledge values from 5 to 8, while advanced stereotype holds knowledge values from 9 to 10. Each knowledge value represents a set of a quizzes' scores. For example, knowledge value 1 corresponds to scores from 0 to 10, knowledge value 2 from 11 to 20, and so forth.

The reason for representing users' knowledge level in that way is to give the system the maximum flexibility of changing the ranges that each knowledge level embraces. Thus, the system authors have the ability to resize the range of each knowledge value, and subsequently the range of each knowledge level will be resized. For example, knowledge value 1 could embrace quiz scores from 1 to 15, knowledge value 2 from 16 to 25, and so forth. Therefore, the level stereotypes ranges will be augmented, and the vice versa could happen, i.e. shrinking level stereotypes ranges.

6.3.5 Adaptive Lesson Plans

In WHURLE-HM, each lesson plan (which is an XML file stored in the system directory) has its own prerequisites, such as any mandatory lessons that must be taken before it, which are stored in a MySQL database. Thus, for a user to access a lesson plan he/she has to satisfy its prerequisites; also, he/she has to be a member of the same category that that lesson plan serves. The approach of a mandatory lessons

prerequisite helps in case of a topic or a course composed of more than one lesson plan and they should be taught in a certain order; also in case an author of a lesson plan has found that knowing a certain lesson before the user gains access will be useful. Each lesson plan may contain one or more level(s), which could be nested inside each other. Furthermore, levels are composed of one or more pages that embrace one or more chunk(s), as shown above in Figure 6.3.

```

<lesson plan>
.....
<level name="intro-web-basics" title="The Basics of HTML">
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb008</chunk></page>
  <level name="intro-web-tech" title="Important computing technical details">
    <page><chunk domain="general" stereotype1="" stereotype2="">introweb009</chunk></page>
    <page><chunk domain="general" stereotype1="" stereotype2="">introweb010</chunk></page>
  </level>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb011</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb012</chunk></page>
  <page>
    <chunk domain="html" stereotype1="beg" stereotype2="">introweb015</chunk>
    <chunk domain="html" stereotype1="beg" stereotype2="int">introweb015a</chunk>
  </page>
</level>

<level name="intro-web-tags" title="Simple HTML Elements">
  <page><chunk domain="html" stereotype1="beg" stereotype2="int">introweb016</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb017</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb018</chunk></page>
</level>

<level name="intro-web-link" title="Linking in HTML">
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb027</chunk></page>
  <page><chunk domain="general" stereotype1="" stereotype2="">introweb028</chunk></page>
  <page><chunk domain="general" stereotype1="" stereotype2="">introweb029</chunk></page>
</level>

<level name="intro-web-images" title="Images in HTML">
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb030</chunk></page>
  <page><chunk domain="general" stereotype1="" stereotype2="">introweb031</chunk></page>
</level>

<level name="intro-web-advance" title="More advanced HTML">
  <page><chunk domain="general" stereotype1="" stereotype2="">introweb035</chunk></page>
  <page><chunk domain="general" stereotype1="" stereotype2="">introweb036</chunk></page>
</level>

```

Figure 6.7 A simple extraction from an adaptive lesson plan about html - Chunks with the domain's attribute value "general" will be available to every user accessing this lesson regardless of his/her knowledge level about the involved domain. Chunks with the domain's attribute value "html" and stereotype "beg", means that the user should have a beginner stereotype knowledge level to access this chunk, where "beg" abbreviates for Beginner and "int" abbreviates for Intermediate

In adaptive lessons plans, two types of chunks are defined: non-conditional chunks, which do not have prerequisites to be met, and conditional chunks that do have prerequisites to be fulfilled by users before their inclusion. The type of each chunk is defined through a domain attribute, which either holds the name of the domain the chunk is serving, or the value "general" that indicates the non-conditional type of that chunk. Moreover, every conditional chunk has two other attributes in addition to

the domain attribute: stereotype1 and stereotype2. Those attributes determine the required knowledge level(s) to access that chunk, as at least one of them should not hold a NULL value, as shown in Figure 6.7. Thus, a user has to fulfil one of them, if both of them hold a knowledge level to be met, to gain access to that chunk.

The reason for these two stereotype attributes is that any chunk that is essential for users with a lower knowledge level to know could be utilized by users with a higher knowledge level to understand a new piece of information, and that depends on lessons' authors to decide. Furthermore, just two attributes are used, not more or less, because in the current implementation there are only three knowledge level stereotypes. What is more, every lesson a user has finished is recorded into his model for future revising, if needed.

6.3.6 Adaptation Mechanism

According to Brusilovsky [Brusilovsky 2000, Brusilovsky 1997, Brusilovsky 1996], there are two kinds of adaptation widely used in adaptive systems: adaptive presentation and adaptive navigational support. The idea of adaptive presentation is to adapt the content of a page accessed by a user to suit his/her knowledge, goals and other characteristics. On the other hand, the adaptive navigational support helps users to find their path through hyperspace by adapting link presentation to the knowledge, goals and other characteristics of each user.

Due to the nature of WHURLE's infrastructure, these two types of adaptation are combined together in the adaptation process. When a user clicks on a level link, the first associated page within that level will be included with its available chunks. If any page within any level does not have any chunk to be included, the link to this page is removed; the same thing happens if a level does not have any pages to be included. Thus, this link removal technique is used to adapt the content of the lesson plan, and thereby adaptive representation and adaptive navigational support are combined together (collateral structure adaptation technique explained in Chapter III), as shown in Figure 6.8.

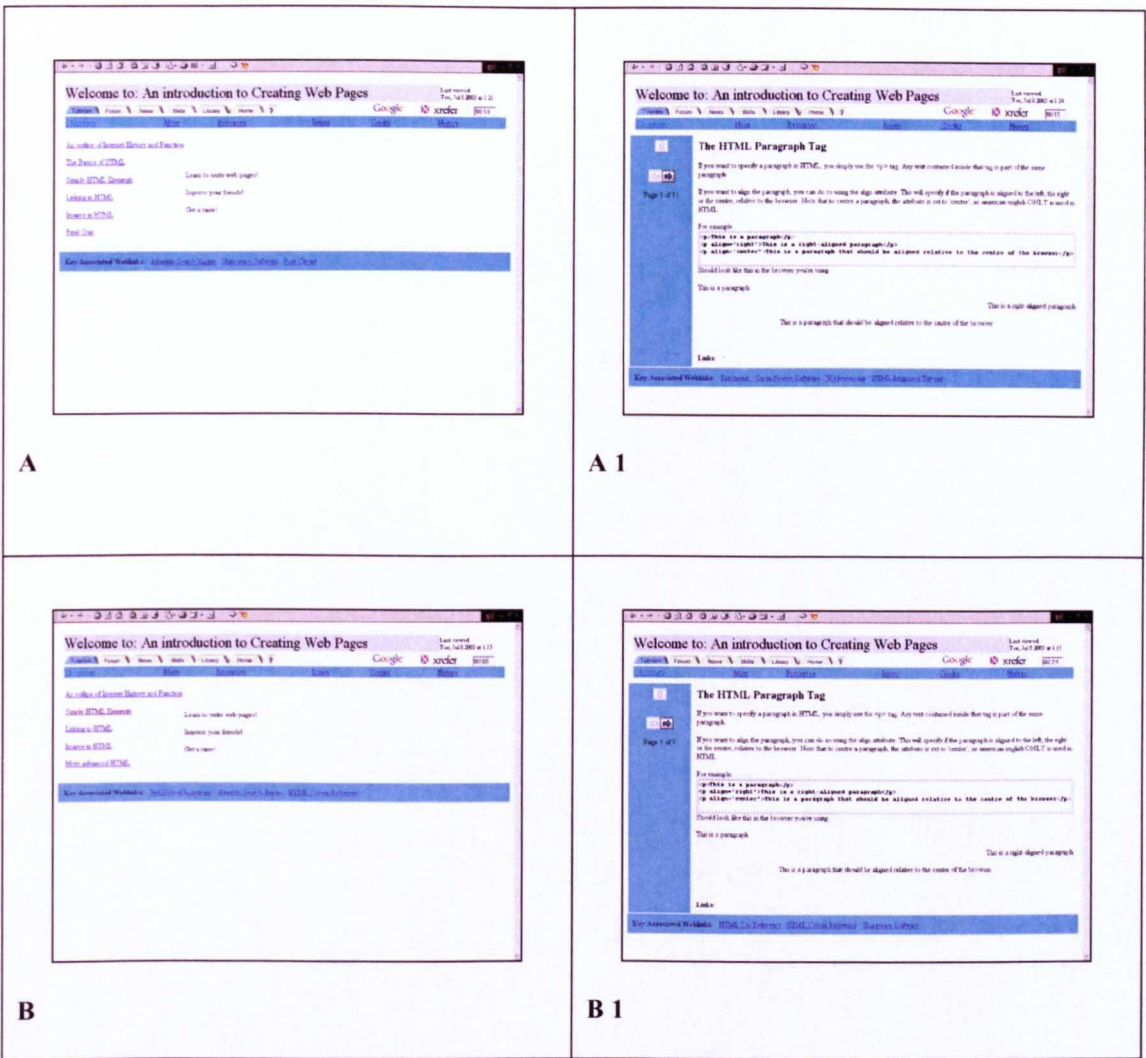
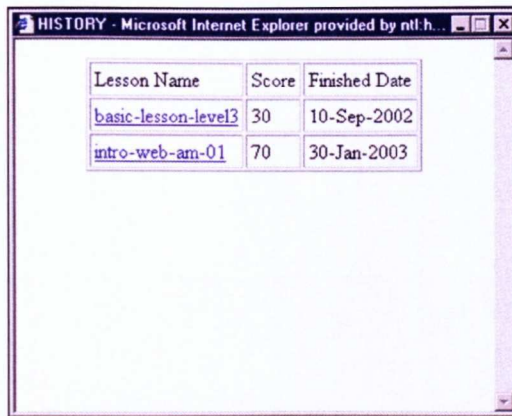


Figure 6.8 Adaptation Snapshots - Image A represents the first page of a non-adapted version of the html lesson plan where links to all levels (6 levels) are found. Image A1 represents the content of the second level, as it is composed of 11 pages, and a user can access them by means of the navigational buttons on the left frame. On the other hand, image B represents the adapted version of the first page of the same lesson plan, where links to five levels only are found and the sixth one is removed as none of its pages has chunks to include. Image B1 represents the content of the same level that image A1 represents, but with links to only seven pages not 11, whereby on the other four pages the user has not met their prerequisites; in addition none of them is of non-conditional type.

The adaptation engine in WHURLE-HM acts as a filter to lessons' content, where conditional chunks whose prerequisites are met and non-conditional chunks are included. Thus, a user will access information that is appropriate for his/her knowledge level. Through the history links, which is on the toolbar at the far right side in Figure 6.8 (images: B and B1), users may access all visited lessons in a non-adaptable version (without excluding any chunk) through the pop-up history window, as shown in Figure 6.9. The idea behind that approach is to build a kind of library for every individual user, composed of all visited lessons in a row format without

adaptation, as he/she can refer back whenever he/she wants to maintain information about studied domains.



Lesson Name	Score	Finished Date
basic-lesson-level3	30	10-Sep-2002
intro-web-am-01	70	30-Jan-2003

Figure 6.9 History Window - When a user clicks on the history link, as in Figure 6.8 Image B or B1, a pop-up history window comes into view, which includes links to all visited lessons, in a non-adapted version as in Figure 6.8-Image A and A1, in addition to the total score the user got at the end of each of them; also the date in which he/she finished each lesson.

In addition, adaptive navigation support is used through the use of colour annotation. When a user accesses the system, all lessons that serve his/her category are listed in a list box with different font colours. Three colours are used: Red – represents lessons that the user has not accessed yet, Orange – represents lessons that are open but not finished, and Green – represents finished lessons. For a lesson to change its state from open to finished, the user has to take its final quiz. As a result, his knowledge level(s) with respect to the involved domain(s) is/are updated and his/her old knowledge level(s) is/are stored in the database. The reason in storing the old knowledge levels is to give the user the ability to revise each finished lesson in its old (before updating his/her knowledge level) adapted format. Thus, users have the option to revise finished lessons either in a raw format through the history link (as in Figure 6.9), or in its old adapted format by choosing it from the lessons' list. Figure 6.10 shows the annotated lessons according to their state. Moreover, the state of the lesson changes from not-accessed to open if the user got the chosen lesson's pre-quiz, or accessed it directly in case that lesson does not have a pre-quiz. The post-quiz and pre-quiz are described in Chapter VII

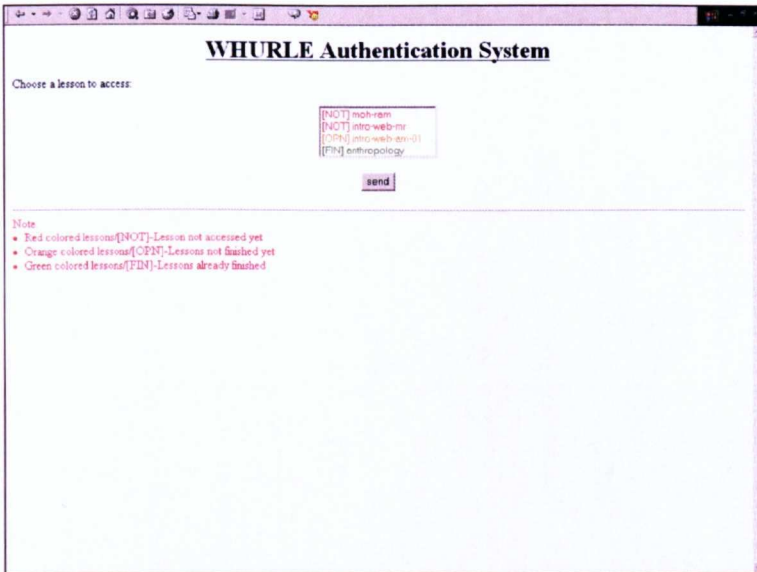


Figure 6.10 Choosing Lessons - In the list box, lessons' names are annotated according to their state. Red font- lessons not accessed yet, Orange font- opened lessons, and Green font- finished lessons.

The Hybrid Model used within WHURLE-HM is a cooperative [Kay 1995] type of user model, because it collaborates with users in gathering information. The users are required to supply the system with personal information when they access the system for the first time, e.g. their occupation/category, preferences, and other information items.

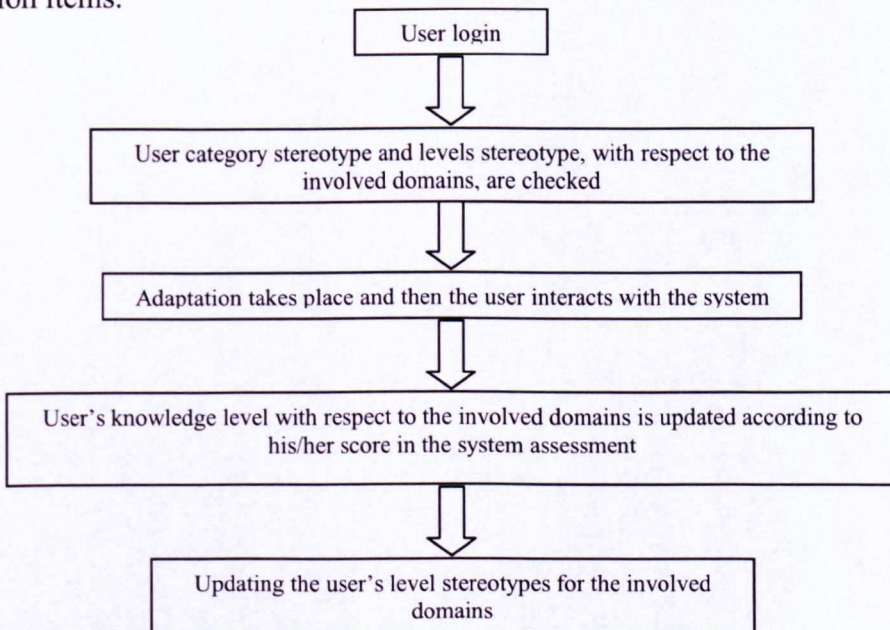


Figure 6.11 Users Interaction - Once the user logs in, the system either adapts its material according to the existing user model, or creates a new one. At the end of each lesson, the user model is updated and the old knowledge levels are stored.

In Figure 6.11, when a user logs on to the system, he/she supplies his/her username and password to verify his/her authentication. Subsequently, lessons that serve the

same category the user belongs to are listed in colours in the list box as shown in Figure 6.10. If a user chooses a lesson that is not accessed yet, the system starts to check if that lesson has any prerequisite lessons; if it has, the system informs the user to finish the prerequisite lesson(s) before accessing the current one. If the system does not have any prerequisite or the user has fulfilled them, the system checks if the lesson has a pre-quiz or not; if it has, the user is then directed to the quiz and then he/she can access the lesson. In case of lessons that do not have pre-quizzes, users can access them directly. Thereby, the lesson state will be changed from not-accessed to open, the adaptation process takes place, and the user starts to navigate through the adapted information. Hence, at the end of each lesson, the user may answer a quiz or take a test, and depending on his/her score, his/her knowledge level about each involved domain is specified. Therefore, the user is re-assigned to one of the level stereotypes for each involved domain, and his old knowledge level(s) is/are stored. Moreover, the lesson state will be changed from open to finished. In case a user has chosen a finished lesson to access, the lesson will be adapted according to the knowledge levels the user had before finishing that lesson. Thus, the user can revise that lesson.

In the case of new users, they have to supply the system with personal information, such as their name, category, etc, before accessing the system. Through this information, the system knows which lessons each user can access and which they cannot, and of course that depends on the category of each user.

6.4 Summary

Throughout this chapter, a detailed overview about the Hybrid Model, which is an abstract and generic model, with its components has been given. In addition, an overview about the WHURLE system has been described, and an explanation given about how every component in the Hybrid Model is applied through WHURLE to produce WHURLE-HM. Moreover, some implementation examples are given. In the Implementation chapter (Chapter VII), full technical details are explained. An attention is given to the new design by which the adaptation filter is integrated in WHURLE to produce WHURLE-HM is explained in Chapter VII. It is important to

notice that this chapter explained the conceptual implementation of the Hybrid Model through WHURLE. This chapter is organized as follows:

- **Introduction:** the introduction explained the motivation behind the Hybrid Model, and why the Hybrid Model is created and for which purpose.
- **The Hybrid Model architecture:** in that section, the Hybrid Model with all its components is explained.
- **The Hybrid Model and WHURLE:** this section gives a brief overview about the WHURLE system and its origin. Furthermore, it explains the conceptual implementation of the Hybrid Model through WHURLE including every component the Hybrid Model embraces. Also, a brief description about the system map is described. In the Implementation chapter (Chapter VII), this map is explained in full detail.

Chapter VII: Implementation

7.1 Introduction

In the Hybrid Model chapter, it is explained what is meant by the Hybrid Model and how it is conceptually implemented through WHURLE. As explained in the former chapters, especially in the technology chapter (Chapter V), in WHURLE-HM different technologies have been used together, such as ESQL, XSP, and JAVA. In this chapter, the logical mechanism of the different components of the system, in addition to how the actual files cooperate with each other will be explained. Before digging deep into technicalities, a full picture about how the whole system is working will be discussed through describing the implementation algorithm of the system.

7.2 Implementation Algorithm

WHURLE-HM is implemented according to the following algorithm:

- When a user logs into the system he/she has to supply his/her user name and password. If the user is not registered, he/she will be declined.
- If the user is registered, his/her category will be checked. Thereby, all lessons that serve that category will be listed in different coloured fonts as explained in the Hybrid Model chapter (Chapter VI) (Red: new lessons, Orange: unfinished lessons, Green: finished lessons).
- In case the user has chosen a new lesson, the system looks for any prerequisites associated with that lesson that the user has not finished. In other words, it looks for mandatory lesson(s) for that lesson that the user has not completed. If there is one, the system directs the user to that particular lesson(s).
- If the user chooses a new lesson to access, for which he fulfilled its prerequisite(s), and that lesson has a pre-quiz, or he/she is directed to a mandatory lesson that has a pre-quiz, then he/she will be directed to that quiz. Moreover, that lesson's state will be changed from new to opened with respect to that user. In case of opened lessons and finished lessons, the user will access them directly even if they have a pre-quiz, which makes sense, as the user had

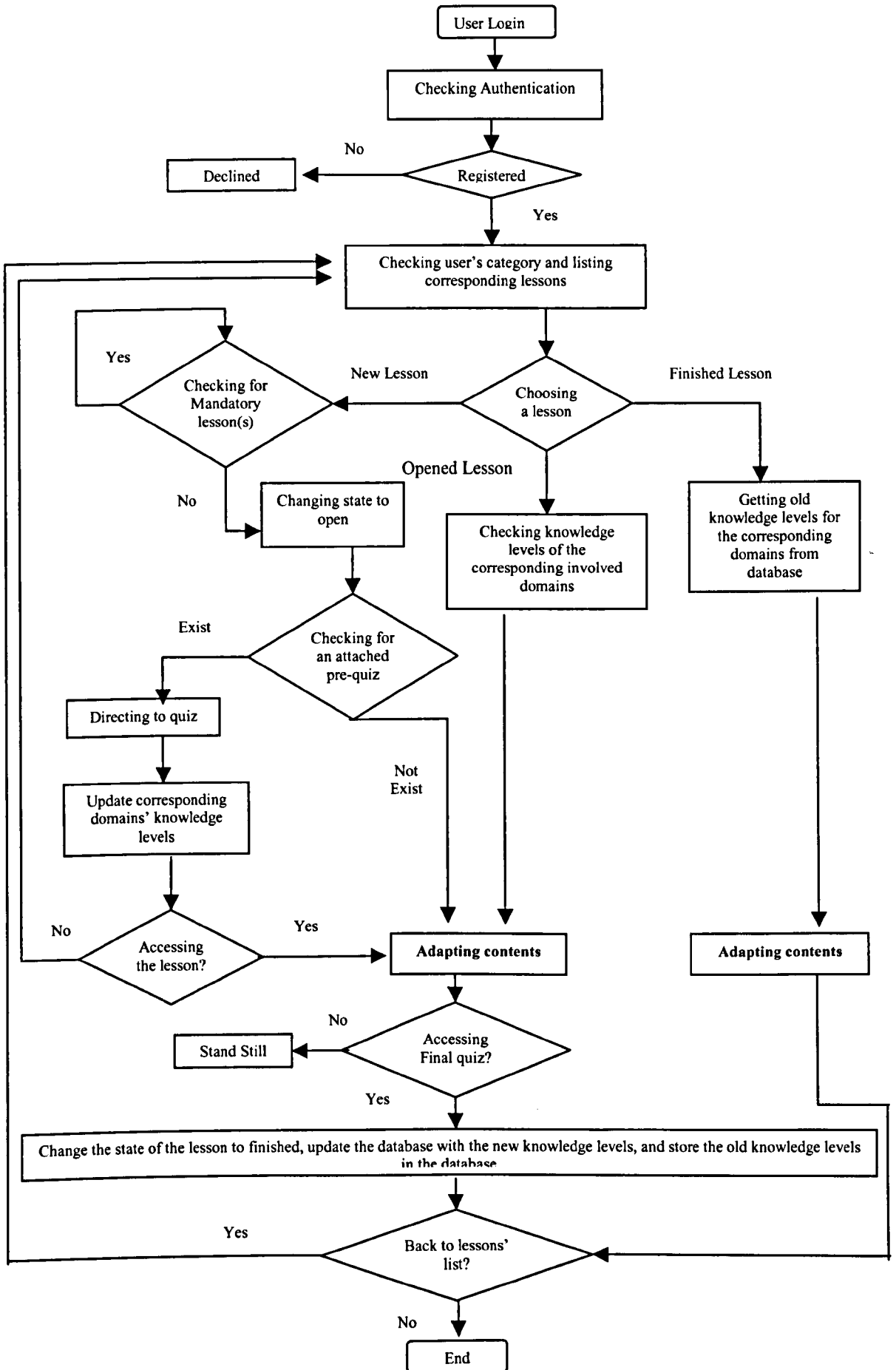


Figure 7.1 WHURLE-HM Flowchart

supposedly finished the associated pre-quiz (if found) in order to access that lesson.

- After the user finishes the pre-quiz of a new lesson, he/she gets the option to either access the lesson or to go back to the lessons list
- The knowledge levels that the user attained in the pre-quiz with respect to the involved domains in that lesson have their main effect in the adaptation process. Thus, the lesson's contents will be adapted according to these levels – as described in Chapter VI.
- The change in knowledge levels will be reflected throughout all opened lessons. For example, if a user with a beginner knowledge level in the Biology domain and in a pre-quiz his level changed to intermediate. Thus, this kind of upgrade in knowledge level will be reflected over opened lessons' adaptation process, which include that domain. It is important to notice that that reflection is not applied to finished lessons.

In case of new lessons and opened lessons, an additional hierarchal level will be added to the body of these lessons called “final quiz”. This level allows users to gain access to the final quiz after finishing with the lesson. Nevertheless, in case of finished lessons, that level does not exist

- In case of opened lessons, when a user finishes the final quiz (post-quiz), the lesson's state changes to finished. Moreover, the knowledge levels with respect to the domains involved in that lesson, before taking the final quiz, will be stored in the MySQL database. Thereby, when a user accesses a finished lesson, that lesson will be adapted according to those stored knowledge levels.

Figure 7.1 shows the flowchart for the current implementation.

7.3 Database Design

In the WHURLE-HM, the MySQL DBMS (Data Base Managing System) is used, as it is well supported by a huge and active mailing list, and also because it has been used by many applications, which means it is a reliable DBMS, moreover the researcher has an experience in using it. The WHURLE-HM database is composed of 13 tables. Each of them will be explored, and then their functionality will be explained. The tables will be divided into two categories: Systems' tables, Users' tables.

7.3.1 System's tables

System's tables mean tables that are essential for configuring the system to carry the adaptation process. These tables are:

- *Category*: this table identify categories where the system is going to serve, and it holds two fields:
 - o *Category_name*: a unique name for each category.
 - o *Category_id*: a unique ID for each category.
- *Conf*:conf abbreviates for configuration file, which holds a skin and other configurations. Thus, this table holds different configuration files that in turn hold different skins a user may choose when he/she registers in the system. That table holds two fields:
 - o *Conf_name*: a description name of the file; for example, Modern Style, Egyptian Style, etc.
 - o *Conf_file*: the actual physical name of the file without extension.
- *Domain*: that table introduce to the system the semantic domains that are involved in the educational process, such as Biology domain, Physics Domain, etc. That table could be incremented by means of lessons' authors as well. That table holds two fields:
 - o *Domain_name*: the names of involved domains, such as Chemistry, Biology and so forth.
 - o *Domian_id*: a unique ID for each involved Domain.
- *Knowledge_scale*: this table hold ranges of quiz scores and their corresponding knowledge values, as explained in Chapter VI. These knowledge values are numerical values that range from 1 to 10. That tables holds three fields :
 - o *Urange*: holds the upper score in a range of scores.
 - o *Lrange*: holds the lower score in a range of scores.
 - o *Knowledge*: holds the corresponding knowledge value for scores that come between the Urange and Lrange; for example, if a user got any score between 10 (Urange) and 0 (Lrange) its knowledge value is 1.
- *Stereotype_scale*: that table holds stereotype classes that are based on the knowledge values that could be obtained from the Knowledge_scale table. That

kind of detaching between knowledge scale and stereotype scale gives a very useful advantage, as system administrators could augment the stereotype classes' ranges without affecting the knowledge value ranges, and the opposite could also happen. Moreover, as described in Chapter VI, the stereotype classes could be Beginner, Intermediate and Advanced. That table holds three fields as well:

- *Urange*: holds the upper knowledge value in a range of knowledge values.
- *Lrange*: holds the lower knowledge value in a range of knowledge values.
- *Stereotype*: holds the corresponding stereotype for knowledge values that come between *Urange* and *Lrange*; for example, if a user got a knowledge value between 5 (*Urange*) and 1 (*Lrange*), he/she would be considered to be a beginner.

7.3.2 Users' tables

Users' tables means tables that are specifically oriented to lessons' authors and users to feed more than a system administrator. Thus, after configuring the system by supplying it with the necessary data as described in the system's tables section, the requirement of each lesson needs to be supplied by its author(s); also users need to register themselves to the system. So, users' tables include:

- *Lesson_flow*: that table holds the names of involved lessons in addition to other related information through the following fields:
 - *Lesson_name*: holds the name of the involved lessons.
 - *Lesson_id*: a unique ID for each lesson that is automatically created by the system.
 - *Categ*: that field holds the ID of each category that each lesson serves.
 - *Closingdate*: holds that date at which each lesson expires, which of course depends on lessons' authors either to set that date or to leave it open. In fact, in the current implementation, that field is left for future usage if needed.

- *Lesson_req*: that table holds information about the requirements of each lesson, which include:
 - o *Lessonid*: the ID of each involved lesson.
 - o *Domainid*: the ID of a domain(s) that each lesson embraces.
 - o *Pretest*: that field holds the names of pre-quizzes' files, if existing, or null.
 - o *Postest*: holds the name of final quizzes' files for every involved lesson.

- *Lessons_record*: when a user finishes a lesson the state of that lesson changes from open to finished by registering itself with other related information in that table, which holds four fields:
 - o *Stid*: holds the ID of every user who finished one or more lesson.
 - o *Lessonid*: that field holds the ID of every finished lesson with respect to every involved user.
 - o *Score*: that field holds the final score that every involved user got with respect to every finished lesson.
 - o *Finishdate*: that field holds the dates at which every involved user finished every involved lesson in the form of dd-mmm-yyyy, such as 23-Sep-2003

- *Mand_lessons*: that table holds the name of mandatory lessons. In other words, identifies the sequence of lessons. For example, as described in the Hybrid Model chapter (Chapter VI), a lesson may need another lesson(s) to be taken by a user before accessing that one. That table holds two fields:
 - o *Lesson_id*: that field holds the ID of every lesson that has a mandatory lesson to be taken before it.
 - o *Mand_lesson_id*: that field holds the IDs of the corresponding mandatory lessons.

- *Old_lvl*: as described in the implementation algorithm section, when a user finishes a certain lesson, by taking its final quiz, its old knowledge levels are stored. The *old_lvl* table is responsible for storing these knowledge levels. That table has four fields:

- *Stid*: ID of involved students.
 - *Lesson_id*: the ID of every finished lesson with respect to every involved user.
 - *Dom_id*: the ID of every domain associated with every finished lesson.
 - *Dom_lvl*: holds the stereotype of every user with respect to every involved domain in every lesson that each user has finished at the time when each lesson is finished, i.e. before taking a final quiz.
- *Open_lessons*: when a user chooses to access a new lesson, the status of that lesson changes from new to open by registering itself in that table, which holds two fields:
- *Stid*: the ID of every involved student; in other words, every student who has an open lesson(s).
 - *Open_lesson*: the ID of every open lesson with respect to every involved user.
- *Stereotype*: that table holds the stereotype class of every user with respect to every domain he/she accessed through a lesson(s). Moreover, all the updates in the stereotype classes for every user are registered in that table. Thus, the stereotype table is considered as being the repository of the adaptation engine. That table holds three fields:
- *Stid*: the ID of every involved student/user.
 - *Domain_id*: that field holds the ID of every domain that each individual user has accessed.
 - *Class*: that field holds the stereotype class, such as beginner, intermediate or advanced for every domain with respect to every involved individual user.
- *User*: that table holds related information for every user registered in the system. It is composed of seven fields:
- *Fname*: the first name of every registered user.
 - *Lname*: the last name of every registered user.
 - *Username*: a user name that each user chooses to identify him/herself to the system through the login screen.

- *Password*: a password that each user chooses to identify him/herself. It is important to notice that the combination between the user name and password for every individual user should be unique. Thus, it does not matter if two users have the same user name or the same password.
- *Stid*: a unique ID for every student, such as his university card number or his ID number.
- *Categ*: that field holds the category that each registered student belongs to through the ID of that category.
- *Configfile*: holds the name of the configuration file that each student has chosen while registering in the system.

The full structure of the database and the tables with examples can be found in Appendix B.

7.4 System Components

In addition to the melange, lessons' plans, the display engine (the same in WHURLE), and the database (described above) WHURLE-HM is composed of another three parts: Authentication system, Quiz engine, and Adaptation filter, as shown in Figure 7.2. These parts plays an important role in:

- Providing access to registered users.
- Testing users' knowledge with respect to the domains involved in the chosen lesson (first time access a lesson or after finishing it).
- Providing adapted versions of a lesson according to users knowledge level with respect to the involved domains.

From Figure 7.2, it could be observed that the whole communication process between the three components of the system in addition to the other components is in a black box, therefore, users would not see the underlying structure of the system. All what an end user (student) can see is a quiz (post- or pre-) and an adapted lesson plan. The following sections will describe each of these three parts technically, and will explain how they communicate between each other and the database.

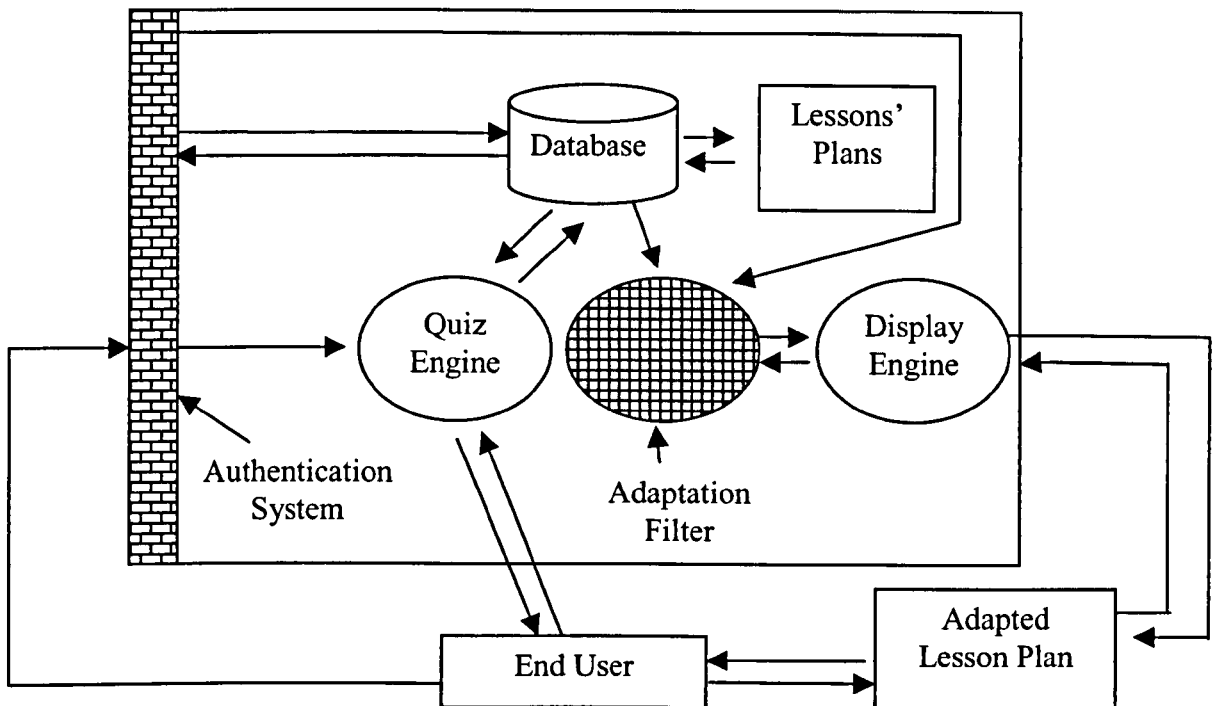


Figure 7.2 WHURLE-HM Components – a) Authentication system – users have to be registered to gain access to the system. Moreover, it is responsible for directing users either to the quiz engine, in case of new lessons that have pre-quizzes, or to the adaptation filter, in case of open lessons or new lessons that do not have pre-quizzes; b) Quiz Engine – if a user has chosen a lesson that has a pre-quiz, then he/she will be directed to the quiz engine, which will produce the quiz and then the knowledge levels of the corresponding domains will be updated; in case of a post-quiz, the role of the quiz engine will be activated once more; c) Adaptation filter- it is responsible for filtering the contents of a lesson plan (depending on users' knowledge level stored in the database) and pass the adapted version to the display engine to present it to the end user.

7.4.1 Authentication system

The authentication system is composed of five files: *was.html*, *was_mid_les.xml*, *was-tst.xml*, *was_les2.xml*, and *was.xml* (the full code is found in Appendix A). When a user logs on to the system, he/she will access *was.html*, which is responsible for the design of the login screen (a snapshot of the login screen is in Appendix C). Thus, he/she has to supply his/her user name and password to access the system, thereby, being directed to *was_mid_les.xml*. In fact, the *was_mid_les.xml* file imports a custom Java class, which is responsible for checking if the user name and password are registered in the user table in the database. If they are registered then the user is allowed to access the system, otherwise, he/she will be declined. Moreover, that file is responsible for getting all the lessons that serve the same category the user belongs to, once he/she is granted access to the system, from the database by means of ESQl technology, as described in Chapter V. In addition, it is responsible for classifying

them into three categories: new lessons, open lessons, and finished lessons (a snapshot of the lessons' screen is in Appendix C). It does that by checking lessons in the `lessons_record` table, `open_lessons` table and `lesson_flow`. Technically, lessons that are in the `open_lessons` table are displayed in an orange coloured font, those in the `lessons_record` table are in a green coloured font, and those in the `lesson_flow` table are in a red coloured font. Moreover, *was-tst.xml* is responsible for colouring the font of lessons' names. Thus, users (students) have three options to choose from, either to choose a new lesson (not-accessed yet), an opened lesson, or a finished lesson to revise. In all cases, after choosing a lesson, users will be directed to the *was_les2.xml* file, which will behave differently in each case, as follows:

- Lessons not accessed yet: in the case of a new lesson, the *was_les2.xml* will communicate with the database to check if that lesson has any mandatory lesson(s) that the user has not finished. Then it checks if that lesson has a pre-quiz or not. Hence, we have two cases:
 - A mandatory lesson(s) exists: if the lesson has a mandatory lesson the user has not finished yet, *was_les2.xml* directs that user to that particular lesson and so forth through *was.xml*.
 - No mandatory lesson(s): if the lesson has no prerequisite lessons the user has not fulfilled, *was_les2.xml* starts checking if that lesson has a pre-quiz or not. Thus one of the two following routes the user will follow:
 - Pre-quiz route: if that lesson has a pre-quiz, *was_les2.xml* will direct the user via *was.xml* to the quiz engine to access the corresponding quiz.
 - Normal route: in that case, that lesson has no quiz. Thereby, *was_les2.xml* will communicate once more with the database to get all related information, such as a configuration file from the user table and knowledge levels of domains associated with that lesson from the stereotype table. After that, *was.xml* will direct the user to the chosen lesson that will be filtered by means of the adaptation filter. Moreover, an additional hierarchal level in the lesson's hierarchy called "quiz" will be added (a snapshot of an adaptive lesson is in Appendix C), which

directs users to the final quiz associated with the accessed lesson.

- Open lessons: in the case of open lessons, the normal route will be applied. The only difference is that the system will acquire the knowledge levels of associated domains directly from the database, instead of getting them from the quiz engine.
- Finished lessons: with the finished lessons case the scenario differs, as *was_les2.xml* will not communicate with the stereotype table to get the knowledge levels of the associated domains. Instead, it will get these values from the *old_lvl* table. Moreover, the additional hierarchical level “quiz” will be removed from the adapted lesson, which makes sense because this kind of lesson is only for revising, and the user will supposedly have finished its final quiz in order to change the lesson’s state to finished.

7.4.2 Adaptation filter

The adaptation filter is considered to be the main gear of adaptation, as through it the content of any lesson is filtered to suit users’ knowledge, goals and background. In fact, *whurle-filter.xsl* is the file that is responsible for this operation (the full source code is in Appendix A). In WHURLE, before adaptation, the system was composed of two parts, as shown in Figure 7.3.

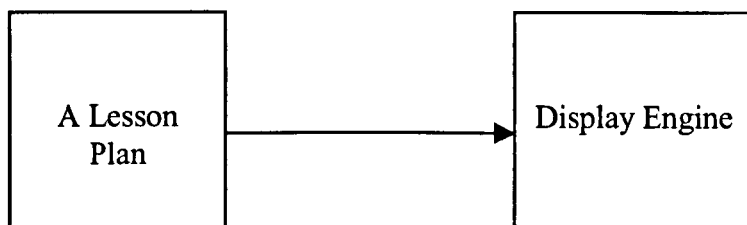


Figure 7.3 WHURLE Old Infrastructure - was composed of two parts: a) a lesson plan, which is composed of a number of chunks, as described in Chapter VI; b) a display engine, which is responsible for drawing the hierarchical levels of lessons, and displaying them according to the chosen configuration files, as described in Chapter VI.

During the design process of the adaptation filter, it was important to detach the adaptation filter from the display engine. Thus, any change or update in the adaptation filter will not affect the display engine and vice versa, which in turn provides the system with maximum flexibility [Zakaria et al. 2003]. Therefore, the adaptation filter is fitted to the system as an intermediate stage between the involved lessons’ plans and the display engine as shown in Figure 7.4. Technically, *was_les2.xml* file before activating the adaptation filter it puts in session variables the

knowledge levels of a user with respect to every associated domain with the chosen lesson. After that, the adaptation filter reads these variables, which may hold for example, Biology = Intermediate, Chemistry = Beginner. Afterwards, it starts to compare the required knowledge level for every chunk that serves a certain domain with those found in session variables.

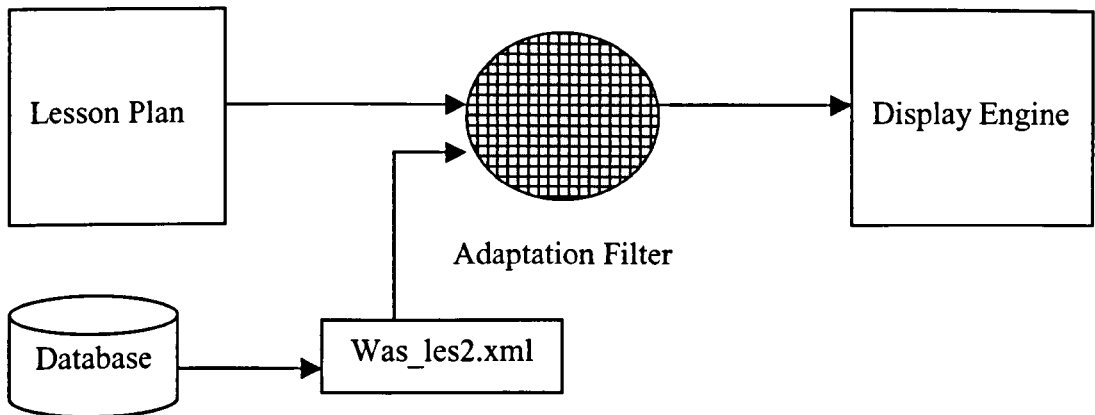


Figure 7.4 WHURLE New Infrastructure - the adaptation filter filters the contents of lessons' plans based on the knowledge levels of users with respect to the associated domains, and then pass the final adapted contents to the display engine to be displayed to the end users.

For example, let us consider the introduction to the html lesson plan in the Hybrid Model chapter (Chapter VI). Figure 7.5 shows a part of it.

```

<lesson plan>
.....
<level name="intro-web-basics" title="The Basics of HTML">
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb008</chunk></page>
  <level name="intro-web-tech" title="Important computing technical details">
    <page><chunk domain="general" stereotype1="" stereotype2="">introweb009</chunk></page>
    <page><chunk domain="general" stereotype1="" stereotype2="">introweb010</chunk></page>
  </level>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb011</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb012</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb015</chunk></page>
</level>

<level name="intro-web-tags" title="Simple HTML Elements">
  <page><chunk domain="html" stereotype1="beg" stereotype2="int">introweb016</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb017</chunk></page>
  <page><chunk domain="html" stereotype1="beg" stereotype2="">introweb018</chunk></page>
.....
</lesson plan>
  
```

Figure 7.5 Adaptive Lesson Plan - a part of an adaptive lesson plan about introduction to HTML

From the above figure, as explained in the Hybrid Model chapter, each level may nest other levels. In addition, each level may embrace pages and each page may hold one or more chunks. We find that each chunk has three attributes: a) domain, which

holds the name of the associated domain or “general” in case that chunk has no specific domain and should be displayed to all users; b) *stereotype1*, which holds a stereotype class for the associated domain; c) *stereotype2*, which also holds a second stereotype class for the same domain.

Whurle-filter.xsl behaviour depends on a session variable that indicates if the system is in the adaptation mode or not, as follows:

- Adaptation Mode is off: in that case, the filter will not be active and will pass all chunks to the display engine.
- Adaptation mode is on: in that case, *whurle-filter.xsl* checks the domain attribute. If it contains the value “general” then the chunk will be passed to the display engine. On the other hand, if the domain attribute holds the name of a domain, the filter will check for the session variable whose name is the same as the name of that domain. Afterwards it checks for the value that variable holds and compares it with *stereotype1* and *stereotype2* attributes. If any of them matched with the value of that variable, then the chunk would be passed to the display engine, otherwise it would not. This passing process acts as UNIX pipeline and XML pipeline concept, where the output of one programme is the input of another. Therefore, the output of the adaptation filter is the input for the display engine to present the chunks that passed the filter as they suit a user’s knowledge level [Zakaria et al. 2003].

Another factor that affects the behaviour of the adaptation filter is the type of lessons. If a lesson is of the open or new type, then a new level will be added to the hierarchy of that lesson. That level is called quiz, through which users find a link to the final quiz associated with that lesson. On the other hand, if a lesson is of the finished type, thereby that level will not be added. It is important to notice, as described in Chapter VI, when a user accesses any kind of lessons, a dynamic link appears at the tool bar, which is called “History”. That link is responsible for providing lessons that a user has finished in a raw format, i.e. without adaptation. In fact, the *history.xml* file is responsible for communicating with the database and getting all lessons that a user has finished. In addition, it switches the adaptation engine off for those lessons. On the other hand, the *history.xsl* file is responsible for displaying a table in a separate page to users, containing links to finished lessons (in a raw format) in addition to the score they got in each lesson, and the dates at which they finished those lessons. Thus, users could revise the full information in those lessons; also, they will have

their personal library that may include different subjects, which may be used as references if needed. The full code for *history.xml* and *history.xsl* is provided in Appendix A.

7.4.3 Quiz engine

The quiz engine plays an important and crucial part in the adaptation process. As explained in the Hybrid Model chapter, through quizzes, the system could infer users' knowledge level with respect to the domains associated with lessons. The quiz engine is composed of three parts: Quiz, Auto marking engine, and Upgrade engine, as shown in Figure 7.6. Each part of the quiz engine has its own files. Thus, each of them in more detail will be described individually.

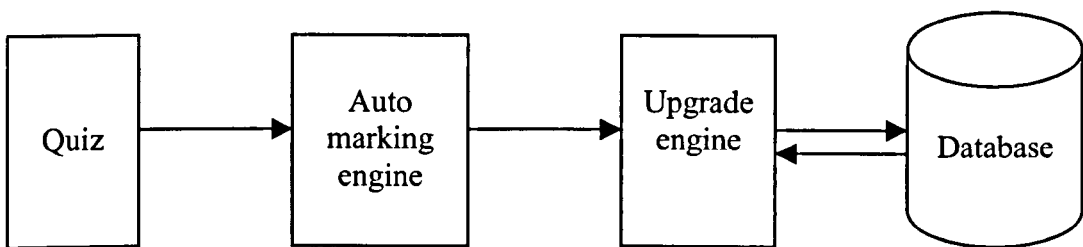


Figure 7.6 Quiz Engine Components - the quiz engine is composed of three parts. a) **Quiz**: this is the quiz that a lesson author is creating to test the knowledge of involved users with respect to the domains associated with the lesson. That quiz could be a pre-quiz or a post-quiz. b) **Auto marking engine**: that engine auto marks the results and analyses the answers to obtain the score a user attained with respect to every associated domain and with respect to the overall quiz. c) **Upgrade engine**: compares the new knowledge levels with those in the database and updates the database with the new levels, if any.

7.4.3.1 Quiz

Through a quiz, lessons' authors could find a way to know the knowledge level of the involved users before accessing lessons, so the system can adapt the contents of these lessons to each individual user's knowledge level, or can know what these users have learned from a particular lesson if a quiz is a post-quiz. In fact, there are two kinds of quizzes: pre-quiz – users could take that kind before accessing a lesson, and post-quiz – users must take it after they finish any lesson. The code in Figure 7.7 shows a part of a quiz file.

From Figure 7.7, it can be seen that the quiz file is composed of five parts:

- **Quiz Name**: holds the name of the quiz to be displayed on the screen for users.
- **Type**: the type of the quiz identifies it either as a pre-quiz (pretest) or a post-quiz (posttest). Depending on the type, the upgrade engine behaves differently as will be explained later.

- Domains: through that part, lessons' authors could identify the involved domains. The top-level tag of that part called "domains", which has an attribute called "num". That attribute holds the n number of the involved domains. Thus, inside that level there is an n number of domain tags. Each domain tag holds the name of a domain in addition to two other attributes:
 - a) id: that attribute holds a unique ID that a lesson author gives for that domain.
 - b) qid: that attribute holds the IDs of questions that represent that domain.
- Mark-base: that part identifies the score that a user will get each time he/she answers a question correctly. For example, if the mark-base is 5, and a user has answered three questions correctly out of five, then his final score will be $5 \times 3 = 15$ out of 25. Thus, it is important to balance the score with the knowledge value scale. As explained in Chapter VI, the knowledge value scale ranges from 1 to 10, and each value represents 10 score numbers. Therefore, lessons' authors have to make sure that if a user has answered all questions, with respect to a certain domain, correctly his/her score must be at maximum equals 100.

```

<quiz>
<quiz-name> Web Technologies Quiz 1</quiz-name>
<type>pretest</type>
<domains num="4">
  <domain id="1" qid="1,3">chemistry</domain>
  <domain id="2" qid="1,2,3,4">biology</domain>
  <domain id="4" qid="1,3,5">Physics</domain>
</domains>
<mark-base>20</mark-base>
<questions>
  <question type="choice" media="text" id="1">
    <question-text> Single Choice - True/False </question-text>
    <answer type="no" id="1">first choice</answer>
    <answer type="no" id="1">second choice</answer>
    <answer type="yes" id="1">Third choice</answer>
    <answer type="no" id="1">Fourth Choice</answer>
  </question>
  <question type="multiple" media="text" id="2" choices="2.1,2.2">
    <question-text>Choose the best answer(s) </question-text>
    <pic caption="caption">image file name</pic>
    <answer id="2.0">first choice</answer>
    <answer id="2.0">second choice</answer>
    <answer id="2.1">Third choice</answer>
    <answer id="2.2">Fourth Choice</answer>
  </question>
  .....
  <question type="choice" media="image" id="5">
    <question-text> Single Choice - True/False fifth choice</question-text>
    <answer type="no" id="5">steel/st-down.gif</answer>
    <answer type="no" id="5">steel/st-up.gif</answer>
    <answer type="yes" id="5">steel/st-right.gif</answer>
    <answer type="no" id="5">steel/left.gif</answer>
  </question>
</questions>
</quiz>

```

Figure 7.7 Quiz File - this is a part of a quiz that was originally composed of five questions. There are two kinds of questions: a) choice: single choice questions; b) Multiple: multiple choice questions.

They can do that by compromising between the number of questions and the mark-base. For example if a quiz is composed of 20 questions and involved two domains (10 questions for every domain), then the mark-base should be at maximum 10, as in $10 \times 10 = 100$.

- Questions: the questions part embraces the questions that a lesson author wants to ask. The “question” tags below the top level tag of this part, which is called “questions”, is composed of a different number of attributes depending on its type attribute, also embracing other child tags. Thus, there is two types of questions:
 - Choice: this kind of question means choosing only one answer from the other possible answers;
 - Multiple: this kind means to choose more than one answer from the other possible answers.

If a question is of the choice type, then there is other two other attributes:

- a) media: that attribute holds the type of answer either being a text or an image answer. It is important to notice that mixing between text and image in the answers of the same question is not allowed.
- b) id: this is a unique id for every question, which is used as a reference for a question in the qid attribute in the “domain” tag.

Following that tag, another child tag that is called “question-text”, which holds the question text itself. After the “question-text” tag may come the “pic” tag. That tag is responsible for displaying an image as a part of a question by holding either the path or the name of the image file. The “pic” tag holds one attribute that is called “caption”, which could hold the caption of the associated image or could be left empty, if no caption is required.

Afterwards come the “answer” tags, as each of which holds a different answer. In addition, each of them holds two attributes:

- a) type: that attribute either holds a “yes” value, which indicates that this answer is the right one, or a “no” value, which indicates that this answer is a wrong one.
- b) Id: the id of the question that this answer belongs to.

In questions of type multiple, the “question” tag holds one more attribute than choice questions, which is called “choices”. That attribute holds the id of right answers. Each answer of a question of type multiple holds only one attribute

called “id”. That id is composed of two parts separated by a dot. For example, the second block of questions in the questions part in Figure 7.7, answer tags hold ids with values “2.0”, “2.0”, “2.1” and “2.2”. It could be observed that the first number on the left is 2 and it doesn’t change; that is because this is the id of the question that these answers belong to. The right number, which varies between 0, 1 and 2, changes. If that right number is 0, this means that this is a wrong answer, but if it has another value, such as 1 and 2, this means that this is the id of that correct answer. The reason for that is because the auto-marking engine, in the case of questions of multiple type, checks for the ids of coming answers and questions. Thus, if it finds that the right side number is 0, it does not compare the whole id with the set of ids in the choices attribute that that question tag holds.

The *quiz-results.xsl* file is responsible for the display style of questions to the final users; also, it is responsible for delivering answers to the auto-marking engine. The full code can be found in Appendix A. In addition, a snapshot of the quiz screen (of the experiment described in Chapter VIII) can be found in Appendix C.

7.4.3.2 Auto-marking engine

After a user finishes a quiz and answers its question, *quiz-results.xsl* directs answers to *result.xml*, which is responsible for marking the quiz. Thus, *result.xml* is the heart of the auto-marking engine. The technical mechanism of that engine can be summarized as follows:

- The auto-marking engine starts by reading the number of involved domains in the quiz, the IDs of those domains, and the IDs of the correct answers with respect to every involved domain, in addition to the domains’ names from the quiz.
- Next, it starts building a two-dimensional vector, where the first column is the names of the involved domains, and the remaining columns contain the IDs of involved questions with respect to every domain. For example, the first row may look like: Chemistry, 1, 2.3, 2.5, 6, and the second may look like: Biology, 1, 3, 4.3, 4.6, 4.8, 7, and so forth.
- Afterwards, it starts by getting the questions’ IDs into one dimension vector, after removing any duplication. Moreover, it builds a sister vector to hold the result of each corresponding question.

- Later, the auto-marking engine starts reading the IDs of the coming answers. At that point, the auto-marking starts. Thus there are two cases depending on the type of questions:
 - Single choice questions: in case of single choice questions, the ID of the question will be the same as its associated answers. Thus, the engine starts to look for the type of the coming answer, to check if it is “yes”. Therefore, it is a right answer, and thereby the value of the base-mark will be added for that question to the corresponding results vector, otherwise 0 will be added and so forth.
 - Multiple-choice questions: in that case, the engine starts by reading the IDs of correct answers. If a user has answered all of them, then the value of the base-mark will be added for that question to the corresponding results vector, otherwise 0 will be added and so on.
- After building a vector that holds the IDs of questions, and another sister vector that holds the score of each question, the scoring process starts, by matching the questions’ IDs with respect to every involved domain in the two-dimension vector with those in the questions vector, and adding scores from the corresponding results vector with respect to every associated domain.
- Finally, the engine puts each domain and its score in a session variable; it also adds the overall score to a session variable to be read by the upgrade engine.

The full code of *result.xml*, and *result.xsl* that is responsible for auto-directing to the upgrade engine, is found in Appendix A.

7.4.3.3 Upgrade engine

At that stage, the auto-marking is completed, and the knowledge level stereotypes are needed to identify the knowledge level of users with respect to the involved domains. Thus, the role of the upgrade engine begins. In fact, *uengine4.xml* is the heart of that engine. Thus, the mechanism of that engine could be summarized as follows:

- *uengine4.xml* reads the names of involved domains and their scores from session variables in addition to the overall score, which are created by the auto-marking engine.

- Afterwards, the upgrade engine initiates a connection with the database, to get the stereotype class of every involved domain by comparing its score with the knowledge scale and then with the stereotype scale. Moreover, it reads the stored knowledge level (before taking the quiz) of every involved domain from the database.
- Later, a comparison starts between the old knowledge level and the new one with respect to every domain. If there is a difference it upgrades the database with the new value, otherwise no upgrade takes place. For example, if there is a domain called HTML, and from a quiz a user got an intermediate level. Therefore, if the old knowledge level of that user with respect to that domain, which is stored in the database, is beginner or advanced, thereby the database will be upgraded to the new level, which is intermediate.
- After this upgrade, another upgrade takes place, which is the status of the lesson to which the quiz is associated. If that lesson is a new lesson and the quiz is of the pre-quiz type, therefore its status will be changed to open. On the other hand, if the quiz is of the post-quiz type, and the lesson is of the open type, therefore:
 - a) The lesson status changes to finished; and
 - b) The former knowledge levels with respect to the involved domains are stored in the *old_lvl* table in the database.

The full code of *uengine4.xml* and *uengine.xsl* (which is responsible for displaying the final results) is found in Appendix A. Moreover, an example for the result screen is in Appendix C.

7.5 Administrative Tools

From the earlier sections, it has been seen how the system components communicate together and with the database. Thus, it is important to get all of these technicalities hidden from the end user, such as system administrators and lessons' authors. Therefore, two Administrative tools had been built: one for registering students, and one for registering new lessons.

7.5.1 Students' registration tool

It is known from the authentication component sub-section that the system needs to identify users by recognizing certain information about them, such as their ID, user name, password, etc. Moreover, it is difficult leave the task of registering every user's information to the database on the shoulders of system administrators. Because of that, this tool had been created. In fact, that tool is composed of four files: *streg.xml*, *streg.xsl*, *studreg.xml*, and *studreg.xsl*. Technically, the mechanism of that tool could be summarized, as follows:

- *streg.xml* and *streg.xsl* are responsible for creating a registration form that students have to complete. The following are the required information to be supplied:
 - Student ID: a unique ID that each student has, such as his/her university card number, or his/her student ID.
 - First Name: the first name of the student/user.
 - Last Name: the last name of the student/user.
 - User Name: any user name a student chooses.
 - Password: any password a student chooses.
 - Confirming Password: a reconfirmation of the entered password.
 - Category: this is a drop down menu list of categories, which is created automatically by means of *streg.xml*. That file communicates with the database and gets all categories listed in the category table. Thus, users have to choose the category that suits their cases. For example, undergraduate student, postgraduate student, etc.
 - Style: this is also a drop down menu list, which contains a description for the configuration files. Similar to the Category list, the *streg.xml* communicates with the database and gets the description of all configuration files from the conf table. Thus, users may choose the configuration they like. For example, the description could point to the skin that a configuration file may hold such as, Modern Style, Egyptian Style, etc. It is important to notice that, if a user wants to change the configuration/skin in the future, he/she has to contact the system administrator for that, as they cannot do it by themselves. This process needs the administrator to manually change that from inside the database.

The reason for that is that some students may like to change the configuration/skin each time they access the system, or many times during the same session, which in turn may result in an overload on the server, especially if a big number of users are using the system at the same time.

- After a user fills in this information and sends it, the *streg.xml* (before directing this information to the *studreg.xml*) checks that the user has entered all the required information, Additionally it checks that the entered password matches with the password that the user has re-entered. It does that by means of a JavaScript code within the *streg.xml* file.
- When *studreg.xml* receives the entered information from the *streg.xml*, it checks for the combination of the user name and password that the user has entered with those in the database. If that combination is found, then the user is asked to re-enter a new user name. On the other hand, if the combination is not found, then, this information will be inserted into the database. In fact, *studreg.xml* is responsible for displaying messages to the user. Therefore, in case the information is registered, the user will see “registered” on the screen, in addition to a link to the login screen to start accessing the system. The reason in offering a link and not getting that user directly to the system is because it may be that the user is just interested in only registering him/herself. Thus, in that way, there will be no overhead on the server by not doing extra processes the user is not explicitly asking for.

The full source code for *streg.xml*, *streg.xml*, *studreg.xml*, and *studreg.xml* may be found in Appendix A. Also, a snapshot of the registration screen can be found in Appendix C.

7.5.2 Lessons’ registration tool

To register a new lesson to the system, certain information related to that lesson is required, such as its name, associated domain(s), pre-quiz and post-quiz names. The lessons’ registration tool was build up for the same reason of building the Students’ registration tool. That tool is composed of four files: *ladmin.xml*, *ladmin.xml*,

ladmin2.xml, and *ladmin2.xsl*. The technical mechanism of that tool could be briefed as follows:

- *ladmin.xml* and *ladmin.xsl* files are responsible for creating a form for lessons' authors to register their lessons into the database. That form requires the following information to be supplied:
 - Lesson name: the name of the lesson file.
 - Involved domains: this item is composed of a list of check box items, as each one of them holds the name of a domain. Technically, *ladmin.xml* communicates with the database and gets all registered domains in the domain table, then, *ladmin.xsl* displays each of them in a check box form, where lessons' authors may choose among them.
 - Category: this is a drop down menu list. Through this list, lessons' authors have to choose the category that their lesson serves. For example, if that lesson is for undergraduate students, postgraduate students, etc. *ladmin.xml* gets from the category table all the involved categories, and *ladmin.xsl* displays them in the form of a list menu.
 - Name of pre-quiz: here lessons' authors either enter the name of the pre-quiz file or 0 (zero) in the case there is no pre-quiz for that lesson.
 - Name of post-quiz: through that field, lessons' authors supply the system with the name of the post-quiz file. It is important to notice, that the name of the post-quiz and pre-quiz could include a full path, if they were not in the same directory as *ladmin.xml* and the other associated files.
- After filling in the required information, *ladmin.xsl* activates *ladmin2.xml*. That file checks if the name of the lesson is found in the database or not (specifically in the *lesson_flow* table). If it is found, *ladmin2.xsl* asks the user to rename its lesson's file and re-enter it again, but if not, the user sees "registered" on the screen.

The full code of *ladmin.xml*, *ladmin.xsl*, *ladmin2.xml*, and *ladmin2.xsl* could be found in Appendix A. In addition, a snapshot for the tool screen is provided in Appendix C.

7.6 Summary

Throughout this chapter, full technical details about the infrastructure of the WHURLE-HM system is given. Technically, this infrastructure is composed of the MySQL database, which holds 13 tables, and 12 files that serve the adaptation process, in addition to eight files that serve two Administrative tools. Moreover, the 12 files have been categorized according to their role in the system into: files that serve the authentication part of the system, files that control the adaptation process, and files that build the quiz engine and its components. The chapter is divided into five sections:

- **Introduction:** a brief introduction that establishes the nature of this chapter
- **Implementation Algorithm:** this section describes the overall architecture of the system. In addition, a flow chart that graphically describes the system mechanism.
- **Database design:** a description for every table in the database and its role in the adaptation process.
- **System components:** that section describes every component of the system and how these components cooperate together to carry the adaptation process.
- **Administrative tools:** this section describes two Administrative tools, which are especially built to hide a major part of complexity from system administrators, lessons' authors, and end users (students).

Chapter VIII: User Trial

8.1 Introduction

In the previous two chapters (Chapters VI and VII), the Hybrid Model and its implementation in WHURLE have been explained. In this chapter a user trial will be described, where the system was used in support of a third year “Biological Anthropology” module delivered in the School of Life and Environmental Sciences of the University of Nottingham. This trial consisted of the detailed monitoring of the activities of 32 students using the system during a period of about a month. The main goal of this chapter is to demonstrate that the Hybrid model could be implemented in adaptive educational hypermedia systems such as WHURLE-HM. Moreover, it can be used to deliver educational materials to users based on the idea of mapping the concepts presented in these materials to semantic domains, and measure users’ knowledge with respect to these involved domains. Furthermore, during this chapter an attempt has been made to show if the model has contributed (even in a small scale) to students’ knowledge with respect to an educational curriculum. It is important to stress that this chapter does not present an evaluation of the model, but rather it is a user trial. In Chapter IX (under further research section) a description for how to create an evaluation for the model is described.

8.2 Experiment Design

8.2.1 Lesson plan design

A lesson plan called “Anthropology Revision Guide” was created for this module by the module convenor (Dr. Peter Davies). This was intended for use as a revision guide before the exams, and it was presented to the students in the last month of the course. This lesson plan utilised two semantic domains: Fossils and Functions – as classified by the author. The lesson is composed of two categories of information: a) information that should be seen by all students regardless of their knowledge level with respect to the Functions and Fossils domains; b) information that depends on users’ knowledge level with respect to the Fossils domain and Functions domain. The code in Figure 8.1 illustrates a part of that lesson.

```

<level name="Discovery" title="Discovery">
  <page><chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-001</chunk></page>
  <page><chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-002</chunk></page>
  <page><chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-003</chunk></page>
  <page>
    <chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-004</chunk>
    <chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-005</chunk>
  </page>
  <page >
    <chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-006</chunk>
    <chunk domain="60" stereotype1="beg" stereotype2="">primate-ad-007</chunk>
  </page>
</level>

<level name="Description" title="Description">
  <page><chunk domain="60" stereotype1="int" stereotype2="">primate-ad-008</chunk></page>
</level>

<level name="Dating" title="Dating & Naming">
  <page><chunk domain="general" stereotype1="" stereotype2="">primate-ad-009</chunk></page>
  <page><chunk domain="general" stereotype1="" stereotype2="">primate-ad-010</chunk></page>
  <page>
    <chunk domain="general" stereotype1="" stereotype2="">primate-ad-011</chunk>-
    <chunk domain="general" stereotype1="" stereotype2="">primate-ad-012</chunk>
    <chunk domain="general" stereotype1="" stereotype2="">primate-ad-013</chunk>
  </page>
</level>

```

Figure 8.1 An extract from the Anthropology lesson plan - Full details about lessons plans technical description are provided in Chapter VII.

As can be seen in the above code, only two knowledge levels are utilised: intermediate and beginner. Detailed instructions on to how to build a lesson plan are provided in Appendix D. Moreover, a snapshot of the lesson plan can be found in Appendix C.

8.2.2 Pre-quiz design

As described in Chapter VI and Chapter VII, the pre-quiz plays a crucial role in determining a user's knowledge level with respect to the domains utilised by a lesson plan. Therefore, a pre-quiz composed of 20 questions (supplied by the lesson author) was constructed. The questions were divided into two categories, according to the two knowledge domains in use. As explained in Chapter VII, there are two types of questions supported by the quiz engine: single choice questions and multiple-choice questions. In that experiment the single choice question type is used. Moreover, some of these questions were a combination of text and image. The code in Figure 8.2 illustrates part of that pre-quiz.

```

<quiz>
<quiz-name> Anthropology Pre-Quiz</quiz-name>
<type>pretest</type>
<domains num="2"><!-- important to set the number of involved domains -->
  <domain id="1" qid="1,2,3,4,5,6,7,9,10,11,12">fossiles</domain>
  <domain id="2" qid="14,15,16,17,18,19,20,21,22">functions</domain>
</domains>
<mark-base>5</mark-base>
<questions>
  <question type="choice" media="text" id="1">
    <question-text>Fossil bones are found only in igneous rocks </question-text>
    <answer type="no" id="1">True</answer>
    <answer type="yes" id="1">False</answer>
    <answer type="no" id="1">I don't know</answer>
  </question>

  <question type="choice" media="text" id="2">
    <question-text>Most fossils are made of stone, not organic matter</question-text>
    <answer type="yes" id="2">True</answer>
    <answer type="no" id="2">False</answer>
    <answer type="no" id="2">I don't know</answer>
  </question>

  <question type="choice" media="text" id="3">
    <question-text>Indirect 'traces' of ancient life such as footprints are fossilise</question-text>
    <answer type="yes" id="3">True</answer>
    <answer type="no" id="3">False</answer>
    <answer type="no" id="3">I don't know</answer>
  </question>

```

Figure 8.2 An extract from the Anthropology pre-quiz - Full details about quizzes technical description are provided in Chapter VII.

From the above code, it can be seen that, although there are nine questions dedicated to the functions domain and 11 to the fossils domain, the mark-base is only five. The reason for this is that only two knowledge levels are involved: intermediate and beginner. Therefore, the highest level will be intermediate. Thus, $5 \times 9 = 45$ and $11 \times 5 = 55$, which ensures that the maximum a user will be is intermediate. For a user to be intermediate he/she will be required to answer at least six questions correctly in any associated domain. At the time of the experiment, the knowledge scale and the stereotype scale was as follows:

- The beginner level: embraced scores between 0 and 30.
- The intermediate level: embraced scores ranging between 31 and 80.
- The advanced level: contained a range of scores between 81 and 100.

A decision was made for the post-quiz to consist of the same questions that were used in the pre-quiz, but presented on paper as a part of the module's assessment.

A guideline on how to create a quiz is provided in Appendix D. In addition, a snapshot of the pre-quiz is provided in Appendix C.

It is important to stress that the data presented here does not show any identifiable personal information about students in the course, to ensure their privacy.

8.3 Methodology

Due to technical reasons (such as network congestion and server overload), students were divided into two groups. The first group consisted of 22 students who were instructed to register for use of the system using the registration form, and then log in to the system, which would direct them automatically to the pre-quiz to solve. That process took about 30 minutes. After that, the second group, consisting of 23 students, passed through the same experience. It could be observed that the total number of students who registered in the tutorial time was 45 and one student registered later. Therefore, the total count of students was 46. From the 46 students only 32 students used the system, 12 did not and the remaining two students were excluded because they registered to the system more than once with different user names and passwords. Figure 8.3 shows the results obtained from the pre-quiz with respect to the two involved domains, which are Functions and Fossils.

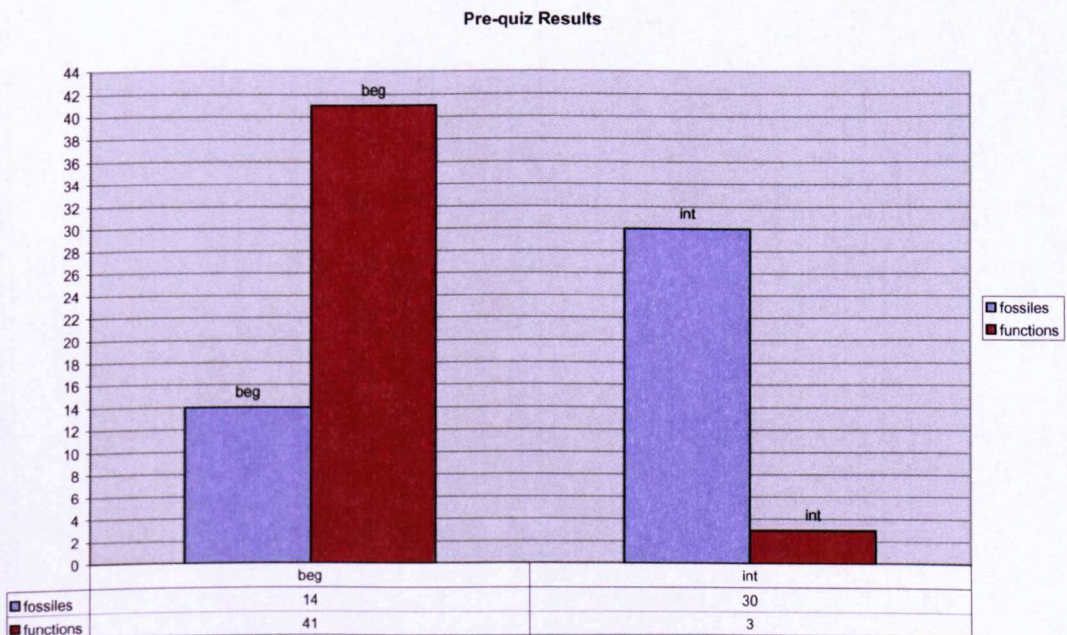


Figure 8.3 Pre-quiz Results – from the above figure it can be observed that the total number of students who were assigned as beg (beginner) in the Fossils domain is 14 and those in the Functions domain are 41. Furthermore, the number of students who were assigned to the int (intermediate) level in the Fossils domain is 30 and those in the Functions domain are 3.

To monitor the access of students to the system, an extra code was created, in the file that is responsible for checking user names and passwords, together with a database table called monitor. When a student accesses the login screen, that extra code retrieves the student's ID, and the machine date and time, and then stores them in the monitor table. In addition to this, the web server logs were also analysed. As a result, it was found that 32 students accessed the system prior to the day of the post-quiz, and 12 students registered but did not use the system. This means that now there are two groups, a group accessed the system and another one that did not. This thus provided an opportunity to try to find if the system had any effect upon the students' performance. It is important to notice that students were entitled to use other resources, such as books, lecture notes, etc., alongside the system, which was optional. Therefore, the difference between these two groups is the usage of the system. The focus of the experiment is on the students whose knowledge level, with respect to any involved domain, is upgraded from beginner (beg) to intermediate (int). Figure 8.4 shows users' new knowledge level after taking the post-quiz. It is important to notice that these results include students who have not accessed the system.

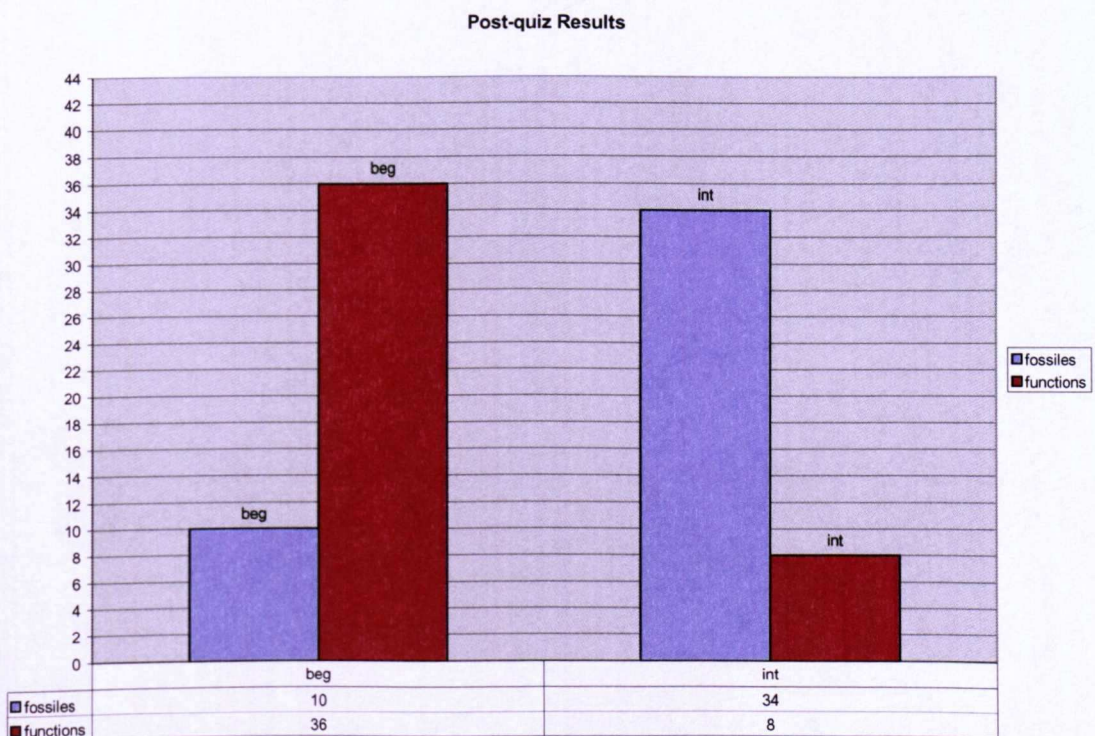


Figure 8.4 Post-quiz Results – it can be observed from the above chart that the students with beg (beginner) knowledge level in the Fossils domain are 10, and those in the Functions domain are 36. In addition, students with int (intermediate) knowledge level in the Fossils domain are 34, while those in the Functions domain are 8.

8.4 Data Analysis

To use a suitable statistical test, the distribution type of the data must be known [Tabachnick and Fidell 2001]. According to Bryman [Bryman and Cramer 1999], the most important distribution is the normal distribution, which is known by its bell shape curve. According to that bell shape, 50% of the cases (data) should lie in one side of the middle of that curve (mean), and the other 50% lies on the other side [Bryman and Cramer 1999]. To test the normality of the cases, two components should be measured: a) Skewness, which deals with the symmetry of the distribution, i.e. skewed cases are those whose mean is not at the centre of the distribution; b) Kurtosis, which deals with the peak of the distribution, i.e. distribution is either very peaked or very flat [Tabachnick and Fidell 2001]. The normally distributed data has the skewness and the kurtosis components equal or close to zero [Bryman and Cramer 1999, Tabachnick and Fidell 2001]. Moreover, if the skewness value of cases has a positive value that means that the cases are clustering to the left of the curve with a long tail at the right, while the negative value indicates the opposite. Furthermore, if the kurtosis value is more than zero that indicates that the curve is too peaked, while below zero indicates that the curve is flat [Bryman and Cramer 1999, Tabachnick and Fidell 2001].

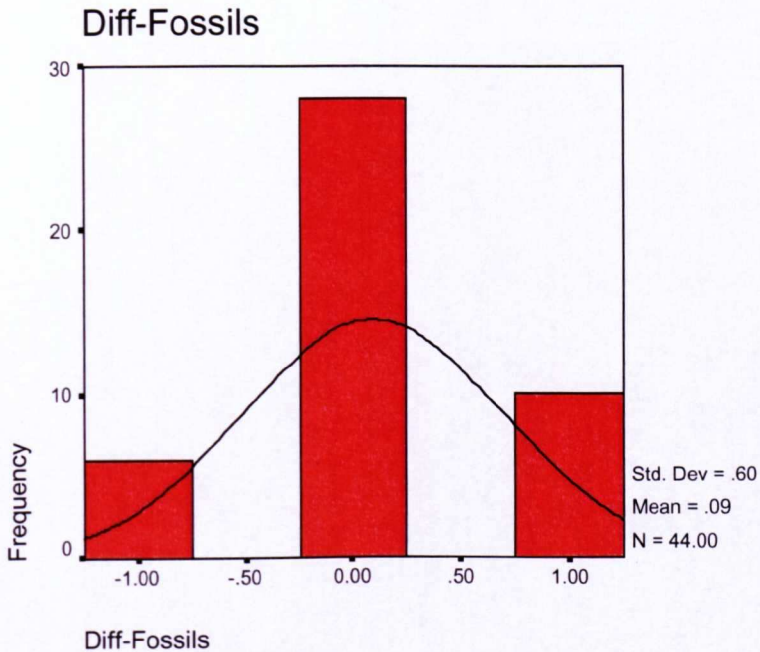
By using SPSS statistical package the normality of the data is measured as shown in the table in Figure 8.5.

		Diff-Fossils	Diff-Functions	log-num
N	Valid	44	44	44
	Missing	227	227	227
Skewness		-.033	1.155	-1.057
Std. Error of Skewness		.357	.357	.357
Kurtosis		-.121	3.051	-.927
Std. Error of Kurtosis		.702	.702	.702

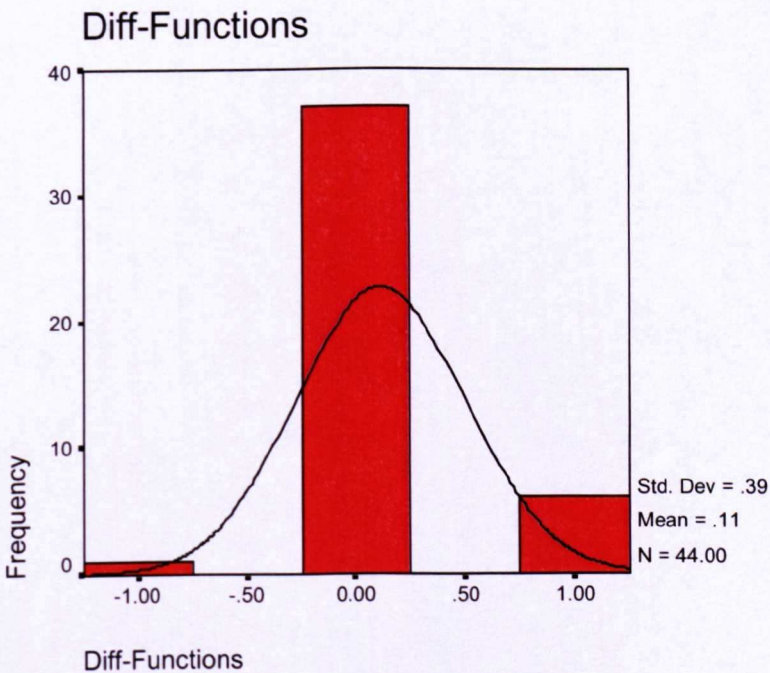
Figure 8.5 Normality Test – it can be seen from the table that three sets of data were measured: **Diff-Fossils** – the difference in knowledge with respect to the Fossils domain by subtracting the knowledge level the students got after the post-quiz with the knowledge they got after the pre-quiz; **Diff-Functions**: the same as the Diff-Fossils; **log-num**: it is a flag that indicates if a student accessed the system (1) or not (0).

By analysing the results in Figure 8.5, it could be perceived that the skewness of Diff-Fossils is close to zero (-0.033) and the kurtosis value is also close to zero (-0.121). Therefore, the cases in the Diff-Fossils columns are almost normally distributed. On the other hand, the skewness of the Diff-Functions is far from zero

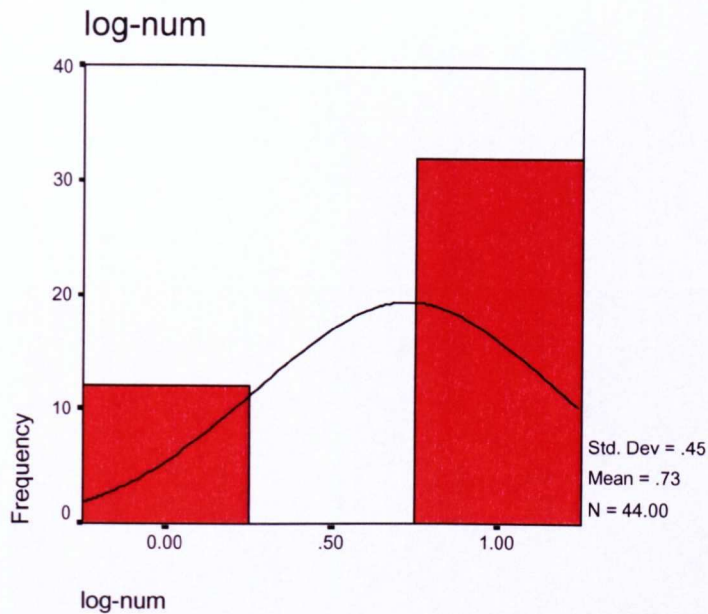
(1.155) and the kurtosis value also is far from zero (3.051). As a result, it can be seen that the data the Diff-Functions holds is not normally distributed. By repeating the same for the log-num, it could be observed also that it is not normally distributed as the skewness is far from zero, although the kurtosis value is in some way close to zero. Figure 8.6 shows the distribution of the three sets.



A



B



C

Figure 8.6 Distribution Graphs – in diagram A, the Diff-Fossils is almost normally distributed. In diagram B, the Diff-Functions is not normally distributed, as the curve is too peaked; also the curve is leaning to the left. In diagram C, the log-num is also not normally distributed as the curve is leaning to the right. Std. Dev: standard deviation (distance from the mean), Mean: average, N: number of cases.

According to Bryman [Bryman and Cramer 1999], in a case where data is not normally distributed and the sampled population is below 15 and not equal (students who accessed the system are 32 while those who did not are 12), it is more appropriate to use the nonparametric/distribution-free tests. Nonparametric tests are tests that do not depend on assumptions about the distribution form of the sampled population [Bryman and Cramer 1999]. Therefore, a nonparametric statistical test called the Mann-Whitney U test was chosen. According to Bryman, that test can be used to “compare the number of times a score from one of the samples is ranked higher than a score from the other sample” [Bryman and Cramer 1999, p 137].

The hypothesis that the test is based on is that there is a difference between the two groups with respect to their performance. The table in Figure 8.7 shows the result of the test.

	Diff-Fossils	Diff-Functions
Mann-Whitney U	176.000	183.500
Wilcoxon W	704.000	261.500
Z	-.494	-.353
Asymp. Sig. (2-tailed)	.621	.724

Grouping Variable: log-num

Figure 8.7 Statistical Results - measures the ranking in both Diff-Fossils and Diff-functions. The **Wilcoxon W**: is the sum of the ranks of the smaller group, which can be ignored. The **Z (Z-test)** gives the confidence level (**Asymp. Sig. (2-tailed)**) for the tested hypothesis, which is the output of the test the researcher is looking for.

From the above table, the Mann-Whitney U test is carried for both columns Diff-Fossils and Diff-Functions with respect to the logging flag. It is important to notice that to accept the presumed hypothesis the significance level must be less than 0.05 (the 0.05 means that the accidental rejection of the null hypothesis is no more than 5% of the time when it is true), which is a standard measure of acceptability [Tabachnick and Fidell 2001].

The significance level (**Asymp. Sig. (2-tailed)**) given to the presumed hypothesis against the null hypothesis (there is no difference between the two groups of users) is more than 0.05 (greater the 5%) for both columns, which suggests the rejection of the presumed hypothesis. Therefore, statistically there is no significant difference in the performance between users who accessed the system and those who did not.

The reason for the obtained analysis is not statistically significant could be attributed to the small size of the sampled population, the system being optional, and the presence of networking congestion in addition to the server hardware limitation. The latter factor (networking problems and hardware limitations) had a profound impact over the number of times the students accessed the system as shown in Figure 8.8.

Times of Access

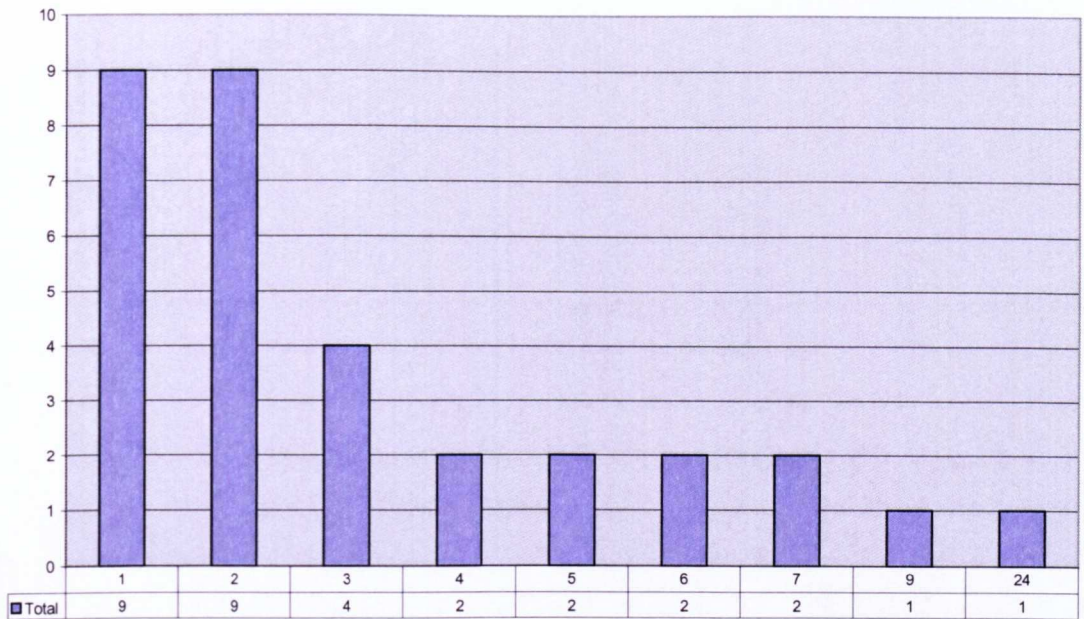


Figure 8.8 Times of Access – from the above figure it can be observed that the total number of students who accessed the system only once is nine, and those who accessed it twice is also nine, and who used it three times is four, four times is two, and so forth.

The following table converts the number of accesses presented in Figure 8.8 into percentages as follows:

Access times	1	2	3	4	5	6	7	9	24
Students' percentage	28.125%	28.125%	12.5%	6.25%	6.25%	6.25%	6.25%	3.125%	3.125%

From the table it can be seen that the majority who accessed the system lie in the interval of one and two times. By including students who accessed the system more than two times only, the resulting sample will be 43.75% (14 students), and this percentage is less than half of the group who accessed the system. Therefore, it is difficult to predict any precise correlation between the times that users accessed the system, and the changes that occurred in their knowledge level.

The following section will discuss the questionnaires that were given to the students to answer and how those answers suggest the success of the system and the model to an extent.

8.5 Students' opinion

In light of the factors described in the former section, extra data was needed in order to show if the system had any positive influence. Therefore, six questions, five quantitative and one qualitative, were handed to students to answer. Out of 32 students who accessed the system, 30 students answered the quantitative questions and 23 students answered the qualitative. It is important to notice that all the answers provided were anonymous.

8.5.1 Quantitative questions

Five questions were designed to explore students' opinion with respect to the system. 30 students answered the following questions:

1. I found the revision aid useful.
2. Using it encourages me to probe topics more deeply.
3. The revision aid covered topics I already knew well.
4. I would like similar revision aids used in other modules/courses.
5. The revision aid did not really provide me with enough detailed information.

Students had a scale of five categories to choose from that ranged between strongly agree to strongly disagree.

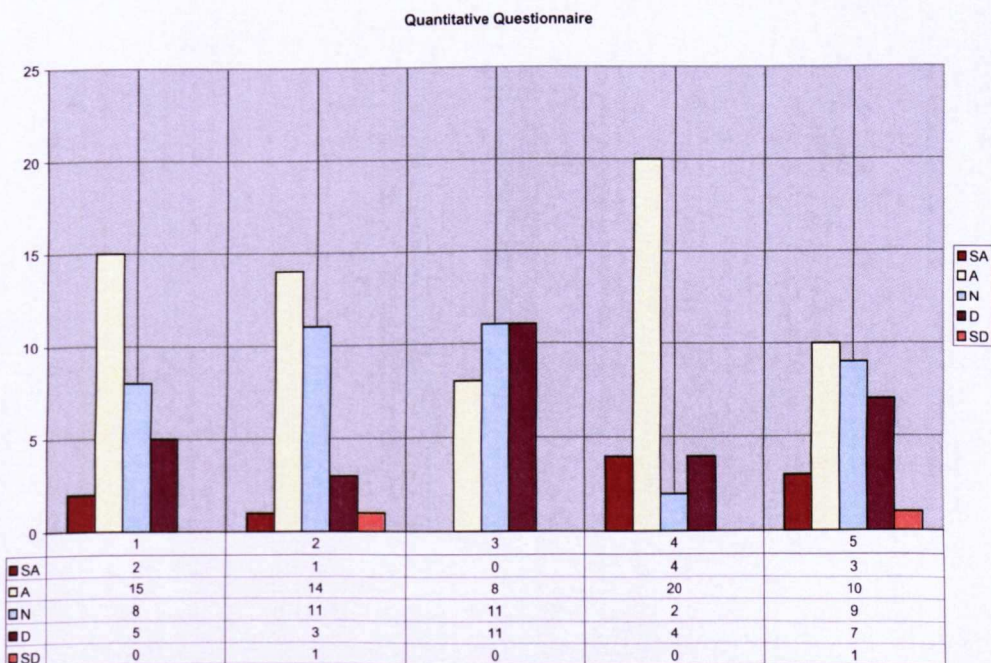


Figure 8.9 Quantitative Questions Results – the X-axis of the graph represents question numbers and the Y-axis represents the number of students. SA stands for Strongly Agree, A Agree, N Neutral, D Disagree, and SD Strongly Disagree.

It is important to stress that these five questions were integrated with other questions that evaluate the whole module. Therefore, it was preferable to follow the same answering categories.

Figure 8.9 shows students' answers to the questionnaire. It can be observed that the number of students who found the system useful (question number 1) is 15, this represents 50% of the group that answered the questions, which is the highest evaluation in the first question. For the second question, 14 students, which represents 46.6%, agreed that the system helped them to understand topics more deeply; also, this is the highest response in the second question. Furthermore, it could be observed that 11 students (36.6%) agreed that the system covered topics they were not familiar with (by disagreeing (D) with the third question). However, the same percentage was neutral, and this maybe because they found some information related to any of the two-presented domains more familiar than the other. Therefore, the response to the question provides some evidence that the system has successfully carried out the adaptation process according to the author's intentions. In the fourth question, which is a rider to question number 1, 20 students (66.66%) agreed that they would like to see the same system used with other modules or courses, and this is the highest response percentage in the overall questionnaire. Therefore, the system was successful, which means the model was also successful. Finally, the fifth question related to the amount of detailed information in the material, 10 students (33.33%) agreed that more detailed information was needed. Nevertheless, 30% of students remained neutral. Therefore, there is a very close percentage between students who agreed and students who were neutral. However, students who agreed that more details were needed support the notion that more detailed information maybe required to increase students' knowledge. Therefore, this may be another reason why there is no significant difference between students who used the system and between those who did not.

8.5.2 Qualitative question

An open question was designed for students to give their opinions and suggestions regarding the system and the content. The question was:

- If you used the online Revision Aid, we would welcome your comments and suggestions.

Only 23 students answered the question and they represent 71.87% of the whole group that accessed the system.

By categorising students into three categories:

- a- Students with positive response (found the system useful).
- b- Students with negative response (found the system not useful).
- c- Students could not access the system.

The following table summarises the percentage of students in each category

Categories	A	B	C
Percentages	73.91% (17 students)	6.25% (2 students)	17.39% (4 students)

The table highlights that a high percentage of students found the system useful and a low percentage found the system not useful. Students who found difficulty in accessing the system as seen from the table are 17.39% (4 students). By analysing the response of the 4 participants, it was found that two of them stated that the system crashed frequently and was very slow; whereas of the other two, one of them could not access because he/she could not get an authentication to enter despite the fact he/she is registered. This feature is normal if the user registered his/her user name and password in different letter cases (upper case or lower case) than that he/she used to log in to the system (i.e. the authentication system is case sensitive). Full quotes can be found in Appendix E categorised as A, B and C.

A common response that is found among students is that the system was very slow, and this is related to external factors such as network congestion coupled with the time of access. Therefore, this common ground justifies why the researcher could not perform any kind of correlation between the times of access and the change in knowledge levels.

Another common response from users related to the amount of presented information. Around 47% of students who found the system useful suggested more

detailed information needed to be presented. This provides support for the suggestion that more in depth information maybe required.

8.6 Conclusion

From the user trial, it has been demonstrated that the Hybrid model is applicable to adaptive educational hypermedia systems, such as WHURLE-HM, and it is possible to be used in an educational setting, which is the primary goal of this experiment. Moreover, suggestive evidence has been found to imply that the system (and in turn the model) is successful in terms of adapting materials to users' knowledge levels, despite technical difficulties, small sample sizes and the need for more detailed information. However, the extent to which this success can be measured requires a full evaluation, which should be carried out in a more controlled environment and with a higher number of participants as described in the discussion chapter (Chapter IX) under the further research section. In addition, the limitations that faced this trial are described under the limitation section in Chapter IX.

8.7 Summary

This chapter presents an experiment that took place in the School of Life Science at the University of Nottingham. The WHURLE-HM system, which implemented the Hybrid Model, served as a revision guide for a module called Anthropology. The material presented was written by the tutor of that module Dr. Peter Davis.

The Chapter is organised as follows:

- **Introduction:** description for the chapter and its goal.
- **Experiment Design:** this section explains the experiment with respect to the presented material and the pre-quiz.
- **Methodology:** this section describes how the experiment was carried out.
- **Data Analysis:** a statistical analysis for students' knowledge level related to the presented material is described here.
- **Students' opinion:** analysing the response of students in relation to their experience with the system is provided in this section.

- **Conclusion:** summary of the results obtained from the user trial.

Chapter IX: Discussion

9.1 Introduction

During the course of this research, the Hybrid Model [Zakaria and Brailsford 2002, Zakaria et al. 2002] that is a user model for adaptive hypermedia in education has been developed, which is a hybrid of two major approaches in user modelling. That model has been experimentally implemented and validated using the WHURLE [Brailsford et al. 2001, Brailsford et al. 2002] system as a vehicle and as a result WHURLE-HM [Zakaria and Brailsford 2002] has been produced. In this chapter, comparisons are made between this model, and those that are utilised by various other adaptive educational hypermedia systems. In addition to this, a comparison between the adaptation techniques that are used by WHURLE-HM, and those used by other systems are described.

9.2 Why the Hybrid Model?

The user models utilized by the systems explained in the survey chapter (Chapter IV), can be classified into two categories: a) systems that use the overlay model, such as AHA! [De Bra et al. 2002], ELM-ART [Weber and Specht 1997], NetCoach [Weber et al. 2001], Interbook [Brusilovsky et al. 1998] and TANGOW [Carro et al. 2000]; and b) systems that mix the overlay model with the stereotype model, such as CHEOPS [Negro et al. 1998], Hypadapter [Hohl et al. 1996] and Metadoc [Boyle and Encarnacion 1994].

One important feature that all of these systems have in common is their use of the overlay model. As described in Chapter II, the overlay could be in the form of “concept-value” pairs. All of the mentioned systems use a measurement of this type, although with different parameters. For example, in NetCoach, the concept is considered to be learned (value) if it is tested or inferred. It should be noted that those systems are capable of utilizing only a single knowledge domain at any one time. An exception to this is provided by the Metadoc system. As described in previous chapters, Metadoc has tried to solve the multi-knowledge domain problem to a certain extent, by means of stereotyping the involved knowledge domains independently. Regardless of this, the user model architecture used by Metadoc

cannot differentiate between users with different backgrounds and knowledge levels. Metadoc categorized the knowledge level about Unix/AIX and general computer concepts into different knowledge levels each. If that system, with its current architecture is applied to users or students of two different educational levels – such as postgraduate students and first year undergraduate students – and both of them share the same knowledge class; the system will not be able to offer each of them the right kind of adaptation because there is no method to use to differentiate between users of the same stereotype.

The Hybrid Model modified the overlay model in order to measure users' knowledge levels with respect to semantic domains and not with respect to topics' concepts, as semantic domains embrace different topics and topics are composed of different concepts as explained in Chapter I. Therefore, instead of the classical measurement of the knowledge through "concept-value" pairs, it becomes a "Domain-value" pair [Zakaria and Brailsford 2002, Zakaria et al. 2002]. As a result, The Hybrid Model is capable of managing multiple knowledge domains concurrently. Furthermore, systems that used the stereotype model mixed with the overlay model, such as CHEOPS, Hypadapter and Metadoc, classify the users' knowledge level with respect to the involved concepts into different classes such as novice, beginner, intermediate and advanced. However, the Hybrid Model uses the stereotype to classify users' knowledge with respect to semantic domains. Therefore, this type of stereotyping can be called global stereotyping, as it is not dedicated to any particular topic. Additionally, a very important characteristic of the Hybrid Model is that, in addition to classifying users' knowledge, it classifies users with respect to their educational level and profession. That classification, as described in Chapter VI, helps the model to differentiate between users who share the same knowledge class of the same domain and provides to each of them the appropriate kind of personalization.

The Hybrid Model has principle advantages over other user models utilised by the described systems, and these are:

- a) The Hybrid Model handles the overlay model in a way that can measure users' knowledge with respect to semantic domains rather than educational curricula (this is described in detail in Chapter VI). This provides the Hybrid Model with the capability of measuring users' knowledge globally (i.e. how much does a user

know about semantic domains which are involved in an educational curriculum) rather than locally (i.e. how much does a user know about an educational curriculum), which has not happened before, as far as the researcher is aware. As a result, the model could maintain user's knowledge about the involved domains through different educational states, such as described in Chapter VI. For example, if a student is studying certain curriculum about biochemistry and that user is a second year undergraduate. Moreover, if that curriculum requires that the user should have certain knowledge level in the chemistry domain while he/she was in the first year. Therefore, the system that implements the model could handle such kind of prerequisites because the hybrid model is managing users' knowledge globally, i.e. with respect to semantic domains.

- b) It is designed to be used by frameworks that make use of multiple knowledge domains concurrently. Therefore, any change in a user's knowledge level with respect to any of the domains that an educational curriculum involves will be reflected to the other curricula that involve such domain(s). For example, by considering the student's example, if this student is studying a curriculum about programming, and he/she performed badly with respect to the concepts that belong to the object oriented programming. As a result, the other curricula that involve concepts about object oriented programming will present to the that particular students more examples or explanations about that particular domain which are suggested by the lesson's author(s) for students who belong to a knowledge level with respect to that domain less than that of the student.
- c) It has the capability of distinguishing between users with different educational levels and background that share the same stereotypical knowledge level. Therefore, the framework that implements that model could be used among different educational states simultaneously. For example, by following the student's example, if this student is in the second year and another one in the first year. Furthermore, both students share the same knowledge level in programming domain. Therefore, the framework that implements the model will be able to provide each of them with the appropriate kind of adaptation that suits their educational state, and this is because of the category stereotypes.
- d) The model gives the opportunity for the frameworks that implement it to share educational materials. However, the concepts that belong to these materials should be mapped to the same semantic domains. Therefore, these frameworks

should use the same knowledge classification methodology such as DDC (explained in Chapter VI).

9.3 The Hybrid Model – Implicit and Explicit

User models, such as those in systems described in Chapter IV, can be classified into either: a) implicit models; b) a mixture of explicit and implicit models. Explicit or cooperative user models (as described in Chapter II), are models that rely upon user cooperation to obtain information about him/her, such as his/her preferences, his/her knowledge, etc. On the other hand, implicit user models try to deduce these characteristics through monitoring users' actions during navigating the hyperspace (navigational behaviour). Of the eight systems described in Chapter IV, six of them implement a mixture of explicit and implicit methods to obtain information about their users that will provide the basis for adaptation. The analysis of these systems with respect to the user modelling methods is discussed in Chapter IV.

The way the Hybrid Model is implemented is a combination of implicit and explicit methodologies. Explicitly, when users access the system (WHURLE-HM) for the first time, they are asked to supply personal information, such as their first name and last name, as well as their student ID, which is used in the adaptation process. Moreover, users are asked to specify the category that they belong to, for example, first year undergraduate biology students, mathematics postgraduate students, etc. This categorization plays a crucial role in adaptation, as explained in Chapter VI. Furthermore, they explicitly specify their preferred interface layout, using any one of a number of pre-defined skins. After providing this information, the system implicitly starts to deduce users' knowledge levels about the specific semantic domains that are involved in the chosen topic through a pre-quiz, which is used to provide the basis for the adaptation. Furthermore, when users finish with that lesson they take a final quiz through which the system once more deduces their new knowledge levels with respect to the involved domains.

The reasons for mixing implicit and explicit methods in applying the Hybrid Model are:

- a) Some characteristics about users, such as their preferences, are difficult to infer [Brusilovsky 1996]. Because of that, it is chosen to make the system explicitly know users' preferred layout style and category.
- b) To allow a user to change some aspects in the user model, such as his/her knowledge level with respect to the domains involved in the chosen topic, may result in a wrong way of adaptation for that particular user. Brusilovsky [Brusilovsky 1996] argued that some information in the user model is very critical and any change in it could end up with faulty adaptation. Therefore, this information should be accessed only by an experienced user such as a teacher or system administrator, to set the model correctly. The only exception could be given to systems that have predefined assumptions about users' knowledge, such as Metadoc and Hypadapter, as it is not compulsory for their users to express their knowledge against the knowledge domain before accessing these systems. Therefore, the users may find some concepts they are familiar with, so they could tell the system about their familiarity with that particular concept(s). In AHA! [De Bra and Ruiter 2001] there is a kind of compromise regarding this issue, as authors may permit the knowledge about particular concepts to be altered by users, but not all concepts. However, in the case of WHURLE-HM where the Hybrid Model is implemented, there are no pre-defined assumptions about users' knowledge levels, as the knowledge levels of users are inferred from their answers in the pre-quiz and the post-quiz, which are produced by topics' authors. By this way, each time a user chooses a topic the system reassures the user's knowledge about the involved domains in that topic through these kinds of quizzes.

9.4 WHURLE-HM and Adaptation Techniques

The systems discussed in Chapter IV used different adaptation techniques. For example, AHA! used link annotation and link hiding for adaptive navigational support, and conditional text for adaptive presentation. In Hypadapter, hiding, sorting and annotation are used for adaptive navigation support and frame based technique for adaptive presentation, where frames are composed of slots that are sorted according to users' preferences and knowledge. WHURLE-HM, as described in Chapter VI, combined adaptive presentation and adaptive navigation support

together through using link removal. Lesson plans, which are explained in detail in Chapter VI, in the WHURLE system (and in turn WHURLE-HM) are composed of levels and these levels are composed of pages that are constituted of chunks. Therefore, the materials are presented in the form of links to levels that in turn present navigational buttons to the included pages. Also, each page may include one or more chunk. As explained in Chapter VI, in WHURLE-HM, each chunk has certain attributes that specify the domain that this chunk is serving and the required knowledge level with respect to that particular domain. Therefore, excluding or including any chunk is conditional, as it is based on the knowledge level. Therefore, if a page has no chunks to present, a link to that page will be removed. Similarly, if a level does not have any pages to include, the link to that level will be removed. Hence, adaptive presentation is provided through the conditional assembly of fragments of content. Additionally, some chunks could contain links, and by removing those chunks their associated links in turn will be removed. In adaptive navigational support techniques, that technique is called collateral structure adaptation, which is described in Chapter II.

De Bra [De Bra and Calvi 1998a], has reported that users of AHA! did not like the link removal technique, as they could not access what is removed. Brusilovsky and Pesin [Brusilovsky and Pesin 1998] have reported the use of link removal in educational context reduced the amount of navigation required to learn part of a course without reducing the quality of learning. However, they found that link annotation is preferred more than link removal. To overcome this debate a compromise solution is proposed. This solution offers the student the ability to revise the topic(s) they have finished in a non-adaptive and adaptive format, as explained in Chapter VI and Chapter VII. Therefore, by a non-adaptive format, users could use the material as a kind of reference, and thereby, they can build their own library where all information is available. By the adaptive format, they could retrieve the previous adaptive status of the finished topics; if they just want to swiftly retrieve certain information they have already studied without the need for extra explanations or introductions they already know or remember.

In addition to the adaptive links, WHURLE-HM supports non-adaptive links such as in AHA!. Those links, which are created by the lesson's author(s), are always found

on the menu bar and they refer to: a) other resources, which are external and internal, related to the topic under study; b) search engines such as Google.

9.5 Limitations

In Chapter VIII a user trial is described. Albeit the primary goal of the trial - which is the applicability of the Hybrid Model in educational setting and in adaptive educational hypermedia systems - is established, some limitations faced this trial. Those limitations can be summarised as follows:

- a) University regulations and time constraints: for the WHURLE-HM, which implements the Hybrid Model, to be tested with real students, it had to be applied through a real course. However, due to the university regulations the system is used as a revision guide. Moreover, the time taken to build the material of one lesson (revision guide) took about two months to finish, and two months for the trial, collecting data and analysing it.
- b) Environment: the environment at which the experiment was held was not controlled adequately. As explained in Chapter VIII, two main factors affected the results of the experiment: a) Technically – users experienced network congestion which affected the correlation between the number of times they accessed and the upgrade in their knowledge level, in addition to the server's hardware limitation; b) Sample size – the number of students who accessed the system (32 students) was not equal to those who did not (12 students). Another factor that may have affected the results and influenced the majority of students to access the system once or twice only is the fact that the usage of the system was optional. Therefore, some of students may have accessed the system just for curiosity and when they found technical problems, they did not try again. Thus, this factor is correlated in some way to the technical problem factor.

9.6 Further research

Many research areas related to the Hybrid Model deserve to be investigated. Those areas can be as follows:

- Evaluation: due to the nature of the Hybrid Model as being abstract and generic user model it is difficult to evaluate its potential through implementing it in one

system. Therefore, the researcher suggests a simple methodology for testing the model by applying it through a number of different systems using different implementation and adaptation techniques. This methodology could be designing different lessons (or courses) with common semantic domains among them, and those lessons are dependent on each other. Two groups with same number of subjects should be set up for that evaluation. One group would take the same courses without using the system, and the other group would use the system only. Moreover, all the technical issues such as network flow should be smooth and controlled. The time scale of the evaluation should be at least one month for every lesson, where students have to take a pre-quiz at the beginning of each lesson and a post-quiz after it. After that, a comparison between the changes in the knowledge level between users who accessed the system against those who did not should be performed. Furthermore, a correlation between the number of accesses, within the group who accessed the system, and the change in their knowledge level should be investigated. Moreover, a general exam that includes both groups should be done to see the impact of the system on students' performance. Through that suggested design, concrete evidence could be drawn from different involved systems regarding the success of the model. However, other aspects should be taken in account before carrying out this evaluation, as shown in the next two points.

- Lesson plan design: another area that is worth investigating is to find the best way to present information under the shadow of semantic domains. As explained in former chapters, the model is heavily dependent on authors' vision about the presented lessons and how they should be adapted. This area is very critical for the success of the Hybrid Model.
- Knowledge levels and learning styles: although many systems have classified their students into novice, beginner and advanced, further research is needed for that classification to find if it is enough or if more stereotypes are needed. Moreover, the Hybrid Model could include different learning styles with the knowledge levels. People who are involved in educational theories may be in a better position to find the validity of that widely used classification and to integrate proper learning styles.
- Semantic web: as explained in Chapter VI, the Hybrid Model uses the same approach of DDC (Dewey Decimal Classification) ontology to classify students'

knowledge. Furthermore, it is explained that the semantic web [Berners-Lee 1998, Fensel and Musen 2001] relies heavily on ontology. Therefore, the Hybrid Model could be used with semantic web to provide an adapted version of it. For example, if the framework, which implements the model, stated that a certain user is a beginner in a certain domain. Therefore, an agent can navigate the semantic web, and get all the documents that present that domain and are suitable for beginner users. This could happen by defining a standard ontology, such as DDC, for documents with an educational purpose. Therefore, the information pool of the Hybrid Model will not be dedicated to the available material within any system that implements it, but it will comprise the whole web. The big advantage in that integration is that users themselves can change the state of the documents, for example from intermediate to beginner, through their comments over the author(s) description for the document educational level – which is what the semantic web is offering. Therefore, users with different knowledge backgrounds will evaluate and change the presented material. Further research is needed to find the best way to integrate the Hybrid Model with the semantic web and to evaluate this new technology.

- Other fields: the Hybrid Model could be applied in different fields such as e-commerce. For example, if there is a company that sells mobile phones, instead of presenting information about accessories, ring tones, logos, etc. at the same time, the company web site could classify this information by using certain kind of ontology. Therefore, by some way or another, the site should identify the type of user that is accessing the site; for example if he/she is just interested in accessories, ring tones, models, etc. and based on that, a proper kind of information is presented. Hence, the overhead of information that is popular among most commercial web sites will be reduced and users could find what they want easily without jumping between different pages. In addition, if such kinds of sites share the same ontology they could define a kind of cooperation between them. Therefore, further research is needed to find how to use the advantages that the model is presenting to serve other different areas.

9.7 Conclusion

The question that motivated the researcher and upon which the research is based is:

What form of user model is needed to build an adaptive web-based educational framework, which has the ability to serve users with diverse goals and abilities; that also has the capability to define users' knowledge globally and not to be restricted to a particular topic or knowledge domain?

Answering this question has led to the development of the Hybrid Model that is described herein. This model is an attempt to open the gates for more advanced web-based educational applications with the use of techniques that are well examined in previous researches and modifying them for its purpose.

The Hybrid Model is a user model that combines the most commonly used techniques of user-modelling for adaptive hypertext. This utilises the various benefits of each of these techniques in a novel manner - to provide a detailed understanding of the user's needs and requirements on several different levels. Furthermore, this model is a generic abstract model that is flexible, and applicable to distinct knowledge fields as described in the user trial.

In addition to the Hybrid Model, another contribution is presented in this research, which is the new structure design of the WHURLE system. This presented design provides the system with the capability of implementing different user models easily. Moreover, this infrastructure provides a clean separation between the layout and the adaptation processes. Therefore, any changes in the display engine will not affect the utilised user model, and the opposite is true.

Finally, through this research the researcher hopes that the Hybrid Model has contributed to a new vision with regards to user modelling in adaptive educational hypermedia systems. This is a vision whereby educational systems are not restricted to a single knowledge domain but can employ different domains concurrently.

Appendices

Appendix A: Source Code

A.1 Authentication system files

was.html:

```
<html>
<head><title>WHURLE Authentication System</title></head>
<body>
<h1 align="Center" style="color:Green font-size:20 pt"><u>
WAS</u></h1>
<br/><br/>
<form method="post" action="was_mid_les.xml">
<table>
<tr>
<td> User Name :</td>
<td><input type="text" name="usnam"/></td>
</tr>
<tr>
<td>
Password :</td><td><input type="password" name="uspas"/></td>
</tr>
</table>
<br>
<input type="hidden" name="cmb" value="no"/>
<center><input type="submit" value="Verify"/></center>
<hr/>
New Users Need To <a href="streg.xml">Register</a>
</form>
</body>
</html>
```

was_mid_les.xml:

```
<?xml version="1.0"?>
<?xml-stylesheet href="was-tst.xml" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
  language="java"
  xmlns:esql="http://apache.org/cocoon/SQL/v2"
  xmlns:xsp="http://www.apache.org/1999/XSP/Core"
  xmlns:session="http://www.apache.org/1999/XSP/Session"
  create-session="true">

<xsp:structure>
<xsp:include>wasmysql</xsp:include>
</xsp:structure>

<tst>
<xsp:logic>
String cmb=request.getParameter("cmb");
wasmysql chk=new wasmysql();
String usnam="";
String uspas="";
if(cmb.equals("no"))
{
usnam=request.getParameter("usnam");
uspas=request.getParameter("uspas");
session.putValue("usnam", usnam);
session.putValue("uspas", uspas);
}
if(cmb.equals("yes"))
```

```

{
usnam= (String)session.getValue("usnam");
uspas=(String)session.getValue("uspas");
}
Vector lnames=new Vector();
Vector lrec=new Vector();
Vector lopr=new Vector();
Vector lid=new Vector();

if(chkk.tst(usnam,uspas))
{
<xsp:content><pass typ="yes"/></xsp:content>
<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurle</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>

<esql:execute-query>
<esql:query>
select lesson_flow.lesson_name, lesson_flow.lesson_id, user.stid from lesson_flow, user where
user.username='<xsp:expr>usnam</xsp:expr>' and
user.password='<xsp:expr>uspas</xsp:expr>' and user.categ = lesson_flow.categ
</esql:query>
<esql:results>
<esql:row-results>
String ln = <esql:get-string column="lesson_name"/>;
String ld = <esql:get-string column="lesson_id"/>;
lnames.addElement(ln);
lid.addElement(ld);
String id = <esql:get-string column="stid"/>;
session.putValue("stid", id);
</esql:row-results>
</esql:results>
</esql:execute-query>

<!-- monitoring purposes -->
if(cmb.equals("no"))
{
Locale currentLocale= new Locale("en","GB");
Date today1 = new Date();
SimpleDateFormat formatter1 = new SimpleDateFormat("hh:mm:ss a");
String time_c =formatter1.format(today1);

Date today = new Date();
SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yyyy");
String accdate = formatter.format(today);

<esql:execute-query>
<esql:query>
insert into monitor
values('<xsp:expr>session.getValue("stid")</xsp:expr>','<xsp:expr>accdate</xsp:expr>','<xsp:expr>tim
e_c</xsp:expr>')
</esql:query>
</esql:execute-query>
}
<!-- monitoring purposes -->

<esql:execute-query>
<esql:query>
select lesson_flow.lesson_name from lesson_flow, lessons_record where
lessons_record.stid='<xsp:expr>session.getValue("stid")</xsp:expr>' and lessons_record.lessonid =
lesson_flow.lesson_id
</esql:query>
<esql:results>
<esql:row-results>
String lr = <esql:get-string column="lesson_name"/>;

```

```

lrec.addElement(lr);
</esql:row-results>
</esql:results>
</esql:execute-query>

<!-- end of monitoring -->

<esql:execute-query>
<esql:query>
select lesson_flow.lesson_name from lesson_flow, open_lessons where
open_lessons.stid=<xsp:expr>session.getValue("stid")</xsp:expr> and open_lessons.open_lesson =
lesson_flow.lesson_id
</esql:query>
<esql:results>
<esql:row-results>
String lpn = <esql:get-string column="lesson_name"/>;
lopn.addElement(lpn);
</esql:row-results>
</esql:results>
</esql:execute-query>
</esql:connection>

for (int i=0; i &lt; lnames.size(); i++)
{
String chkl = (String)lnames.elementAt(i);
if (lopn.contains(chkl))
{
lnames.removeElement(chkl);
lid.removeElementAt(i);
}
if (lrec.contains(chkl))
{lnames.removeElement(chkl);
lid.removeElementAt(i);
}
}

}

for (int i =0; i &lt; lnames.size(); i++)
{
<xsp:content><lessons><xsp:expr>lnames.elementAt(i)</xsp:expr></lessons></xsp:content>;
}

for (int i =0; i &lt; lrec.size(); i++)
{
<xsp:content><finished><xsp:expr>lrec.elementAt(i)</xsp:expr></finished></xsp:content>;
}

for (int i =0; i &lt; lopn.size(); i++)
{
<xsp:content><open><xsp:expr>lopn.elementAt(i)</xsp:expr></open></xsp:content>;
}

session.putValue("lrec", lrec);
session.putValue("lopn", lopn);
session.putValue("lid", lid);
session.putValue("lnames", lnames);

}

if(!chkk.tst(usnam,uspas))
{<xsp:content><pass typ="no"/>
<err>Sorry.... You Are Not Authorized To Access The System</err>
</xsp:content>
}

</xsp:logic>
</tst>

```


</xsp:page>

was-tst.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<body>
<center><h1><u>WHURLE Authentication System</u></h1></center>
<p/>
<p/>
<xsl:variable name="ushk">
<xsl:value-of select="child::pass/@typ"/>
</xsl:variable>
<xsl:if test=" $ushk = 'yes'">
<style type="text/css">
option.les{color:red}
option.fin{color:green}
option.opn{color:orange}
</style>
Choose a lesson to access:
<center>
<form method="post" action="was_les2.xml">
<select name="lname" size="4">
<xsl:for-each select="lessons">
<xsl:variable name="ls1">
<xsl:value-of select="."/>
</xsl:variable>
<option value="{ $ls1}" class="les"/>[NOT] <xsl:value-of select="."/>
</xsl:for-each>

<xsl:for-each select="open">
<xsl:variable name="ls2">
<xsl:value-of select="."/>
</xsl:variable>
<option value="{ $ls2}" class="opn"/>[OPN] <xsl:value-of select="."/>
</xsl:for-each>

<xsl:for-each select="finished">
<xsl:variable name="ls3">
<xsl:value-of select="."/>
</xsl:variable>
<option value="{ $ls3}" class="fin"/>[FIN] <xsl:value-of select="."/>
</xsl:for-each>

</select><br/><br/>
<input type="hidden" name="opnl" value="no"/>
<input type="submit" value="send"/>
</form>
</center>
<hr/>
<font color="red">
Note: <br/>
<li> Red colored lessons/[NOT]-Lesson not accessed yet</li><br/>
<li> Orange colored lessons/[OPN]-Lessons not finished yet</li><br/>
<li> Green colored lessons/[FIN]-Lessons already finished</li>
</font>
</xsl:if>

<xsl:if test=" $ushk = 'no'">
<xsl:for-each select="err">
<font color="red"><br/><br/><center><xsl:value-of select="."/></center></font>
</xsl:for-each>
</xsl:if>
```

```

</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

was_les2.xml

```

<?xml version="1.0"?>
<?xml-stylesheet href="was.xsl" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
  language="java"
  xmlns:esql="http://apache.org/cocoon/SQL/v2"
  xmlns:xsp="http://www.apache.org/1999/XSP/Core"
  xmlns:session="http://www.apache.org/1999/XSP/Session"
  create-session="true">

<tst>
<xsp:logic>
String lname=request.getParameter("lname");
session.putValue("lname",lname);
String chkq="1";
String chkpost="0";
String stid = (String) session.getValue("stid");
Vector lrec2 = new Vector();
Vector lopn2 = new Vector();
Vector lid = new Vector();
Vector lnames= new Vector();
lrec2 =(Vector)session.getValue("lrec");
lopn2 =(Vector) session.getValue("lopn");
lid =(Vector) session.getValue("lid");
lnames =(Vector) session.getValue("lnames");
String rec="0";
String opnl= request.getParameter("opnl");
<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurie</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>

<esql:execute-query>
<esql:query>
select DISTINCT lesson_req.pretest, lesson_req.posttest from lesson_req, lesson_flow where
lesson_flow.lesson_name =
'<xsp:expr>lname</xsp:expr>' and lesson_flow.lesson_id = lesson_req.lessonid
</esql:query>
<esql:results>
<esql:row-results>
chkq = <esql:get-string column="pretest"/>;
chkpost = <esql:get-string column="posttest"/>;
</esql:row-results>
</esql:results>
</esql:execute-query>

session.putValue("ptest", chkpost);

if (lrec2.contains(lname) || lopn2.contains(lname) || opnl.equals("yes"))
{
if (lopn2.contains(lname) || opnl.equals("yes"))
{ rec ="1";
  session.putValue("rec", rec);
<xsp:content><direct tst="no"/></xsp:content>
<esql:execute-query>
<esql:query>

```

```

select DISTINCT lesson_req.domainid, stereotype.class from lesson_flow, lesson_req, stereotype
where
lesson_flow.lesson_name='<xsp:expr>lname</xsp:expr>' and
stereotype.stid='<xsp:expr>session.getValue("stid")</xsp:expr>' and
stereotype.domain_id=lesson_req.domainid
</esql:query>
<esql:results>
<esql:row-results>
String dom = <esql:get-string column="domainid"/>;
String stype = <esql:get-string column="class"/>;
session.putValue(dom,stype);
</esql:row-results></esql:results>
</esql:execute-query>
}

else if (lrec2.contains(lname))
{ session.putValue("rec", rec);
<xsp:content><direct tst="no"/></xsp:content>
<esql:execute-query>
<esql:query>
select old_lvl.dom_id, old_lvl.dom_lvl from old_lvl, lesson_flow where
lesson_flow.lesson_name='<xsp:expr>lname</xsp:expr>' and
old_lvl.stid='<xsp:expr>session.getValue("stid")</xsp:expr>' and
old_lvl.lesson_id=lesson_flow.lesson_id
</esql:query>
<esql:results>
<esql:row-results>

String dom = <esql:get-string column="dom_id"/>;
String stype = <esql:get-string column="dom_lvl"/>;
session.putValue(dom,stype);
</esql:row-results></esql:results>
</esql:execute-query>
}

<info>
  <esql:execute-query>
  <esql:query>
  select configfile from user where stid='<xsp:expr>session.getValue("stid")</xsp:expr>'
  </esql:query>
  <esql:results><esql:row-results>
  <conf><xsp:content><xsp:expr><esql:get-string
column="configfile"/></xsp:expr></xsp:content></conf>
  <lesname><xsp:content><xsp:expr>lname</xsp:expr></xsp:content></lesname>
  String conf=<esql:get-string column="configfile"/>;
  session.putValue("conf",conf);
  <xsp:content><error>lname second<xsp:expr>lname</xsp:expr></error></xsp:content>

  </esql:row-results>
  </esql:results>
  </esql:execute-query>

</info>
}

else{
  rec = "1";
  session.putValue("rec", rec);
  int elid = lnames.indexOf(lname);
  String eid = lid.elementAt(elid).toString();
  Vector chkls = new Vector();
  Vector tofin = new Vector();
  int rlchk = 0;
  <esql:execute-query>
  <esql:query>

```

```

select lesson_flow.lesson_name from mand_lessons, lesson_flow where
mand_lessons.lesson_id='<xsp:expr>eid</xsp:expr>' and
mand_lessons.mand_lesson_id=lesson_flow.lesson_id
</esql:query>
<esql:results>
<esql:row-results>
String mandls = <esql:get-string column="lesson_name"/>;
chkls.addElement(mandls);
</esql:row-results>
</esql:results>
</esql:execute-query>
for (int i=0; i &lt;&lt; chkls.size(); i++)
{
String mandl2 = chkls.elementAt(i).toString();
if(!lrec2.contains(mandl2))
{ rchk=1;
tofin.addElement(mandl2);
}
}
}

if ( rchk == 1)
{
<xsp:content><error> You Should Finish The Following Lesson(s) First: </error></xsp:content>
for (int i=0; i &lt;&lt; tofin.size(); i++)
{
<xsp:content><tofin><xsp:expr>tofin.elementAt(i)</xsp:expr></tofin></xsp:content>
}
}
}

if (rchk == 0 && chkq.equals("0"))
{

int lind = lnames.indexOf(lname);
String lessid = lid.elementAt(lind).toString();

<esql:execute-query>
<esql:query>
insert into open_lessons
values('<xsp:expr>session.getValue("stid")</xsp:expr>', '<xsp:expr>lessid</xsp:expr>')
</esql:query>
</esql:execute-query>

<xsp:content><direct tst="no"/></xsp:content>
<esql:execute-query>
<esql:query>
select DISTINCT lesson_req.domainid, stereotype.class from lesson_flow, lesson_req, stereotype
where
lesson_flow.lesson_name='<xsp:expr>lname</xsp:expr>' and
stereotype.stid='<xsp:expr>session.getValue("stid")</xsp:expr>' and
stereotype.domain_id=lesson_req.domainid
</esql:query>
<esql:results>
<esql:row-results>
String dom = <esql:get-string column="domainid"/>;
String stype = <esql:get-string column="class"/>;
session.putValue(dom,stype);
</esql:row-results></esql:results>
</esql:execute-query>

<info>
<esql:execute-query>
<esql:query>
select configfile from user where stid='<xsp:expr>session.getValue("stid")</xsp:expr>'
</esql:query>
<esql:results><esql:row-results>

```

```

    <conf><xsp:content><xsp:expr><esql:get-string
column="confgfile"/></xsp:expr></xsp:content></conf>
<lesname><xsp:contenti><xsp:expr>lname</xsp:expr></xsp:content></lesname>
String conf=<esql:get-string column="confgfile"/>;
session.putValue("conf",conf);
<xsp:content><error>lname second<xsp:expr>lname</xsp:expr></error></xsp:content>
</esql:row-results>
</esql:results>
</esql:execute-query>

</info>

}

if ( rchk == 0 &&& !chkq.equals("0"))
{
    <xsp:content><direct tst="yes"/></xsp:content>
<esql:execute-query>
<esql:query>
select DISTINCT pretest from lesson_req, lesson_flow where
lesson_flow.lesson_name='<xsp:expr>request.getParameter("lname")</xsp:expr>' and
lesson_req.lessonid = lesson_flow.lesson_id
</esql:query>
<esql:results>
<esql:row-results>
String pretest = <esql:get-string column="pretest"/>;
<xsp:content><toqz><xsp:expr>pretest</xsp:expr></toqz></xsp:content>
</esql:row-results>
</esql:results>
</esql:execute-query>
}
}

</esql:connection>

</xsp:logic>
</tst>
</xsp:page>

```

was.xsl:

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<body>
<center><h1><u>WHURLE Authentication System</u></h1></center>
<p/>
<p/>
<xsl:for-each select="direct">
<xsl:variable name="chk">
<xsl:value-of select="@tst"/>
</xsl:variable>
<xsl:if test="chk = 'no'">
<xsl:apply-templates select="info"/>
</xsl:if>
<xsl:if test="chk = 'yes'">
<xsl:apply-templates select="toqz"/>
</xsl:if>
</xsl:for-each>

<xsl:apply-templates/>

```

```

</body>
</html>
</xsl:template>

<xsl:template match="toqz">
<xsl:variable name="chk2">
<xsl:value-of select="."/>
</xsl:variable>
<form name="yyy" method="post" action="{chk2}.xml">
</form>
<script language="javascript">
document.yyy.submit();
</script>
</xsl:template>

<xsl:template match="info">
<xsl:variable name="conf"><xsl:value-of select="conf"/></xsl:variable>
<xsl:variable name="lesname"><xsl:value-of select="lesname"/></xsl:variable>
<form name="xxx" method="post" action="{lesname}.xml">
<input type="hidden" value="{lesname}" name="file"/>
<input type="hidden" value="whurle" name="sheet"/>
<input type="hidden" value="student-links" name="links"/>
<input type="hidden" value="{conf}" name="conf"/>
<input type="hidden" value="on" name="adap"/>
</form>
<script language="javascript">
document.xxx.submit();
</script>
</xsl:template>

<xsl:template match="error">
<font color="red">
<br/><br/><xsl:value-of select="."/><br/>
</font>
</xsl:template>

<xsl:template match="tofin">
<xsl:variable name="bk">
<xsl:value-of select="."/>
</xsl:variable>

<lj><a href="was_les2.xml?lname={bk}&amp;&amp;opnl=no"><xsl:value-of select="."/></a></lj><br/>
</xsl:template>
</xsl:stylesheet>

```

A.2 Adaptation filter

whurle-filter.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xinclude="http://www.w3.org/1999/XML/xinclude"
xmlns:xsp="http://www.apache.org/1999/XSP/Core">

<xsl:param name="adap"/>

<xsl:template match="lesson-plan">
<xsl:processing-instruction name="cocoon-process">type="xsp"</xsl:processing-instruction>
<xsl:processing-instruction name="cocoon-process">type="xinclude"</xsl:processing-instruction>
<xsl:processing-instruction name="cocoon-process">type="xslt"</xsl:processing-instruction>
<xsl:processing-instruction name="xml-stylesheet" href="whurle-wrapper.xsl"
type="text/xsl"</xsl:processing-instruction>

<xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:xinclude="http://www.w3.org/1999/XML/xinclude"
xmlns:request="http://www.apache.org/1999/XSP/Request"
create-session="true">

<wrapper>
<xsl:apply-templates/>
</wrapper>
</xsp:page>
</xsl:template>
```

```
<xsl:template match="versionlist">
<xsl:copy-of select="."/>
</xsl:template>
```

```
<xsl:template match="lesson">
<xsp:logic>
String chk = request.getParameter("adap");
</xsp:logic>
<xsl:copy>
<xsl:attribute name="title">
<xsl:value-of select="@title"/>
</xsl:attribute>
<xsl:attribute name="name">
<xsl:value-of select="@name"/>
</xsl:attribute>
<xsl:attribute name="xmlns:xinclude">
<xsl:value-of select="@xmlns:xinclude"/>
</xsl:attribute>
<xsl:apply-templates />
```

```
<xsp:logic>
if(chk.equals("on"))
{
String rec = (String) session.getValue("rec");
if (rec.equals("1"))
{
String lesschk = (String) session.getValue("ptest");
if (!lesschk.equals("0"))
{
<xsp:content>
<xsl:call-template name="advl"/>
</xsp:content>
}
}
}
```

```

}
</xsp:logic>
</xsl:copy>
</xsl:template>

<xsl:template match="level">
<xsl:copy>
<xsl:attribute name="name">
<xsl:value-of select="@name"/>
</xsl:attribute>
<xsl:attribute name="title">
<xsl:value-of select="@title"/>
</xsl:attribute>
<xsl:apply-templates />
</xsl:copy>
</xsl:template>

<xsl:template match="page"> <!-- cheking if the child chunks has something to display or not -->
<xsp:logic>
if(chk.equals("on")) <!-- if adaptation is on -->
{
if("<xsl:value-of select="."/chunk/@domain"/>".equals("general"))
{
<xsp:content>
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsp:content>
}
if ("<xsl:value-of select="."/chunk/@stereotype1"/>".equals(session.getValue("<xsl:value-of
select="."/chunk/@domain"/>")) || "<xsl:value-of
select="."/chunk/@stereotype2"/>".equals(session.getValue("<xsl:value-of
select="."/chunk/@domain"/>"))))
{
<xsp:content>
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsp:content>
}}
if(chk.equals("off")) <!-- if adaptation is off -->
{
<xsp:content>
<xsp:content>
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsp:content>
</xsp:content>
}
</xsp:logic>
</xsl:template>

<xsl:template match="chunk">
<xsp:logic>
if(chk.equals("on"))
{
if("<xsl:value-of select="@domain"/>".equals("general"))
{
<xsp:content>
<xsl:copy>
<xsl:apply-templates/>
</xsl:copy>
</xsp:content>
}
if ("<xsl:value-of select="@stereotype1"/>".equals(session.getValue("<xsl:value-of

```



```

select="@domain"/>")) || "<xsl:value-of select="@stereotype2"/>".equals(session.getValue("<xsl:value-
of
select="@domain"/>"))))
{
<xsp:content>
<xsl:copy>
<xsl:attribute name="domain">
<xsl:value-of select="@domain"/>
</xsl:attribute>
<xsl:apply-templates />
</xsl:copy>
</xsp:content>
}
}
if(chk.equals("off"))
{
<xsp:content>
<xsl:copy>
<xsl:attribute name="domain">
<xsl:value-of select="@domain"/>
</xsl:attribute>
<xsl:apply-templates />
</xsl:copy>
</xsp:content>
}
</xsp:logic>
</xsl:template>
<xsl:template match="linkbase">
<xsl:copy-of select="."/>
</xsl:template>
<xsl:template match="more">
<xsl:copy-of select="."/>
</xsl:template>
<xsl:template match="resources">
<xsl:copy-of select="."/>
</xsl:template>
<xsl:template match="issues">
<xsl:copy-of select="."/>
</xsl:template>

<xsl:template match="objectives">
<xsl:copy-of select="."/>
</xsl:template>

<xsl:template match="briefing">
<xsl:copy-of select="."/>
</xsl:template>

<xsl:template name="advl"> <!-- adding quiz level -->
<xsl:element name="level">
<xsl:attribute name="name">
<xsl:text>web-quiz</xsl:text>
</xsl:attribute>
<xsl:attribute name="title">
<xsl:text>Final Quiz</xsl:text>
</xsl:attribute>
<xsl:element name="page">
<xsl:element name="chunk">
<xsl:text>quiz</xsl:text>
</xsl:element>
</xsl:element>
</xsl:element>
</xsl:template>

</xsl:stylesheet>

```

A.3 History files

history.xml:

```
<?xml version="1.0"?>
<?xml-stylesheet href="history.xml" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
  language="java"
  xmlns:esql="http://apache.org/cocoon/SQL/v2"
  xmlns:xsp="http://www.apache.org/1999/XSP/Core"
  xmlns:session="http://www.apache.org/1999/XSP/Session"
  create-session="true">

<tst>
<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurle</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>
<esql:execute-query>
<esql:query>
select lesson_flow.lesson_name, lessons_record.score, lessons_record.finishdata from lesson_flow,
lessons_record where
lessons_record.stid='<xsp:expr>session.getValue("stid")</xsp:expr>' and
lessons_record.lessonid=lesson_flow.lesson_id
</esql:query>
<esql:results>
<esql:row-results>
<lname>
<name><esql:get-string column="lesson_name"/></name>
<conf><xsp:expr>session.getValue("conf")</xsp:expr></conf>
<score><esql:get-string column="score"/></score>
<date><esql:get-string column="finishdata"/></date>
</lname>
</esql:row-results>
</esql:results>
</esql:execute-query>
</esql:connection>
</tst>
</xsp:page>
```

history.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:param name="sheet">whurle</xsl:param>
<xsl:param name="links">student-links</xsl:param>
<xsl:param name="adap">off</xsl:param>

<xsl:template match="/tst">
<html>
<head>
<title>HISTORY</title>
</head>
<body>
<center>
<table border="1" cellpadding="3">
<tr><font color='red'><td>Lesson Name</td><td>Score</td><td>Finished Date</td></font></tr>
<xsl:for-each select="lname">
<tr>
<td>
```

```
<a>
<xsl:attribute name="href">
<xsl:value-of select="/name"/>.xml?sheet=<xsl:value-of select="$sheet"/>&amp;file=<xsl:value-of
select="/name"/>&amp;links=<xsl:value-of select="$links"/>&amp;conf=<xsl:value-of
select="/conf"/>&amp;adap=<xsl:value-of select="$adap"/>
</xsl:attribute>
<xsl:attribute name="target">_blank</xsl:attribute>
<xsl:value-of select="/name"/>
</a>
</td>
<td><xsl:value-of select="/score"/></td>
<td><xsl:value-of select="/date"/></td>
</tr>
</xsl:for-each>
</table>
</center>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

A.4 Quiz engine

quiz-results.xsl:

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="quiz">
    <html>
      <title>Quiz</title>
      <body>
        <form action="result.xml" method="post">
          <xsl:apply-templates />
          <br/><br/><center><input type="submit" value="Send"/></center>
        </form>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="questions">
    <xsl:for-each select="child::question[@type= 'choice']">
      <br/><li><xsl:value-of select="child::question-text"/></li>
      <br/>
      <xsl:call-template name="pics"/> <!-- for picuters if found -->
      <xsl:for-each select="child::answer">
        <xsl:variable name="chid">
          <xsl:value-of select="@id"/>
        </xsl:variable>
        <xsl:variable name="chval">
          <xsl:value-of select="@type"/>
        </xsl:variable>
        <input type="radio" name="{\$chid}" value="{\$chval}">
          <xsl:if test="parent::question[@media = 'image']">
            <xsl:variable name="sr">
              <xsl:value-of select="."/>
            </xsl:variable>
            
          </xsl:if>
          <xsl:if test="parent::question[@media = 'text']">
            <xsl:value-of select="."/>
          </xsl:if>
        </input><br/>
      </xsl:for-each>
    </xsl:for-each>
  </br/><center>-----</center><br/>

  <br/><br/><div style="color:green"><ul><center> Choose More Than One Answer
</center></ul></div><br/>

  <xsl:for-each select="child::question[@type= 'multiple']">
    <xsl:variable name="mqid">
      <xsl:value-of select="@id"/>
    </xsl:variable>
    <xsl:variable name="mqch">
      <xsl:value-of select="@choices"/>
    </xsl:variable>
    <xsl:variable name="mtyp">
      <xsl:value-of select="@type"/>
    </xsl:variable>
    <xsl:variable name="sid">
      <xsl:value-of select="concat(@id, 'm' )"/>
    </xsl:variable>
    <input type="hidden" name="{\$mqid}" value="{\$mtyp}"/>
    <input type="hidden" name="{\$sid}" value="{\$mqch}"/>
  <br/><li><xsl:value-of select="child::question-text"/></li>
```

```

<br/><xsl:call-template name="pics"/> <!-- for picuters if found -->
<xsl:for-each select="child::answer">
<xsl:variable name="ansid">
<xsl:value-of select="@id"/>
</xsl:variable>
<xsl:variable name="anstyp">
<xsl:value-of select="@id"/>
</xsl:variable>
<input type="checkbox" name="{\$ansid}" value="{\$anstyp}">
<xsl:if test=" parent::question[@media = 'image']">
<xsl:variable name="sr1">
<xsl:value-of select="."/>
</xsl:variable>

</xsl:if>
<xsl:if test=" parent::question[@media = 'text'] ">
<xsl:value-of select="."/>
</xsl:if>
</input><br/>
</xsl:for-each>
</xsl:for-each>
</xsl:template>

```

```

<xsl:template match="domains">
<xsl:variable name="dcnt">
<xsl:value-of select="@num"/>
</xsl:variable>
<input type="hidden" name="domcount" value="{\$dcnt}"/>
<xsl:for-each select="child::domain">
<xsl:variable name="domn">
<xsl:value-of select="concat('domain', @id )"/>
</xsl:variable>
<xsl:variable name="nm">
<xsl:value-of select="."/>
</xsl:variable>
<xsl:variable name="qids">
<xsl:value-of select="@qid"/>
</xsl:variable>
<input type="hidden" name="{\$domn}" value="{\$nm}"/> <!-- sending domain names as domain1 =
html(value) -->
<input type="hidden" name="{\$nm}" value="{\$qids}"/> <!-- sending question ids with domain name as
input name -->
</xsl:for-each>
</xsl:template>

```

```

<xsl:template match="mark-base">
<xsl:variable name="qnm">
<xsl:value-of select="."/>
</xsl:variable>
<input type="hidden" name="qstnum" value="{\$qnm}"/>
</xsl:template>

```

```

<xsl:template match="quiz-name">
<br/>
<div style="color:green ; border = thin blue solid; bgcolor:blue"><center>
<xsl:value-of select="."/>
</center></div>
</xsl:template>

```

```

<xsl:template match="type">
<xsl:variable name="typ">
<xsl:value-of select="."/>
</xsl:variable>
<input type="hidden" name="typ" value="{\$typ}"/>
</xsl:template>

```

```

<xsl:template name="pics">
<xsl:for-each select="child::pic">
<xsl:variable name="srs">
<xsl:value-of select="."/>
</xsl:variable>
<center><table border="0">
<tr>
<td>
<center></center>
</td>
</tr>
<tr>
<td>
<center><xsl:value-of select="@cap"/></center>
</td>
</tr>
</table>
</center>
</xsl:for-each>
</xsl:template>

</xsl:stylesheet>

```

A.4.1 Auto-marking

result.xml:

```

<?xml version="1.0"?>
<?xml-stylesheet href="result.xsl" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
  language="java"
  xmlns:xsp="http://www.apache.org/1999/XSP/Core"
  xmlns:session="http://www.apache.org/1999/XSP/Session"
  create-session="true">

<results>
<xsp:logic>
String dnumb=request.getParameter("domcount");
int dn = Integer.parseInt(dnumb);
String[] domain = new String[dn];
String[] qids= new String[dn];
String[][] domqs= new String[dn][50];
String type = request.getParameter("typ");
Vector chk = new Vector();
Vector finalqs = new Vector();
String ques = request.getParameter("qstnum");
String h;
String mqsts;
String qnchk;
String fnlchk;
String questchk;
String chktst;
String chktst2;
int mval = Integer.parseInt(ques);
int cntr = 0;
int x =0;
int k = 0;
int g =0;
int hh = 0;
int z = 0;
int u = 0;
boolean notok = false;

```

<!-- getting domain names and questions ids -->

```
<![CDATA[
for(int i=1; i <= dn; i++)
]]>
{
domain[i-1] = request.getParameter("domain" + i);
qids[i-1] = request.getParameter(domain[i-1]);
}
<!-- splitting questions ids into 2 dim and 1 dim arrays -->
```

```
int[] qpos = new int[dn];
for(int i=0; i < dn ; i++)
{
x=0;
StringTokenizer quest = new StringTokenizer(qids[i], ",");
qpos[i] = quest.countTokens();
x = x+1;
while(quest.hasMoreTokens())
{
chk.addElement(quest.nextToken());
x=x+1;
}
}
```

<!-- filtering 1 dim array for any questions' ids duplications -->

```
while (k < chk.size() )
{ questchk = chk.elementAt(k).toString();
if (!finalqs.contains(questchk))
{
finalqs.addElement(questchk);
k = k + 1;
}
else
{ k = k + 1;}
}
```

<!-- auto-marking question part-->

```
int[] qresults = new int[finalqs.size()];
for (int i=0; i < finalqs.size() ; i++)
{
try <!-- using try and catch to handle h if nothing is checked -->
{
h = request.getParameter(finalqs.elementAt(i).toString());
if (h != "multiple")
{ if (h.equals("yes"))
{
qresults[ctr]=mval;
}
if(h.equals("no"))
```

```

    {qresults[ctr]=0;}
}

if (h.equals("multiple"))
{
    mqsts = request.getParameter(finalqs.elementAt(i).toString() + ".0"); <!-- to prevent choosing all
options -->

    if(mqsts == null)
    {
        mqsts = request.getParameter(finalqs.elementAt(i).toString() + "m" );

        <xsp:content>-domains:<xsp:expr>mqsts</xsp:expr>-</xsp:content>

        StringTokenizer qanl = new StringTokenizer( mqsts , ",");
        while (qanl.hasMoreTokens())
        { qnchk = qanl.nextToken();
          fnlchk = request.getParameter(qnchk);
          String fnlchks ="ch";
          <xsp:content>#domain:<xsp:expr>qnchk</xsp:expr>#value:
<xsp:expr>fnlchk</xsp:expr></xsp:content>
          if (!fnlchk.equals(qnchk))
          { <xsp:content>-I'm empty-</xsp:content>
            notok = true;
            break;
          }
        }
        if(notok)
        {
            qresults[ctr]=0;
            notok = false;
        }
        else
        {qresults[ctr]=mval;}
    }
    else
    {qresults[ctr]=0;}
}
ctr = ctr + 1;
}
catch (NullPointerException e)
{
    e.printStackTrace(); <!-- if no data is sent put 0 in the qresults vector -->
    qresults[ctr]=0;
    ctr = ctr + 1;
}
}

```

<!-- adding scores with respect to the corresponding domains -->

```

x =0;
String[] score = new String[dn];

for (int i = 0; i <&lt; dn; i++)
{
    g = qpos[i];
    g = g + u;
    while (hh &lt; g)
    {
        chktst = chk.elementAt(hh).toString();

        while( z &lt; finalqs.size())
        {
            chktst2 = finalqs.elementAt(z).toString();

```



```

        if (chkst2.equals(chkst))
        {
            x = x + qresults[z];
        }
        z = z + 1;
    }
    hh = hh + 1;
    z = 0;
}
u = g;
score[i] = Integer.toString(x);
x = 0;
}
        <!-- Report -->

```

```

for (int i = 0; i &lt; dn; i++)
{
    <xsp:content><dom><xsp:expr>domain[i]</xsp:expr></dom></xsp:content>
    <xsp:content><score><xsp:expr>score[i]</xsp:expr></score></xsp:content>
}

```

<!-- getting overall score -->

```

x=0;
for (int i = 0; i &lt; finalqs.size(); i++)
{ x = x + qresults[i];}
session.putValue("score", Integer.toString(x));
<xsp:content>
    <total-score><xsp:expr>x</xsp:expr></total-score>
</xsp:content>
String dmns = Integer.toString(dn);
session.putValue("ndom", Integer.toString(dn));
<!-- status = request.getParameter("status");
session.putValue("status",status); -->
session.putValue("qinvdom", domain);
session.putValue("qinvdomrs", score);
session.putValue("type", type);
</xsp:logic>

</results>

</xsp:page>

```

result.xsl:

```

<?xml version="1.0"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/results">
<html>
<body>
<form name="zzz" method="post" action="uengine4.xml">
<xsl:for-each select="error">
<div style="color:red ; font: 15pt; font-weight:bold; border: thin solid blue">
<center>
<xsl:value-of select="."/>
</center>
</div>
<br/><br/><br/><br/>
</xsl:for-each>
<center>
<table background="green">
<tr bgcolor="gree">

```

```

<xsl:for-each select="dom">
<td>
<center><xsl:value-of select="."/></center>
</td>
</xsl:for-each>
</tr>
<tr bgcolor="yellow">
<xsl:for-each select="score">
<td><center><xsl:value-of select="."/></center></td>
</xsl:for-each>
</tr>
</table>
</center>
<br/><br/><br/>
ToTal Score : <xsl:value-of select="total-score"/>
</form>
<script language="javascript">
document.zzz.submit();
</script>
</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

A.4.2 Upgrade engine

uengine4.xml:

```

<?xml version="1.0"?>
<?xml-stylesheet href="uengine.xml" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
language="java"
xmlns:esql="http://apache.org/cocoon/SQL/v2"
xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:session="http://www.apache.org/1999/XSP/Session"
create-session="true">

<tst>
<xsp:logic>
String[] complvl=new String[3];
complvl[0]="beg";
complvl[1]="int";
complvl[2]="adv";
int[] comp=new int[3];
comp[0]=1;
comp[1]=2;
comp[2]=3;
int nvl=0;
int nvl2=0;
String contdom = (String)session.getValue("ndom");
int cont = Integer.parseInt(contdom);
String type = (String)session.getValue("type");
String[] stype=new String[cont];
int[] scr=new int[cont];
String[] nlevel=new String[cont];
String[] domid=new String[cont];
String[] qinv = new String[cont];
String[] qinvr = new String[cont];
qinv= (String[])session.getValue("qinvdom");
qinvr= (String[])session.getValue("qinvdomrs");
String lid="0";

```

```

<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurle</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>

<![CDATA[
for(int i=0; i < cont; i++)
]]>
{
<esql:execute-query>
<esql:query>
select stereotype.class from stereotype, domain where
stereotype.stid='<xsp:expr>session.getValue("stid")</xsp:expr>' and
domain.domain_name='<xsp:expr>qinvr[i]</xsp:expr>' and
domain.domain_id = stereotype.domain_id
</esql:query>
<esql:results>
<esql:row-results>
try{
stype[i] = <esql:get-string column="class"/>;
}
catch(NullPointerException e)
{
e.printStackTrace();
}
</esql:row-results></esql:results>
</esql:execute-query>
}

<!-- building arrays from the returning domain array and their number (domains) -->
<![CDATA[
for (int i=0; i < cont; i++)
]]>
{
String ff = qinvr[i];
scr[i] = Integer.parseInt(ff);
}

<![CDATA[
for(int i=0; i < cont; i++)
]]>
{
if (scr[i] == 0)
{
scr[i]=1;}
}

<esql:execute-query>
<esql:query>
select stereotype_scale.stereotype from knowledge_scale, stereotype_scale where
knowledge_scale.lrange
&lt; '<xsp:expr>scr[i]</xsp:expr>' and knowledge_scale.urange &gt;=
'<xsp:expr>scr[i]</xsp:expr>' and knowledge_scale.knowledge &gt;= stereotype_scale.lrange and
knowledge_scale.knowledge &lt;= stereotype_scale.urange
</esql:query>
<esql:results>
<esql:row-results>

nlevel[i]= <esql:get-string column="stereotype"/>; <!-- getting corresponding s-type for the present
scores -->
</esql:row-results></esql:results>
</esql:execute-query>
}
<!-- getting domains ID... -->
<![CDATA[

```

```

for(int i=0; i < cont; i++)
]]>
{
<esql:execute-query>
<esql:query>
select domain_id from domain where domain_name='<xsp:expr>qinv[i]</xsp:expr>'
</esql:query>
<esql:results>
<esql:row-results>
domid[i]= <esql:get-string column="domain_id"/>;
</esql:row-results></esql:results>
</esql:execute-query>
}

```

<!-- Comparison between old and new levels, then updating for the changed levels in involved domains takes place -->

```

for(int i=0; i &lt;&lt; cont; i++)
{

try {
for(int z=0; z &lt;&lt; 3; z++)
{
if (stype[i].equals(complvl[z]))
{
nlvl = comp[z];
}
if (nlevel[i].equals(complvl[z]))
{
nlvl2 = comp[z];
}
}

if ( nlvl2 &gt; nlvl || nlvl2 &lt; nlvl)

{

<!-- upgrading for user's stereotype takes place -->

<esql:execute-query>
<esql:query>
update stereotype set class='<xsp:expr>nlevel[i]</xsp:expr>' where
stdid='<xsp:expr>session.getValue("stdid") </xsp:expr>' and domain_id =
'<xsp:expr>domid[i]</xsp:expr>'
</esql:query>
<esql:results>
<esql:row-results/>
</esql:results>
</esql:execute-query>
}
}
catch (NullPointerException r)
{
r.printStackTrace();
<esql:execute-query>
<esql:query>
insert into stereotype
values('<xsp:expr>session.getValue("stdid")</xsp:expr>', '<xsp:expr>domid[i]</xsp:expr>', '<xsp:expr>nlevel[i]</xsp:expr>')
</esql:query>
</esql:execute-query>
}
}

```

<!-- Getting lesson ID -->

```

<esql:execute-query>
<esql:query>
select lesson_id from lesson_flow where lesson_name =
'<xsp:expr>session.getValue("lname")</xsp:expr>'
</esql:query>
<esql:results>
<esql:row-results>
lid = <esql:get-string column="lesson_id"/>;
</esql:row-results>
</esql:results>
</esql:execute-query>

for (int i=0; i &lt; cont; i++)
{
  <fresult>
  <domain><xsp:content><xsp:expr>qinv[i]</xsp:expr></xsp:content></domain>
  <score><xsp:content><xsp:expr>scr[i]</xsp:expr></xsp:content></score>
  <olevel><xsp:content><xsp:expr>styp[i]</xsp:expr></xsp:content></olevel>
  <nlevel><xsp:content><xsp:expr>nlevel[i]</xsp:expr></xsp:content></nlevel>
  </fresult>
}

if(type.equals("pretest"))
{
<esql:execute-query>
<esql:query>
insert into open_lessons
values('<xsp:expr>session.getValue("stid")</xsp:expr>', '<xsp:expr>lid</xsp:expr>')
</esql:query>
</esql:execute-query>
<xsp:content><type
typ="pretest"><xsp:expr>session.getValue("lname")</xsp:expr></type></xsp:content>
}
  <!-- if it is a post-test update lessons_record, open_lessons, old_level -->

if (type.equals("postest"))
{
  Date today = new Date();
  SimpleDateFormat formatter = new SimpleDateFormat("dd-MMM-yyyy");
  String datetoday = formatter.format(today);

  <esql:execute-query>
  <esql:query>
  insert into lessons_record
  values('<xsp:expr>session.getValue("stid")</xsp:expr>', '<xsp:expr>lid</xsp:expr>', '<xsp:expr>session.
  getValue("score")</xsp:expr>', '<xsp:expr>datetoday</xsp:expr>')
  </esql:query>
  </esql:execute-query>

  <esql:execute-query>
  <esql:query>
  delete from open_lessons where stid = '<xsp:expr>session.getValue("stid")</xsp:expr>' and
  open_lesson = '<xsp:expr>lid</xsp:expr>'
  </esql:query>
  </esql:execute-query>

  for (int i =0; i &lt; cont; i++)
  {
    <esql:execute-query>
    <esql:query>
    insert into old_lvl
    values('<xsp:expr>session.getValue("stid")</xsp:expr>', '<xsp:expr>lid</xsp:expr>', '<xsp:expr>domid[i]<
    /xsp:expr>', '<xsp:expr>styp[i]</xsp:expr>')
    </esql:query>
    </esql:execute-query>
  }
}

```

```

<xsp:content><type typ="postest"/></xsp:content>
}
</esql:connection>
</xsp:logic>
</tst>

</xsp:page>

```

uengine.xsl:

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<head><title>Results Report</title></head>
<body>
<center>
<font size="+2" color="red"><u>Report</u></font>
</center>
<br/>
<center>
<table border="1">
<tr>
<font color="green">
<td>Domain name</td><td>Score</td><td>Old Level</td><td>New
Level</td></font></tr>
<xsl:for-each select="fresult">
<tr>
<td><xsl:value-of select="./domain"/></td>
<td><xsl:value-of select="./score"/></td>
<td><xsl:value-of select="./olevel"/></td>
<td><xsl:value-of select="./nlevel"/></td>
</tr>
</xsl:for-each>
</table>
</center>
<xsl:for-each select="type">
<xsl:if test="@typ = 'pretest'">
<xsl:variable name="ln">
<xsl:value-of select="."/>
</xsl:variable>
If you want to access the lesson now press on the lesson name:
<a href="was_les2.xml?lname={\$ln}&&opnl=yes"><xsl:value-of select="\$ln"/></a>
</xsl:if>

<br/>If you want to return to lessons list
<a href="was_mid_les.xml?cmb=yes"> Press Here</a>

</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

A.5 Administrative tools

A.5.1 Students' registration tool

streg.xml:

```
<?xml version="1.0"?>
<?xml-stylesheet href="streg.xsl" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:esql="http://apache.org/cocoon/SQL/v2"
xmlns:session="http://www.apache.org/1999/XSP/Session"
create-session="true">

<tst>
<xsp:logic>
String[] catg=new String[10];
String[] catid=new String[10];
String[] confn=new String[10];
String[] conff=new String[10];
int i=0;
int x=0;

<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurle</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>
<esql:execute-query>
<esql:query>
select category_name, category_id from category
</esql:query>
<esql:results>
<esql:row-results>
<cate><xsp:content><xsp:expr><esql:get-string
column="category_name"/></xsp:expr></xsp:content></cate>
catg[i] = <esql:get-string column="category_name"/>;
catid[i] = <esql:get-string column="category_id"/>;
i = i + 1;
</esql:row-results>
</esql:results>
</esql:execute-query>

session.putValue("allcat",catg);
session.putValue("allcatid",catid);

<esql:execute-query>
<esql:query>
select conf_name, conf_file from conf
</esql:query>
<esql:results>
<esql:row-results>
<confn><xsp:content><xsp:expr><esql:get-string
column="conf_name"/></xsp:expr></xsp:content></confn>
confn[x] = <esql:get-string column="conf_name"/>;
conff[x] = <esql:get-string column="conf_file"/>;
x = x + 1;
</esql:row-results>
</esql:results>
</esql:execute-query>
session.putValue("allconfn",confn);
session.putValue("allconff",conff);
</esql:connection>
```

```

</xsp:logic>
</tst>

</xsp:page>

```

streg.xml:

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<head>
<title>Registration</title>

<script>
function verify(passform)
{
if(passform.stid.value==" " || passform.fname.value==" " || passform.lname.value==" " ||
passform.usname.value==" " || passform.password.value==" ")
{alert("Some fields are missing");
return false;}

if (passform.password.value != passform.password2.value)
{alert("Entered password did not match");
return false;}
}
</script>

</head>
<body>
<center>
<h1 style="color:green">
<u>Registration Form</u></h1></center>

<form onsubmit="return verify(this)" method="post" action="studreg.xml" align="center">
<p>Student ID<br/><input type="text" name="stid"/>
<p>First name<br/><input type="text" name="fname"/>
<p>Last name<br/><input type="text" name="lname"/>
<p>User name<br/><input type="text" name="usname"/>
<p>Password<br/><input type="password" name="password"/>
<p>Confirm Password<br/><input type="password" name="password2"/>
<p/>
Select a Category:
<select name="cat">
<xsl:for-each select="cate">
<xsl:variable name="catgo">
<xsl:value-of select="."/>
</xsl:variable>
<option value="{ $catgo }"><xsl:value-of select="."/></option>
</xsl:for-each>
</select>
<p/>
Select Style:
<select name="conf">
<xsl:for-each select="confn">
<xsl:variable name="con">
<xsl:value-of select="."/>
</xsl:variable>
<option value="{ $con }"><xsl:value-of select="."/></option>
</xsl:for-each>
</select>
<p/>
<center><input type="submit" value="Register" /></center>
</form>

```



```

</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

studreg.xml:

```

<?xml version="1.0"?>
<?xml-stylesheet href="studreg.xsl" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>
<xsp:page language="java" xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:esql="http://apache.org/cocoon/SQL/v2"
xmlns:session="http://www.apache.org/1999/XSP/Session" create-session="true">

<xsp:structure>
<xsp:include>wasmysql</xsp:include>
<xsp:include>streg</xsp:include>
</xsp:structure>

<tst>
<xsp:logic>
wasmysql verf=new wasmysql();<!-- test -->
streg inst=new streg(); <!-- insdata -->
String stid=request.getParameter("stid");
String fname=request.getParameter("fname");
String lname=request.getParameter("lname");
String username=request.getParameter("username");
String password=request.getParameter("password");
String cat=request.getParameter("cat");
String conf=request.getParameter("conf");
String llesson="0"; <!-- next lesson Id -->
String catid="";
String confile="";
String pass1= password.toLowerCase();
String username1=username.toLowerCase();
String[] catg=new String[10];
String[] catgid=new String[10];
String[] confn=new String[10];
String[] conff=new String[10];
catg = (String[])session.getValue("allcat");
catid = (String[])session.getValue("allcatid");
confn = (String[])session.getValue("allconfn");
conff = (String[])session.getValue("allconff");

if(verf.tst(username1, pass1))
{<xsp:content>
<out>your username is used, please press back and re-enter another one</out>
</xsp:content>
}
if(!verf.tst(username1, pass1))
{
for (int i=0; i &lt; 10; i++)
{
if (catg[i].equals(cat))
{
catid = catgid[i];
break;
}
}
}

for (int x=0; x &lt; 10; x++)
{
if( confn[x].equals(confn))
{

```

```

        confile = conff[x];
        break;
    }
}
inst.insdata(fname.toLowerCase(),lname.toLowerCase(),username.toLowerCase(),password.toLowerCase(),stid,catid,confile);
<xsp:content><out>registered</out><goto/></xsp:content>
}
</xsp:logic>
</tst>
</xsp:page>

```

studreg.xsl

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<body>
<xsl:for-each select="out">
<font color="red">
<center>
<xsl:value-of select="."/>
</center>
</font>
</xsl:for-each>
<xsl:for-each select="goto">
<br/>
<br/>
<br/>
You Need To Login To The System <a href="was.html">Press Here</a> To Go To The Login Screen.
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

A.5.2 Lessons' registration tool

ladmin.xml:

```

<?xml version="1.0"?>
<?xml-stylesheet href="ladmin.xsl" type="text/xsl"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
language="java"
xmlns:esql="http://apache.org/cocoon/SQL/v2"
xmlns:xsp="http://www.apache.org/1999/XSP/Core"
xmlns:session="http://www.apache.org/1999/XSP/Session"
create-session="true">
<xsp:structure>
<xsp:include>ldom</xsp:include>
<xsp:include>domcount</xsp:include>
</xsp:structure>

<tst>
<xsp:logic>
String[] domain=new String[10];
String[] domainid=new String[10];
int i=0;

```

```

<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurle</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>
<esql:execute-query>
<esql:query>
select domain_name, domain_id from domain
</esql:query>
<esql:results>
<esql:row-results>
<lname><xsp:content><xsp:expr><esql:get-string
column="domain_name"/></xsp:expr></xsp:content></lname>
domain[i]=<esql:get-string column="domain_name"/>;
domainid[i]=<esql:get-string column="domain_id"/>;
i=i+1;
</esql:row-results>
</esql:results>
</esql:execute-query>

session.putValue("alldom",domain);
session.putValue("alldomid",domainid);
<esql:execute-query>
<esql:query>
select count(*) from domain
</esql:query>
<esql:results>
<esql:row-results>
<count><xsp:content><xsp:expr><esql:get-string
column="count(*)"/></xsp:expr></xsp:content></count>
</esql:row-results>
</esql:results>
</esql:execute-query>

<esql:execute-query>
<esql:query>
select category_name from category
</esql:query>
<esql:results>
<esql:row-results>
<cat><xsp:content><xsp:expr><esql:get-string
column="category_name"/></xsp:expr></xsp:content></cat>
</esql:row-results>
</esql:results>
</esql:execute-query>

</esql:connection>
</xsp:logic>
</tst>

</xsp:page>

```

ladmin.xsl:

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<body>
<form method="post" action="ladmin2.xml">
Lesson Name: <input type="text" name="lname"/><p/>
Choose involved Domains :<br/>
<xsl:for-each select="lname">
<xsl:variable name="dom">

```

```

<xsl:value-of select="."/>
</xsl:variable>
<table border="0">
<td width="100">
<input type="checkbox" name="{\$dom}"/><xsl:value-of select="."/>
</td>
</table>
</xsl:for-each>
<xsl:for-each select="count">
<xsl:variable name="cnt">
<xsl:value-of select="."/>
</xsl:variable>
<input type="hidden" value="{\$cnt}" name="cnt"/>
</xsl:for-each>
<p/>
Select a Category:
<select name="cats">
<xsl:for-each select="cat">
<xsl:variable name="cate">
<xsl:value-of select="."/>
</xsl:variable>
<option value="{\$cate}"><xsl:value-of select="."/></option>
</xsl:for-each>
</select>

<p/>
Name Of The Pre-quiz <input type="text" name="pre"/> <font color="red">[Write 0 for None]</font>
<br/>
Name Of The Post-quiz <input type="text" name="pos"/> <font color="red">[Each Lesson Should have
a
Post-quiz]</font>
<p/>
<input type="submit"/>
</form>
</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

ladmin2.xml:

```

<?xml version="1.0"?>
<?xml-stylesheet href="ladmin2.xml" type="text/xml"?>
<?cocoon-process type="xsp"?>
<?cocoon-process type="xslt"?>

<xsp:page
  language="java"
  xmlns:esql="http://apache.org/cocoon/SQL/v2"
  xmlns:xsp="http://www.apache.org/1999/XSP/Core"
  xmlns:session="http://www.apache.org/1999/XSP/Session"
  create-session="true">
<xsp:structure>
<xsp:include>lesver</xsp:include>
</xsp:structure>

<tst>
<xsp:logic>
lesver ver=new lesver();
String lname=request.getParameter("lname");
if(!ver.tst(lname))
{
String cat=request.getParameter("cats");
String catid="";
String count= request.getParameter("cnt");

```

```

int cont=Integer.parseInt(count);
String[] domain=new String[10];
String[] domainid=new String[10];
domain=(String[])session.getValue("alldom");
domainid=(String[])session.getValue("alldomid");
int lid=0;
String lid1="";
String Dat="0000-00-0";
String nleson="0";

```

```

<esql:connection>
<esql:driver>org.gjt.mm.mysql.Driver</esql:driver>
<esql:dburl>jdbc:mysql://localhost/whurle</esql:dburl>
<esql:username>mrz</esql:username>
<esql:password>zocka</esql:password>
<esql:execute-query>
<esql:query>
select MAX(lesson_id) as id from lesson_flow
</esql:query>
<esql:results>
<esql:row-results>
lid = Integer.parseInt(<esql:get-string column="id"/>);
</esql:row-results>
</esql:results>
</esql:execute-query>

```

```
lid = lid + 1;
```

```

<esql:execute-query>
<esql:query>
select category_id from category where category_name =
'<xsp:expr>request.getParameter("cats")</xsp:expr>'
</esql:query>
<esql:results>
<esql:row-results>
catid = <esql:get-string column="category_id"/>;
</esql:row-results>
</esql:results>
</esql:execute-query>

```

```

<esql:execute-query>
<esql:query>
insert into lesson_flow
values('<xsp:expr>lname</xsp:expr>','<xsp:expr>lid</xsp:expr>','<xsp:expr>catid</xsp:expr>','<xsp:ex
pr>nleson</xsp:expr>','<xsp:expr>Dat</xsp:expr>')
</esql:query>
<esql:results>
<esql:row-results/>
</esql:results>
</esql:execute-query>

```

```

for (int i=0; i &lt; cont ; i++)
{
if (request.getParameter(domain[i]) != null)
{
<dom><xsp:content><br/>Domain Name:<xsp:expr>domain[i]</xsp:expr>--Required
Level:<xsp:expr>request.getParameter(domain[i]+"1")</xsp:expr></xsp:content></dom>

```

```

<esql:execute-query>
<esql:query>
insert into lesson_req
values('<xsp:expr>lid</xsp:expr>','0','<xsp:expr>domainid[i]</xsp:expr>','<xsp:expr>request.getParame
ter("pre")</xsp:expr>','<xsp:expr>request.getParameter("pos")</xsp:expr>')
</esql:query>
<esql:results>
<esql:row-results/>
</esql:results>

```

```

</esql:execute-query>
}
}
</esql:connection>
}
else
{
<dom>Lesson's Name is used, Press Back and Choose another name. </dom>
}
</xsp:logic>
</tst>

</xsp:page>

```

ladmin2.xsl:

```

<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/tst">
<html>
<body>
<xsl:for-each select="count">
<xsl:value-of select="."/>
</xsl:for-each>
<xsl:for-each select="dom">
<xsl:value-of select="."/>
<br/>
</xsl:for-each>
</body>
</html>
</xsl:template>

</xsl:stylesheet>

```

Appendix B: Database Tables

Note 1: the following is automatically created by the *mysqldump* command of the MySQL server

Note 2: most of the data examples provided in the following tables is removed, as it may identify personal information. Therefore, the provided data is a very small subset of the real data stored by the database tables.

```
# MySQL dump 7.1
#
# Host: localhost      Database: whurle
#-----
# Server version  3.22.32

#
# Table structure for table 'category'
#
CREATE TABLE category (
  category_name varchar(50),
  category_id varchar(5)
);

#
# Dumping data for table 'category'
#

INSERT INTO category VALUES ('Ph.D biology','1');
INSERT INTO category VALUES ('1st year chemistry','2');
INSERT INTO category VALUES ('undergraduate student','3');
INSERT INTO category VALUES ('2nd year math','4');

#
# Table structure for table 'conf'
#
CREATE TABLE conf (
  conf_name varchar(50),
  conf_file varchar(50)
);

#
# Dumping data for table 'conf'
#

INSERT INTO conf VALUES ('Modern Style','intro-web-conf');
INSERT INTO conf VALUES ('Egyption Style','egypt-conf-file');
INSERT INTO conf VALUES ('Metalic Style','metalic-conf-file');

#
# Table structure for table 'domain'
#
CREATE TABLE domain (
  domain_name varchar(30),
  domain_id varchar(10)
);

#
# Dumping data for table 'domain'
#
```

```

INSERT INTO domain VALUES ('chemistry','10');
INSERT INTO domain VALUES ('math','20');
INSERT INTO domain VALUES ('physics','30');
INSERT INTO domain VALUES ('biology','40');
INSERT INTO domain VALUES ('fossiles','60');
INSERT INTO domain VALUES ('html','50');
INSERT INTO domain VALUES ('functions','70');

#
# Table structure for table 'knowledge_scale'
#
CREATE TABLE knowledge_scale (
  lrange int(3),
  urange int(3),
  knowledge int(3)
);

#
# Dumping data for table 'knowledge_scale'
#

INSERT INTO knowledge_scale VALUES (0,10,1);
INSERT INTO knowledge_scale VALUES (11,20,2);
INSERT INTO knowledge_scale VALUES (21,30,3);
INSERT INTO knowledge_scale VALUES (31,40,4);
INSERT INTO knowledge_scale VALUES (41,50,5);
INSERT INTO knowledge_scale VALUES (51,60,6);
INSERT INTO knowledge_scale VALUES (61,70,7);
INSERT INTO knowledge_scale VALUES (71,80,8);
INSERT INTO knowledge_scale VALUES (81,90,9);
INSERT INTO knowledge_scale VALUES (91,100,10);

#
# Table structure for table 'lesson_flow'
#
CREATE TABLE lesson_flow (
  lesson_name varchar(30),
  lesson_id int(11),
  categ varchar(20),
  nlessonid int(11),
  closingdate date
);

#
# Dumping data for table 'lesson_flow'
#

INSERT INTO lesson_flow VALUES ('test-from-home',13,'4',0,'0000-00-00');
INSERT INTO lesson_flow VALUES ('quiz-lesson',14,'2',0,'0000-00-00');
INSERT INTO lesson_flow VALUES ('intro-web-mr',15,'3',0,'0000-00-00');
INSERT INTO lesson_flow VALUES ('anthropology',16,'3',0,'0000-00-00');

#
# Table structure for table 'lesson_req'
#
CREATE TABLE lesson_req (
  lessonid varchar(10),

```



```

required_type varchar(10),
domainid varchar(10),
pretest varchar(20),
posttest varchar(20)
);

#
# Dumping data for table 'lesson_req'
#

INSERT INTO lesson_req VALUES ('13','int','30','0','0');
INSERT INTO lesson_req VALUES ('13','beg','50','0','0');
INSERT INTO lesson_req VALUES ('14','int','10','0','0');
INSERT INTO lesson_req VALUES ('14','beg','30','0','0');
INSERT INTO lesson_req VALUES ('15','0','30','pre','post');
INSERT INTO lesson_req VALUES ('15','0','50','pre','post');
INSERT INTO lesson_req VALUES ('16','0','60','Anthro-quiz-ft','quiz-
xincl');
INSERT INTO lesson_req VALUES ('16','0','70','Anthro-quiz-ft','quiz-
xincl');

#
# Table structure for table 'lessons_record'
#
CREATE TABLE lessons_record (
  stid varchar(20),
  lessonid int(11),
  score int(11),
  finishdata varchar(12)
);

#
# Dumping data for table 'lessons_record'
#

INSERT INTO lessons_record VALUES ('123456',1,10,'10-Sep-2002');
INSERT INTO lessons_record VALUES ('123456',3,30,'10-Sep-2002');
INSERT INTO lessons_record VALUES ('123456',5,0,'30-Jan-2003');
INSERT INTO lessons_record VALUES ('12345678',5,0,'30-Jan-2003');
INSERT INTO lessons_record VALUES ('123456789',5,80,'31-Jan-2003');

#
# Table structure for table 'mand_lessons'
#
CREATE TABLE mand_lessons (
  lesson_id varchar(5),
  mand_lesson_id varchar(5)
);

#
# Dumping data for table 'mand_lessons'
#

INSERT INTO mand_lessons VALUES ('8','5');

#
# Table structure for table 'monitor'
#
CREATE TABLE monitor (
  stid varchar(20),
  date varchar(20),

```

```

    time varchar(20)
);

#
# Dumping data for table 'monitor'
#

INSERT INTO monitor VALUES ('12345678','24-Mar-2003','11:51:30 AM');
INSERT INTO monitor VALUES ('123456','02-Apr-2003','10:53:00 AM');
INSERT INTO monitor VALUES ('123456','24-Mar-2003','11:13:15 AM');
INSERT INTO monitor VALUES ('12345678','20-Mar-2003','01:01:18 PM');
INSERT INTO monitor VALUES ('12345678','20-Mar-2003','07:13:32 PM');
INSERT INTO monitor VALUES ('123456','23-Mar-2003','02:32:02 PM');
INSERT INTO monitor VALUES ('123456','24-Mar-2003','12:10:13 PM');
INSERT INTO monitor VALUES ('654321','25-Mar-2003','02:35:23 PM');
INSERT INTO monitor VALUES ('654321','26-Mar-2003','11:42:57 AM');
INSERT INTO monitor VALUES ('654321','26-Mar-2003','01:29:22 PM');

#
# Table structure for table 'old_lvl'
#
CREATE TABLE old_lvl (
  stid varchar(20),
  lesson_id varchar(10),
  dom_id varchar(10),
  dom_lvl varchar(10)
);

#
# Dumping data for table 'old_lvl'
#

INSERT INTO old_lvl VALUES ('12345678','5','50','int');
INSERT INTO old_lvl VALUES ('12345678','5','30','beg');
INSERT INTO old_lvl VALUES ('12345678','5','40','int');
INSERT INTO old_lvl VALUES ('12345678','5','10','beg');
INSERT INTO old_lvl VALUES ('123456','5','50','int');
INSERT INTO old_lvl VALUES ('123456','5','30','int');
INSERT INTO old_lvl VALUES ('123456','5','40','int');
INSERT INTO old_lvl VALUES ('123456','5','10','int');
INSERT INTO old_lvl VALUES ('654321','5','10','beg');
INSERT INTO old_lvl VALUES ('654321','5','40','beg');
INSERT INTO old_lvl VALUES ('654321','5','30','beg');
INSERT INTO old_lvl VALUES ('654321','5','50','beg');

#
# Table structure for table 'open_lessons'
#
CREATE TABLE open_lessons (
  stid varchar(20),
  open_lesson varchar(20)
);

#
# Dumping data for table 'open_lessons'
#

INSERT INTO open_lessons VALUES ('123456','16');
INSERT INTO open_lessons VALUES ('12345678','5');
INSERT INTO open_lessons VALUES ('654321','5');
INSERT INTO open_lessons VALUES ('123456','1');

```

```

#
# Table structure for table 'stereotype'
#
CREATE TABLE stereotype (
  stid varchar(20),
  domain_id varchar(10),
  class varchar(10)
);

#
# Dumping data for table 'stereotype'
#

INSERT INTO stereotype VALUES ('123456','10','beg');
INSERT INTO stereotype VALUES ('123456','30','beg');
INSERT INTO stereotype VALUES ('654321','30','int');
INSERT INTO stereotype VALUES ('654321','40','beg');
INSERT INTO stereotype VALUES ('123456','40','beg');
INSERT INTO stereotype VALUES ('123456','50','beg');

#
# Table structure for table 'stereotype_scale'
#
CREATE TABLE stereotype_scale (
  lrange int(3),
  urange int(3),
  stereotype varchar(4)
);

#
# Dumping data for table 'stereotype_scale'
#

INSERT INTO stereotype_scale VALUES (1,3,'beg');
INSERT INTO stereotype_scale VALUES (4,8,'int');
INSERT INTO stereotype_scale VALUES (9,10,'adv');

#
# Table structure for table 'user'
#
CREATE TABLE user (
  fname varchar(20),
  lname varchar(20),
  username varchar(20),
  password varchar(10),
  stid varchar(20),
  categ varchar(20),
  configfile varchar(20)
);

#
# Dumping data for table 'user'
#

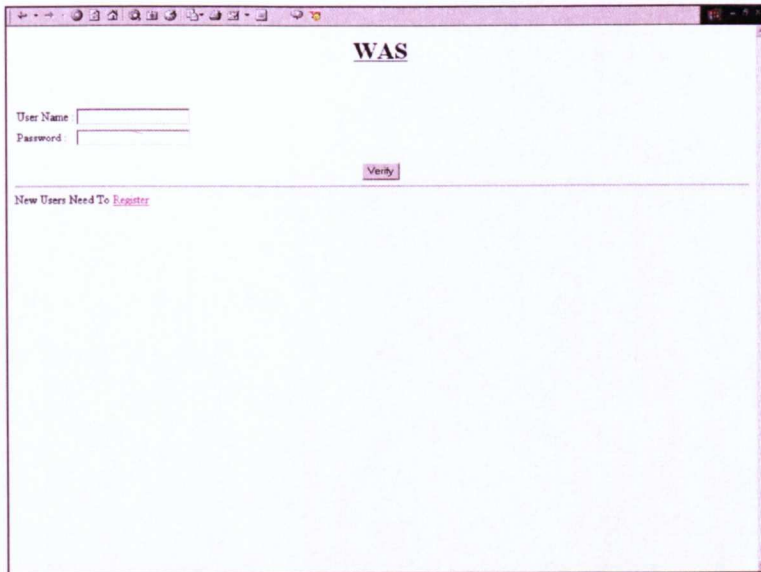
INSERT INTO user VALUES
('Mohamed','Ramzy','moh','ram','123456','3','intro-web-conf');
INSERT INTO user VALUES
('ppp','av','plpmc','prsedf','12345678','3','intro-web-conf');
INSERT INTO user VALUES ('aa','aaa','aa','aa','233456','3','intro-
web-conf');
INSERT INTO user VALUES ('fff','fff','nnn','nnn','333','3','intro-
web-conf');

```

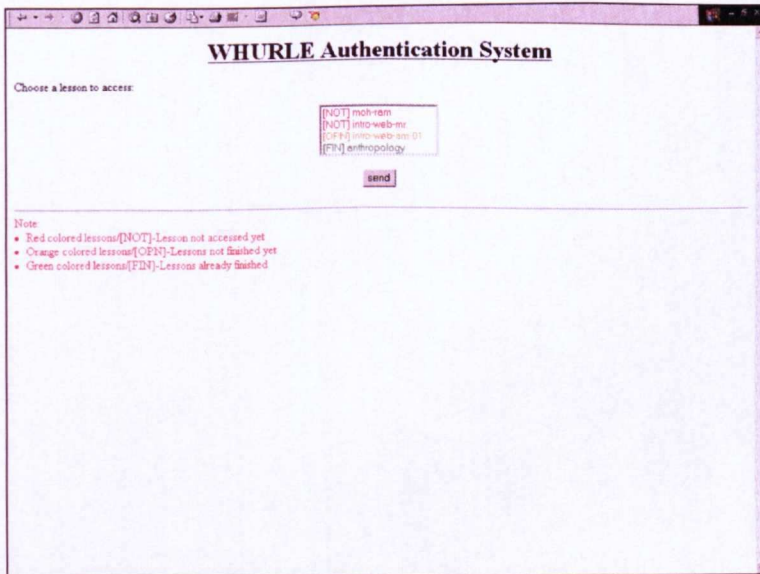
```
INSERT INTO user VALUES ('gg','ss','craig','nec','654321','3','intro-  
web-conf');
```

Appendix C: Snapshots

C.1 Authentication screens



Login screen



Choosing lessons screen

C.2 Quiz

C.2.1 Quiz Screen

Note 1: this is quiz had been used in the user trial experiment as a pre-quiz. However, the post-quiz was paper based and held the same questions as the pre-quiz.

Note 2: the three following images show the full quiz in three parts.

Anthropology Pre-Quiz

- Fossil bones are found only in igneous rocks:
 - True
 - False
 - I don't know
- Most fossils are made of stone, not organic matter:
 - True
 - False
 - I don't know
- Indirect 'traces' of ancient life such as footprints are fossils:
 - True
 - False
 - I don't know
- Mammoths frozen during the last ice age are not fossils:
 - True
 - False
 - I don't know
- Cranial endocasts are rightly called fossils:
 - True
 - False
 - I don't know
- Fossilisation tends to occur non-randomly around the landscape:
 - True
 - False
 - I don't know

- False
- I don't know
- The cranium in Fig 2 is platycephalic:




Fig 2

 - True
 - False
 - I don't know
- The mandible in Fig 3 illustrates the phenomenon of 'molarization':


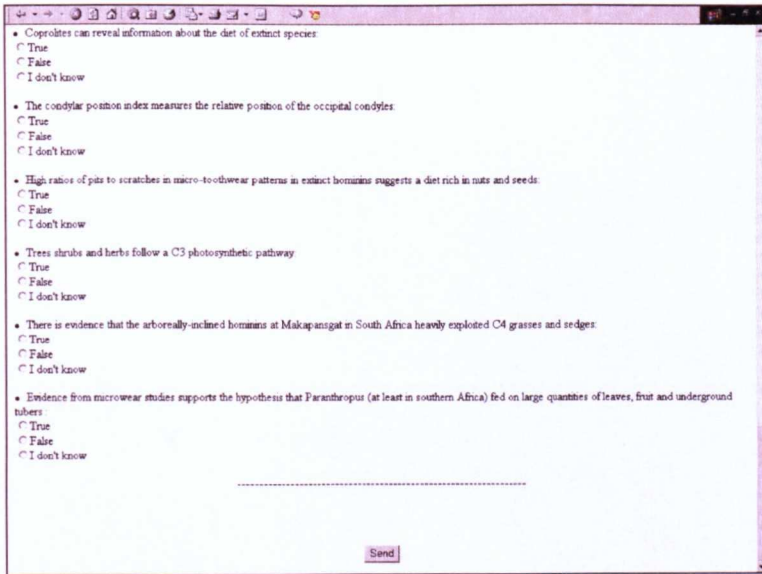
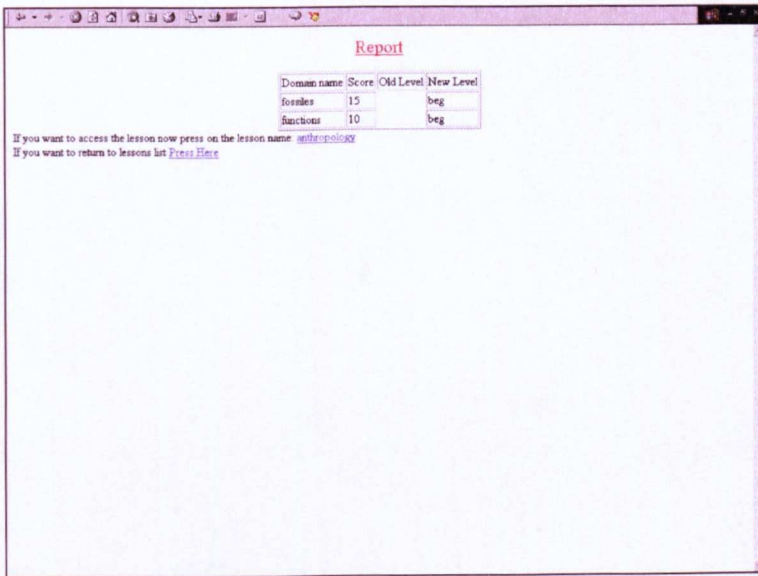


Fig 3

 - True
 - False
 - I don't know
- The jaw is buttressed near the position of the ascending ramus:
 - True
 - False
 - I don't know
- Dental wear gradients can be used to reveal information about life history patterns:
 - True
 - False

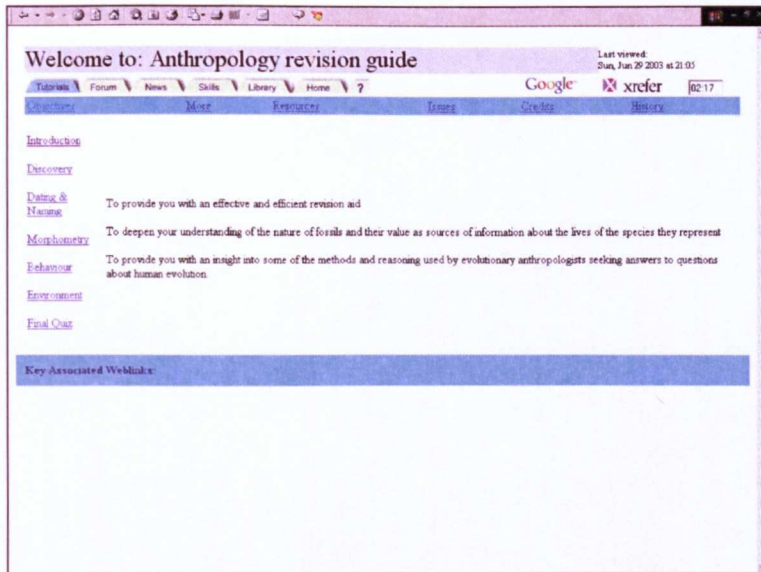


C.2.2 Results screen

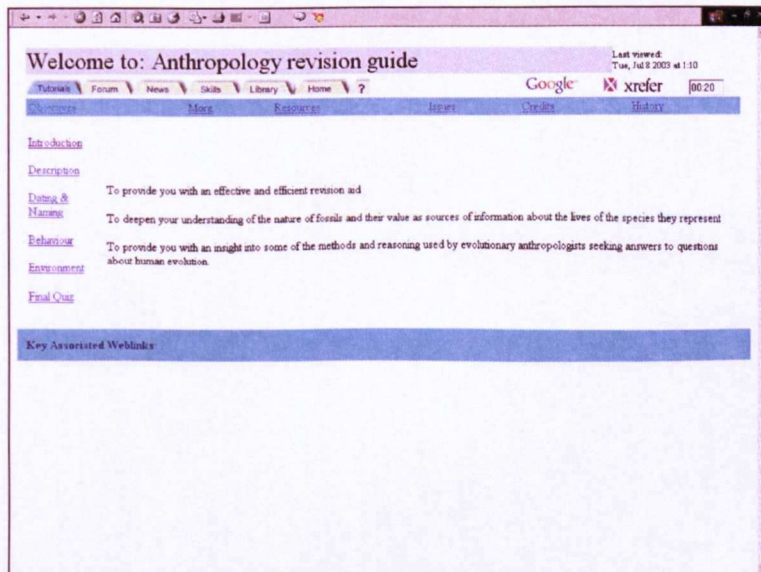


C.3 Lessons screen

Note: this is the Anthropology revision aid used in the user trial experiment.



Users with beginner knowledge level in both domains see this screen (7 levels)



Users with intermediate level in both domains see this screen (6 levels only)

C.4 Administrative tools

Registration Form

Student ID

First name

Last name

User name

Password

Confirm Password

Select a Category

Select Style

Student's registration form

Lesson Name

Choose involved Domains :

- chemistry
- math
- physics
- biology
- fossils
- html
- functions

Select a Category

Name Of The Pre-quiz [Write 0 for None]

Name Of The Post-quiz [Each Lesson Should have a Post-quiz]

Lessons' registration form

Appendix D: Guideline

D.1 How to create an adaptive lesson Plan

Lessons' authors may follow the following suggestions to create an adaptive lesson plan:

- Author(s) have to think about the associated domain(s) with a lesson to be created. For example, if the goal of the lesson is to teach students the following mathematical equation: $X^2 + 2*(X/Y) + \text{square-root}[(X*Y)*2] = Z$, then, author(s) may associate domains that deal with basic mathematical operations such as addition, multiplication, square-root calculation, etc.
- After specifying the involved domains and the main goal(s), authors need to specify what kind of assistance they will provide for users, with respect to the involved domain(s), to reach the level where they can understand the main goal. For example, by following the above mathematical equation, users may need to be aware about different mathematical operations. Thus, beginner users may need a kind of brief explanation for addition, multiplication, square-root calculations, etc. On the other hand, intermediate and advanced users may not need any revision.
- After specifying these major requirements, planning the lesson will start. In WHURLE in general, information could be broken up into small pieces called chunks, as each chunk is self-contained information. In Adaptive WHURLE, these chunks have two types: either general or conditional. As a suggestion, general chunks could be the chunk(s) that present the main goal(s) of the lesson and every-one will be able to access it. On the other hand, conditional chunks are those that have conditions to be fulfilled in order to be available.

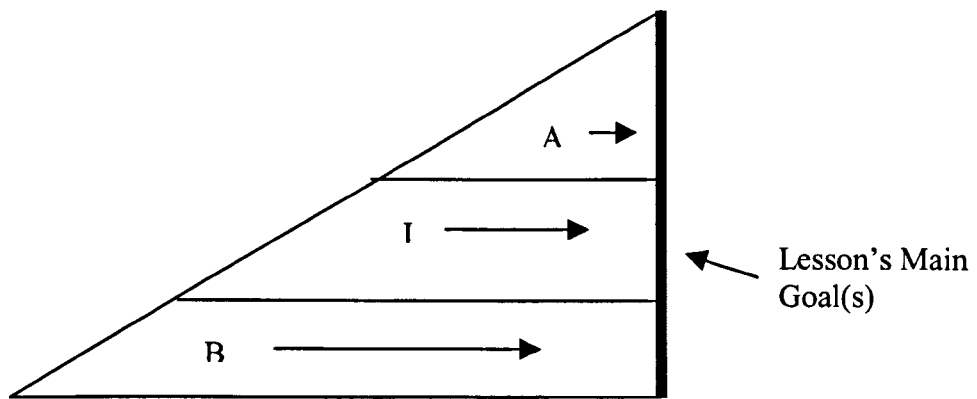


Figure D.1 Adaptive Lesson Map - the bold vertical line (vertical side of the triangle) presents the main goal(s) of the lesson. The first (lower) section shows how beginner users in any involved domain(s) will need more explanations about the necessary basics in that involved domain(s) that they may need to know to understand the main goal(s) of that lesson. The second (middle) section shows that users with intermediate knowledge level in any involved domain(s) may need fewer explanations about the basics that they have to know. In that case, those extra explanations could be considered as a reminder for the main concepts that they (users) need to remember to understand what the lesson is offering. The third (upper) section shows how advanced users do not need any of these reminders (depending on lessons' Author decision) and they could go directly to the new information the lesson is presenting in addition to some extra advanced information. B: Beginner, I: Intermediate and A: Advanced.

For example, following the mathematical example, there may be chunk(s), which belong to the addition domain, and provide a brief explanation about addition calculations. These chunks can only be “SEEN” by users who have a beginner knowledge level in addition. Thus, users who have a higher knowledge level will not see that extra explanation.

From the above guideline, it can be perceived that any lesson plan provides help for users who have a kind of shortage in certain areas to make them able to understand what the lesson is offering. Figure 1 shows a diagram that presents the conceptual idea about creating lessons.

In Adaptive WHURLE, each lesson could have a mandatory lesson to be finished before accessing the current one, and that could be specified inside the database.

D.2 How to create quiz questions

There are two types of quizzes in Adaptive WHURLE: pre-quizzes and post-quizzes. The type of each quiz is specified through the quiz itself. Moreover, authors have to

define what is called the mark-base. The mark-base is the base upon which the scoring depends. For example if the mark-base is 20, thereby for every correctly answered question the score will be added by 20 with respect to the overall quiz and with respect to every domain that that question belong. Thus, it is very important that in case a user has answered all his/her questions correctly, his/her **overall score should be 100 as a maximum**. The knowledge level for every involved domain is calculated as follows:

- Scores: As mentioned before, the overall score for a quiz should be out of 100 and that could be achieved by balancing between the mark-base and the number of questions. For example, if there are 10 questions then the mark-base should be 10, and in case of 20 questions, the mark-base should be 5 and so forth.
- Knowledge scale: that scale ranges from 1 to 10 and each number in that scale represents 10 numbers from the score. For example, a user answered some questions correctly and his score is 40 with respect to a particular domain. Thus, his/her knowledge will be 4 on the knowledge scale for that particular domain.
- Knowledge stereotypes: the current implemented stereotypes are beginner, intermediate and advanced. The beginner level embraces knowledge scales that range from 1 to 4, intermediate embraces from 5 to 8, and advanced from 9 to 10.

In the current implementation of the quiz engine two formats are supported: questions with image answers and questions with text answers.

D.2.1 Pre-quizzes

Every lesson in the system can be associated with a pre-quiz. That pre-quiz reflects users' knowledge levels with respect to the involved domains through the adaptation process of the presented material. Thus, authors have to specify questions about each involved domain in the lesson to get users' knowledge state about those domains. Moreover, each question could serve more than one domain up to all involved domains.

D.2.2 Post-quizzes

Every lesson must be associated with a post-quiz. That post-quiz tests users with respect to the main goal of the studied lesson and every involved domain. Users have to finish the post-quiz of each lesson in order to update the system with their new

knowledge level with respect to every involved domain, which will be reflected on other lessons that may use any, some of, or all of the involved domains in that lesson.

Appendix E: Users Quotes

A (positive response)

- *Good idea – could use more detailed information though. It's quite short. Was useful for revision purposes. (*)*
- *More on morphometry, may be something on key workers in the field would be good. (*)*
- *Content was a bit patchy – was really good for somethings, not as good for others. Also. Where were the results for the online exam? – I couldn't see them on the website. (*)*
- *Doesn't always work. I.e. problems logging in etc. otherwise good. Perhaps provide journal reference list (*)*
- *V.good. bit too slow, especially virtual lab. Good photos + links.*
- *Would be good to have a short self-test to be down at the end in the style of the class test.*
- *Good, can't think of any suggestions.*
- *Found it useful and easy to work with but top links not necessary as got to it via the website .*
- *Needs to be quicker.*
- *The First lesson was very useful but the pages are painfully slow to turn and I had problems accessing other lessons.*
- *When its up and running is should be very useful. A good idea*
- *The appearance could be improved by using different font which is easier to read. It would be better to have it in the same style as the learning support centre.*
- *A course test to practice for the MCQ would have been helpful*
- *More quizzes and tests to evaluate progress or a section going over the questions that were incorrectly answered in the first test with pointers of where to find information for the right answers (*)*
- *V.good. Perhaps more detail would be useful (*)*
- *Good. More depth required. (*)*
- *Wasn't extensive but what was there was useful (*)*

*: Responses that indicated more detailed information is needed.

B (negative response)

- *Personally, I did not find the revision aid useful. It was very slow. Also, I found it very basic and not very taxing.*
- *I only tried to use it once, but thought the online website was better for me in aiding my learning and revision.*

C (accessing problems response)

- *I tried to use my online revision aid but I had no record of my user name/password I chose, and there was no way to find out what it was. This meant I couldn't access the site.*
- *Crashes a lot + quite slow.*

- *Didn't work off campus near exam time. Very slow. Links don't work*
- *Couldn't get it work.*

Appendix F: Publications

Publications presented in this appendix are related to the Hybrid Model and to the new infrastructure adopted by WHURLE system, and they are as follows:

- *"The Hybrid Model for Adaptive Educational Hypermedia"*. Publication presented at the Adaptive Hypermedia 2002 (AH 2002) Conference. This is the first publication about the Hybrid Model. (Page 222)
- *"User Modelling, and Adaptive Hypermedia Frameworks for Education"*. Publication presented in New Review of Hypermedia and Multimedia (NRHM) Journal, 2002 Volume 8. This publication introduces the Hybrid Model and how it is implemented in the WHURLE system. *NOTE: The WHURLE-HM System Introduced throughout the thesis is referred to as "Adaptive WHURLE" in this publication.* (Page 228)
- *"Pluggable user models for adaptive hypermedia in education"*. Publication presented at the Hypertext 2003 Conference. This publication introduces the new infrastructure of the WHURLE system. (Page 240)

The Hybrid Model for Adaptive Educational Hypermedia.

Mohamed Ramzy Zakaria¹, Adam Moore², Helen Ashman¹, Craig Stewart³ and Tim Brailsford¹

¹School of Information Technology and Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK

mrz@cs.nott.ac.uk, {helen.ashman, tim.brailsford}@nottingham.ac.uk

²School of Civil Engineering, University of Nottingham, Nottingham, NG7 2RD, UK

adam.moore@nottingham.ac.uk

³School of Life and Environmental Sciences, University of Nottingham, Nottingham, NG7 2RD, UK

craig.stewart@nottingham.ac.uk

Abstract

Web-based distance learning is becoming increasingly prevalent as the Internet permeates every aspect of our culture, and many educational content management systems are now in use on the web. However, learners' experiences of these systems are almost invariably static, with information being delivered regardless of their background or knowledge. Due to variation between learners', it is suggested that these web-based distance-learning systems would benefit from the capability of adapting their content to meet individual needs. To effectively implement this adaptation of educational material, we require a user model that supplies the system with information about the learners using the system, such as their backgrounds, knowledge, interests and learning styles. This paper focuses on presenting a user model that combines the advantages of two techniques (overlay and stereotyping) in a way that provides the system with the ability to deliver information that is fully informed by the requirements of individual users.

Keywords: user modelling, adaptive hypermedia, educational systems, overlay, stereotyping

1 Introduction

The brave new era of the information age has ramifications for all disciplines, at the most fundamental of levels. From education, to commerce and music, the Internet impinges on every field where data and knowledge are currency. Arising out of this world-wide network of communications comes the globalisation of information – in which hypermedia tools are at the forefront enabling direct user access to information [1].

As the amount of information on the web continues its exponential increase, the number of users with different goals and interests also expands. It therefore becomes increasingly important that the information available be adapted to suit each user's individual knowledge and aspirations.

For example, in traditional web-based educational hypermedia systems, the contents are generally static, in so far as once written, their contents cannot be changed without external intervention. This provides a uniform learning experience to all learners, regardless of their needs and requirements. One example of such a system is WebCT [2]. This is a hypermedia educational system developed in 1995 at the University of British Columbia. It is an environment for authoring and delivering educational materials over the web. WebCT presents a static and inflexible pedagogic experience, without any kind of adaptation at the user level. Hence a web

application, such as those delivered via WebCT, which are designed with a particular class of users in mind, may not suit those even marginally different from the original target audience [3].

In response to this clear need, adaptive hypermedia systems have been created such as AHA[4] and CHEOPS[5]. They build a model of the goals, interests, preference and knowledge of their users; so that they may present them with the information they need in a timely and appropriate manner [1].

This paper describes a hybrid user model, which is cooperative [6], (i.e. it collaborates with users in gathering information, as they are required to supply the system with some personal information, e.g. their occupation and preferences). This model also involves users in the user modelling process, as the contents of topics or courses are adapted according to their knowledge level about the topic they study. The hybrid model described below has the benefit that it should suit any adaptive educational hypermedia system, and we suggest that it is likely to provide a powerful addition to any technology-based learning programme.

2 The Hybrid Model

2.1 Architecture

The hybrid model combines the use of two major techniques that are prevalent within the user modelling community. The first of these is the *overlay model* perhaps currently the most widely used technique of user modelling. This is used to measure the knowledge level of users in any given topic or domain. A user's knowledge according to this model is considered to be an overlay of the total knowledge representing that domain. This knowledge level is represented in the form of "Concept-Value" pairs, [7,8].

The second model is the *stereotype*; this technique assumes that knowledge is customised for specific groups, with each user being assigned to one and only one group at any given time. Thus, users who share the same background or knowledge should be assigned to the same group. Users can not change from one group (or class) to another until they trigger the specific conditions of the new group[9,10].

Aspects of each of these models are utilised by the hybrid model as follows:

- *Overlay technique*: the overlay measures the knowledge level of each learner within certain subject domains. This knowledge level might represent the score achieved in the system assessment at the end of each lesson, although any other parameters the system authors may choose may also be used. For example, the score achieved in self-assessment quizzes is a widely used and well-accepted metric of the comprehension of information.
- *Level stereotype*: level stereotypes mainly depend on the knowledge level of users. For example, they may simply be defined as Beginner, Intermediate and Advanced, but any classes may be used as appropriate to each system. According to the users knowledge level they will be assigned to a single class of the level stereotype within any given domain they of study. For example, a user studying biomechanics might be assigned simultaneously to the novice class in biology and to the advanced class in mathematics. Classes in the level stereotype are concerned with providing assistance that is appropriate, and adapting the contents of the lesson to suit the learner. Each class may define an article or set of articles, links to external documents, or to lessons in other courses. For example, if a user belongs to one of the advanced classes he may

be provided with advanced articles or links to help the user to find more about the topic or domain he studies. Level stereotypes not only adapt the contents to suite a user's level, but they also facilitate learning by identifying domain weaknesses in the topic under study.

- *Category stereotype*: the hybrid model has been designed for systems that simultaneously run multiple courses for different levels of users. For example, the system may be running courses for first year undergraduate as well as postgraduate users. For this reason the users need to be categorised, as the knowledge level of undergraduate users in a certain stereotype level of a certain domain may not be the same as that of postgraduate users in the same stereotype level of the same domain. For example, consider two users, one of them a first year undergraduate and the second one a studying for a higher degree. Both of these students are classified in the intermediate level stereotype for the biology domain. Both of them are in the same level stereotype, but the intermediate level of postgraduates will be much more advanced than that of first year undergraduates. The categories stereotype helps the system to distinguish between different users in the same level stereotype, and to provide each of them the appropriate adaptation and help.

There is one other important aspect of this hybrid model - the information pool. This is categorised by the domain model, and consists of a pool of articles, links, and other items that encapsulate the resources of an adaptive system. The information pool is likely to differ in both form and content from one system to another.

The overlay technique, level stereotype and category stereotype combine to pick from the information pool the most convenient articles and links that suit each user's level, knowledge and background. Thus, according to a user's knowledge level and category, the most appropriate materials will be chosen from the information pool. Figure 1 shows the hybrid model's components, comprising of two stereotypes and one overlay, to provide the maximum flexibility and to have the capability to serve a wide range of users.

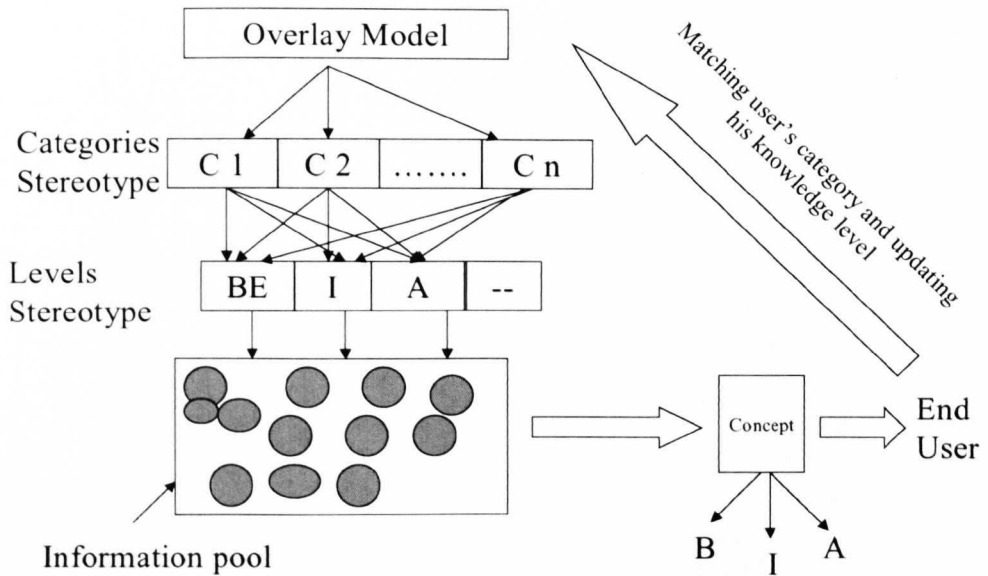


Fig. 1. The components of the hybrid model. An overlay model combines with a category and level stereotype to retrieve appropriate content from the information pool to convey concepts to the user. The user interacts with the system to inform and update the user model. BE – Beginner, B – Basic, I – Intermediate, A – Advanced.

2.2 Mechanism

When a user logs on to the system for the first time, he/she will be given an initial knowledge level value according to an estimate of prior knowledge about the subject under study. That recommends the user to a certain level stereotype, and the category is determined according to any parameters that the system authors may choose such as user's occupation. Each time a user passes from one lesson to another the knowledge level for that user is updated according to the score in the system assessment (as well as other parameters the system's authors may choose). According to the user's new knowledge level, the class assigned according to the level stereotype might be changed or might be the same (i.e. if the user still has the same knowledge level). The adaptation of the contents and the supporting articles are available according to the class of the user in the level stereotype as well as the category. The steps involved in adaptation are illustrated in Figure 2.

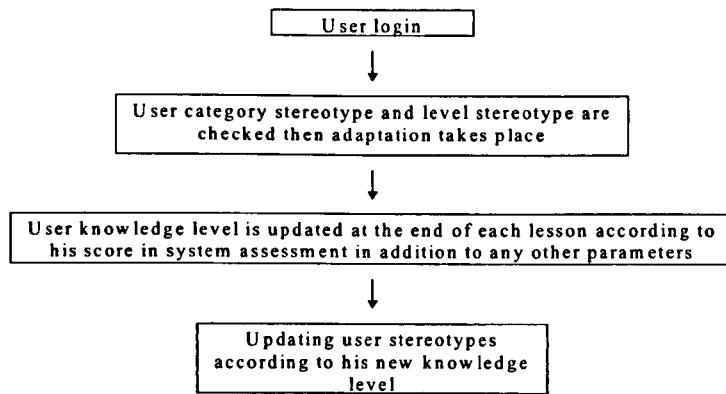


Fig. 2. The hybrid model mechanism to adapt materials to users. Once the user logs in, the system either adapts its material according to the existing user model, or creates a new one. At the end of each lesson, the user model is updated.

3 Implementation of the hybrid model in a hypermedia learning environment

The hybrid model is currently being integrated into an adaptive educational hypermedia system called WHURLE (Web-based Hierarchal Universal Reactive Learning Environment), [11, 12, 13] an XML-based integrated learning environment. In WHURLE the content consists of atomic chunks, each of which consists of the smallest conceptually self-contained unit of information that the author can envisage. The domain of WHURLE content is contained within the melange, which consists of all the available chunks in any single installation. In WHURLE conditional transclusion [14] is used to construct virtual documents, which contain one or more of these chunks. The hybrid model is integrated into WHURLE as the filter for the lesson plan. This filter generates an adapted virtual document dependant upon both the adaptation rules within the lesson plan and user profile. WHURLE lesson plans represent the information pool of the hybrid model [11, 12, 13].

4 Conclusion

The hybrid model is a user model that gathers together the most commonly used techniques of user-modelling for adaptive hypertext. This utilises the advantages of each of these techniques in a way to provide a full understanding for the user's needs and requirements on several different levels. Using this technique we have implemented adaptation within WHURLE to allow students to see pages of information containing one or more chunks in a manner relevant to their skills, knowledge and learning styles, following rules set by the author of the lesson that they are viewing. Using this model we hope that the WHURLE system will provide a strong pedagogic framework for a variety of web-based learning scenarios.

Acknowledgements

We wish to thank Peter Murray-Rust, Peter Davies, and Ban Seng Choo for many useful discussions, and colleagues in the WTG, VSB and IBiS for their support and encouragement. Craig Stewart is a research associate funded by the Hong Kong University Grants Committee.

References

- [1] Brusilovsky, P. (1996). "Methods and techniques of adaptive hypermedia". User modeling and user-adapted interaction, 6(2-3), pp. 87-129.
- [2] Beshears, Fred. "WebCT overview". <http://socrates.berkeley.edu:7521/articles/webct/WebCT-Presentation>
- [3] Eklund, J; Brusilovsky, P; Schwarz, E. (1997). "Adaptive Textbooks on the World Wide Web", Proceedings of AUSWEB97, the third Australian conference on the world wide web, Queensland, Australia, July 5-9, 1997, Southern Cross University press, pp. 186-192. <http://ausweb.scu.edu.au/proceedings/eklund/paper.html>.
- [4] De Bra, P; Calvi, L. (1998). "AHA Adaptive Hypermedia Architecture". NRHM journal, V.4, pp 115 - 139
- [5] Ferrandino, S; Negro, A; Scarano, V. (1997). "CHEOPS : Adaptive Hypermedia on the World Wide Web". Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS '97).
- [6] Kay, J. 1995, "The UM toolkit for cooperative user models". User Models and User Adapted Interaction 4(3), 149-196.
- [7] Valley, K. (1997). "Learning Styles and Courseware Design". Association of Learning Technology Journal, 5(2), p42-51
- [8] Carr, B; Goldstein, I. (1977). "Overlays, a theory of modelling for computer aided instruction". A technical report. AI memo 406, MIT, Cambridge, MA.
- [9] Rich, E. (1983). "Users are individuals: individualizing user models", Journal of man-machine studies vol.18, 199-214.
- [10] Benaki, E; Karkalestsis, V; Spyropoulos, C. (1997). "User modelling in WWW: the UMIE prototype". Proceedings of sixth international conference on user modelling, Chia Laguna, Sardinia, 2-5 June 1997 http://www.contrib.andrew.cmu.edu/~plb/UM97_workshop/Benaki/Benaki.html
- [11] Brailsford, T J ; Moore, A ; Stewart, C D ; Zakaria, M R ; Choo, B S ; Davies, P M C. (2001). "Towards a framework for effective web-based distributed learning". WWW10 proceedings, HongKong.
- [12] Moore, A; Brailsford, T. J, Stewart, C. D. (2001). "Personally tailored teaching in WHURLE using conditional transclusion". The twelfth ACM conference on hypertext and hypermedia. August 14-13, 2001, Denmark.
- [13] Brailsford, T; Stewart, C; Zakaria, M; Moore, A. (2002). "Autonavigation, Links and Narrative in an Adaptive Web-Based Integrated Learning Environment". Proceedings of www2002, Hawaii, USA.
- [14] Nelson, T. H. (1995). "The Heart of Connection: hypermedia unified by transclusion". Communications of the ACM.

User Modelling, and Adaptive Hypermedia Frameworks for Education

Mohamed Ramzy Zakaria and Tim Brailsford

School of Information Technology and Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK

e-mail: mrz@cs.nott.ac.uk; Tim.brailsford@nottingham.ac.uk

Abstract

In this paper, we give an overview about the hybrid model, which is a generic user model that is based on measuring and classifying users' knowledge with respect to multiple knowledge domains simultaneously. In addition, we demonstrate how that model is implemented through the WHURLE (World Hierarchal Universal Reactive Learning Environment) framework, which is an adaptive educational hypermedia framework where different domains could be involved at the same time to concurrently serve a wide range of users of different educational states and of different abilities and backgrounds.

Keywords: User Modelling, Hybrid Model, Adaptive Hypermedia, Education, Learning Environments.

1 Introduction

The amount of information on the web is continuing in its exponential increase, as is the number of users with different goals and interests. It is thus becoming increasingly important that the information provided be adapted to suit each user's individual knowledge and aspirations. Adaptive hypermedia systems are achieving this goal by employing individual user models. User models represent the individual users' knowledge, background, goals and other features that enable these systems to be adapted to the needs and the requirements of that user.

The educational area is considered to be the widest application arena where adaptive hypermedia systems are deployed (1), as it has a wide variety of users with different knowledge, background and goals.

The main purpose of this paper is to give an overview of the hybrid model (2), which is a user model that is mainly designed for adaptive educational hypermedia frameworks. These may run several courses - involving multiple knowledge domains - for a wide range of users with diverse knowledge, backgrounds and goals. We will also demonstrate the potential of this model by showing how it may be implemented through the educational hypermedia framework WHURLE (Web-based Hierarchal Universal Reactive Learning Environment) (3).

Before exploring the hybrid model and its implementation we will briefly review the user models utilized in some of popular adaptive educational hypermedia systems,

such as Metadoc (4) and CHEOPS (5), in section 2. In section 3, we will present an overview about the hybrid model as an abstract model. Thereafter, in section 4 we will introduce the hybrid model implementation through WHURLE.

2 Adaptive Hypermedia in Education

In recent years many adaptive educational hypermedia systems have been developed, a good example of which is Metadoc (4). The Metadoc system provides hypertext documents to potential readers in an adaptive way that suits their knowledge. Moreover, it provides its adaptive capability through the use of an interactive agent that stores knowledge about users in a user model, and that knowledge is used to vary the level of detail presented in the document for each user individually. Furthermore, if a user decided explicitly to modify the level of the presented details, by means of stretchtext operations, the user model is informed and future presentation of information may change. Metadoc classifies users with respect to their knowledge about AIX/Unix and general computer concepts into four classes: novice, beginner, intermediate, and experts. Likewise, AIX/Unix concepts and general computer concepts are classified into different concept levels using the same scale. Thus, users' knowledge level about AIX/Unix and general computer concepts is independently stereotyped, i.e. a user may be a novice in AIX/Unix and a beginner in general computer concepts.

To present the correct level of information, the system varies the amount of explanation through the stretchtext technique depending on an individual user model. Thus, users who belong to a classification whose level is less than the difficulty of a given concept is assumed to be unfamiliar with that concept and extra explanation via stretchtext is provided. On the other hand, expert users would want more in-depth details rather than explanations.

At the *Dipartimento di Informatica ed Applicazioni* of the University of Salerno in Italy, the CHEOPS system (5) has been developed. CHEOPS is a server side implementation of a session-based interaction model. The hyperdocument in CHEOPS is divided into categories to help users in navigation through the presented information. CHEOPS uses a knowledge pyramid, which consists of two normal polygons with vertices that represent the hyperdocument categories. The upper polygon represents the minimum knowledge level and the lower one represents the maximum knowledge level. The edges of the pyramid represent a user's knowledge level in each category.

The user model in CHEOPS classifies the users' knowledge level into novice, amateur and expert. Moreover, a user knowledge level in each category depends on his previous interaction within the category, and this is called a confidence level. The user has the ability to change the confidence level given to him by the system. CHEOPS uses the knowledge pyramid as an important navigational aid to users; at each step, users know their confidence level in each category and act upon that. Categories could be dependent over each other. Hence, according to a user knowledge level in other categories, another category/categories could be accessible to him. The adaptation mechanism in CHEOPS takes the required document as an input, and creates on fly a document that has all the links changed according to a user profile, which is modified each time the user chooses a link.

By analysing the user model employed by these and other systems, we found that they maintain the knowledge value of each individual user, in addition to other characteristics, with respect to concepts that belong to a single domain.

The question that then arises is: what if we want to involve more than one domain, such as mathematics, biology, chemistry, and so forth, at the same time? Also, how do we maintain users' knowledge with respect to each individual domain? (which is important because there are interdependencies between concepts from different domains). For example, in biochemistry, we might use two domains - biology and chemistry. The system then has to be capable of handling the prerequisites of any single involved domain in that subject, such as a certain knowledge level in chemistry and in biology, with respect to each individual user.

Metadoc in some way has tried to solve this issue, to a limited extent, by classifying users with respect to their knowledge level into four stereotypes for every involved domain. Thus, for an n number of involved domains, the system should initiate an n number of four stereotypes, which is not an adequate solution.

We claim that, to create an adaptive educational hypermedia framework, the users' knowledge should not be limited around a topic's concepts, but be extended to domains - as each domain contains topics that help to build users' knowledge about that domain. Moreover, multiple domains should be involved and integrated together in a single domain model, and there must be no limit to the number of involved domains. Furthermore, the user model should be capable of maintaining users' knowledge about each involved domain. As a result, we came up with the fundamental idea behind the hybrid model.

3 The Hybrid Model

The hybrid model (2) is a generic user model for adaptive educational hypermedia frameworks that simultaneously run different courses involving multiple knowledge domains for users of different goals, knowledge and backgrounds. Thus, it is based on measuring and classifying a user's performance within knowledge domains, such as the biology domain, the mathematics domain, or any other domains involved in an educational curriculum.

Briefly, the hybrid model is composed of an overlay model, level stereotypes, category stereotypes, and an information pool, as shown in Figure 1. The Overlay model, the most widely-used technique in user modelling, measures the knowledge level of each learner within subject domain (6) and is represented in the form of "Concept-Value" pairs (7; 8), but in the case of the hybrid model it is represented in the form of "Domain-Value" pairs. Moreover, evidence indicated that overlay modelling significantly increases the appropriateness of material's explanation (8).

Level stereotypes depend on the knowledge level of users, as they may simply be defined as Beginner, Intermediate and Advanced, or any other classes that systems' authors may choose. According to users' knowledge level about different domains, they are assigned to a single level stereotype with respect to every single involved domain. For example, a user studying biomechanics might be assigned

simultaneously to the novice class in the biology domain and to the advanced class in the mathematics domain.

What is more, category stereotypes play a crucial role with respect to the hybrid model in differentiating between users that belong to the same level stereotype. For example, consider two users: one of them is a first year undergraduate and the second one is a student for a higher degree, and both of them are classified as intermediate in the level stereotypes for the biology domain. Despite both students belonging to the same level stereotype, the intermediate level of the postgraduate student is much more advanced than the undergraduate one. Category stereotypes solve that problem by assigning users according to their type of study or occupation into different categories. For example, undergraduate students belong to the undergraduate category and postgraduate students belong to the postgraduate category. Thus, members of each category are provided with information that suits their knowledge level with respect to their category. Finally, the information pool is categorised by the domain model, and consists of a pool of articles, links, and other items that encapsulate the resources of an adaptive system for the given domains. The information pool is likely to differ in both form and content from one system to another. Level stereotypes pick from the information pool the most convenient articles and links that suit the knowledge and the category of each individual user.

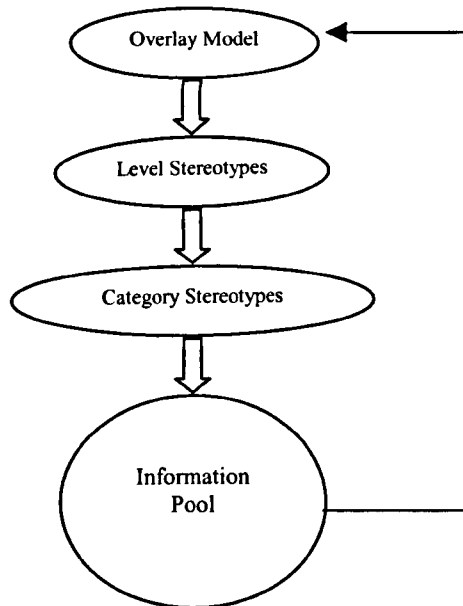


Figure 1: The components of the hybrid model. An overlay model combines with a level and category stereotype to retrieve appropriate content from the information pool to convey materials to the user. The user interacts with the system to inform and update the user model.

This section is divided into six subsections. The first one gives a brief background about WHURLE, and the second subsection describes how the hybrid model information pool is represented in WHURLE. Also, the third subsection describes how knowledge domains are presented through the system, while the fourth one discusses how knowledge level about each domain is classified, measured, and

updated. Subsection five explains how educational materials in WHURLE could be adapted. Finally, in the sixth part the adaptation mechanism is explained.

4.1 WHURLE background

To exploit the functionality of the hybrid model, it has to be implemented through a strong educational hypermedia framework such as WHURLE, which is an XML-based integrated learning environment. WHURLE is a server-based system delivering HTML (or possibly in the future XHTML) dynamically generated by XSLT (extensible style sheet: transformation). In WHURLE the content consists of atomic chunks, each of which consists of the smallest conceptually self-contained unit of information that the author can envisage, where these chunks are totally transparent to the user/learner. All the available chunks are contained in a melange. This melange acts as a huge pool where all the chunks of all involved domains are contained. What an end-user will see is a lesson, which is an apparent docuverse created by the WHURLE system. This contains the contents of any number of chunks together with navigational links and an overlaid environment that is generated by the system. The lesson is defined by another XML file that is called a Lesson Plan, which consists of WLPML (WHURLE Lesson Plan Markup Language).

The lesson plan contains a hypermedia pathway through the melange that is created by teachers using WHURLE (although default lesson plans are provided with a melange distribution). In its simplest conceptual form, a lesson plan consists of a hierarchy of levels, each containing one or more pages. Pages consist of chunks transcluded by means of Xinclude (9). The processing of Xinclude is orthogonal to both parsing and validation, which means that chunks are retrieved as required, rather than during the parse phase. Thus, there is a relatively modest processing overhead at parse time, and the server load is spread evenly during use.

4.2 the hybrid model information pool

Any lesson plan in the WHURLE system is composed of a set of chunks (default narratives). The information pool component in the hybrid model is composed of those chunks, as shown in Figure 2. Thus, the contents of the information pool differs from one lesson plan to another.

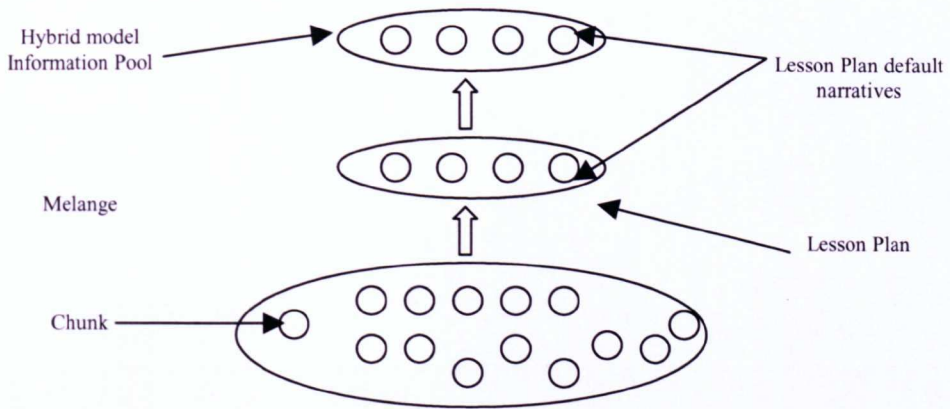


Figure 2: Lesson plan's default narratives are a subset of the Melange and the hybrid model information pool includes all the default narratives of the lesson plan.

4.3 knowledge domains

The hybrid model depends on measuring and classifying users' knowledge with respect to involved domains in an educational curriculum. Thus, we have to find a way to classify domains into sub-domains and sub-sub-domains and so on. Because of that, we decided to use the same approach used in the Dewey Decimal Classification (10) (DDC). DDC is a general knowledge organization tool that is continuously revised to keep pace with knowledge. The DDC numbers are featured in the national bibliographies of sixty countries. The system was established by Melvil Dewey in 1873 and first published in 1876. Furthermore, it is used by many libraries around the world to classify their collections.

4.4 knowledge levels

Users' knowledge level, in the current implantation, in WHURLE is represented through three stereotypes: novice, intermediate, and advanced. This kind of stereotyping is used by other adaptive educational hypermedia systems, such as CHEOPS (5) and Metadoc (4).

The concept behind stereotyping users' knowledge level is not only to provide advanced users with advanced information, or novice users with basics, but also to provide a kind of assistant. Thus, novice users could access other chunks from different domains or the same domain, or access links to other resources over the web, in addition to the presented concepts that suit their knowledge level. Likewise, advanced users could find more interesting advanced information about the topic they study. However, linking to other resources either within the system or outside it relies on the lessons' authors.

Users' knowledge level about involved domains is updated through answering quizzes, and taking tests at the end of each lesson. Moreover, each question in a test or a quiz represents either one or more of the involved domain(s). According to a user's correct answers with respect to every involved domain question(s), his/her knowledge level about each involved domain is determined through a knowledge

scale, which is a numerical value that ranges from 1 to 10. Additionally, for each range of scores, there is a corresponding knowledge value. For example, the knowledge value 1 corresponds to quiz scores that range from 1 to 10 (out of 100), and the knowledge value 2 corresponds to quiz scores that range from 11 to 20, and so forth.

Level stereotypes rely on the knowledge value of each user with respect to each individual domain to assign him/her to the appropriate expertise level. For example, the novice level might embrace users with knowledge values ranging from 1 to 4, the beginner level ranges from 5 to 8, and the advanced level ranges from 9 to 10. Furthermore, according to users' knowledge value, their level of expertise changes either positively or negatively, i.e. upgrading or downgrading. Thus, the system is tracking the performance of each user in each domain, and provides help whenever it is needed.

4.5 adaptive lesson plans

In adaptive WHURLE, each lesson plan has its own prerequisites, such as mandatory lessons to be taken before it, which are stored in a MySQL database. Thus, for a user to access a lesson plan he/she has to satisfy its prerequisites; also he/she has to be a member of the same category that that lesson plan serves. That approach helps in case of a topic or a course composed of more than one lesson plan and they should be taught in a certain order; also in case an author of a lesson plan has found that knowing a certain lesson before the user accesses will be useful.

In addition, each lesson plan may contain one or more level(s), which could be nested inside each other. Furthermore, levels are composed of one or more pages that embrace one or more chunk(s). In Adaptive lessons plans, two types of chunks are defined: non-conditional chunks, which do not have prerequisites to be met, and conditional chunks that do have prerequisites to be fulfilled by users before their inclusion. The type of each chunk is defined through a domain attribute, which either holds the name of the domain the chunk is serving, or the value "general" that indicates the non-conditional type of that chunk. Moreover, every conditional chunk has two other attributes in addition to the domain attribute: *stereotype1* and *stereotype2*. Those attributes determine the required knowledge level(s) to access that chunk, as at least one of them should not hold a NULL value, as shown in Figure 3.


```

<level name="introweb-intro" title="An outline of Internet History and Function">
  <page>
    <chunk domain="general" >introweb001</chunk>
    <chunk domain="general" >introweb002</chunk>
    <chunk domain="general" >introweb003</chunk>
  </page>
<level name="design-intro" title="Introduction to Web Design">
  <page>
    <chunk domain="general" >webgr001</chunk>
    <chunk domain="web_design" stereotype1="beg" stereotype2="int">webgr005</chunk>
  </page>
</level>
<page><chunk domain="html" stereotype1="beg" stereotype2="">introweb006</chunk></page>
<page><chunk domain="html" stereotype1="beg"
stereotype2="int">introweb007</chunk></page>
</level>

```

Figure 3: A simple extraction from an adaptive lesson plan about html. Chunks with domain attribute general will be available to every user accessing this lesson regardless of his/her knowledge level about the involved domain. Chunks with domain attribute html and stereotype “beg”, means that the user should have a beginner stereotype knowledge level to access this chunk, where “beg” abbreviates for Beginner and “int” abbreviates for Intermediate

Thus, a user has to fulfil one of them, if both of them hold a knowledge value to be met, to gain access to that chunk. The reason we have these two stereotype attributes is that we believe any chunk that is essential for users with lower knowledge level to know could be utilized by users with higher knowledge level to understand a new piece of information, and that depends on lessons’ authors to decide. Furthermore, two attributes are used, and not more or less, because in the current implementation we only have three knowledge level stereotypes. What is more, every lesson a user has finished is recorded into his model for future revising, if needed.

4.6 adaptation mechanism

According to Brusilovsky (1, 11, 12), there are two kinds of adaptation: adaptive presentation and adaptive navigational support. The idea of adaptive presentation is to adapt the content of a page accessed by a user to suit his/her knowledge, goals and other characteristics. On the other hand, the adaptive navigational support helps users to find their path through hyperspace by adapting link presentation to the knowledge, goals and other characteristics of each user.

According to the nature of WHURLE’s infrastructure, these two kinds of adaptation are combined together in the adaptation process. As stated before, any lesson plan is composed of a set of chunks organized under pages and these pages included inside levels. When a user clicks on a level link, the first associated page within that level will be included with its available chunks. If any page within any level does not have any chunk to be included the link to this page is removed, and the same thing happens if a level does not have any pages to be included. Thus, link removal technique is used to adapt the content of the lesson plan, and thereby adaptive

representation and adaptive navigational support are combined together, as shown in Figure 4.

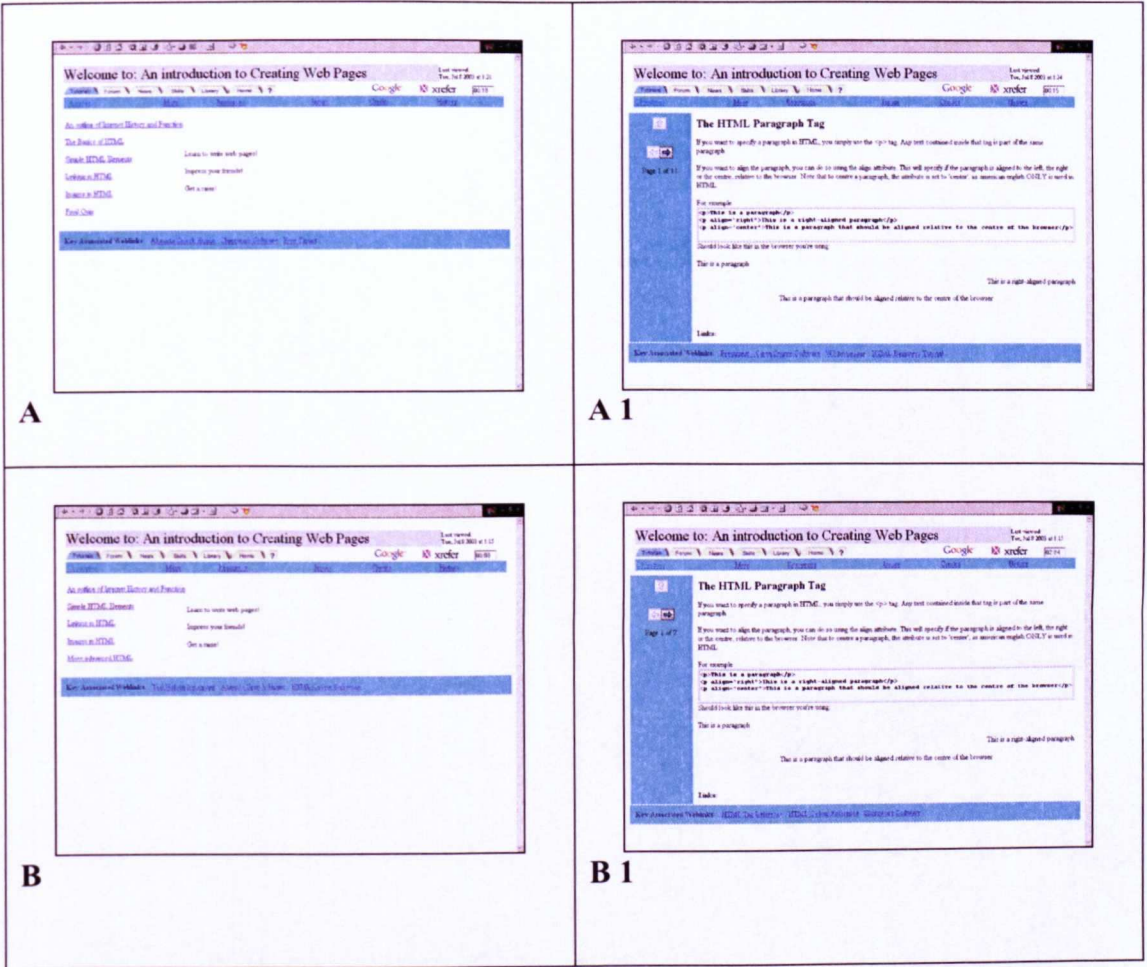


Figure 4: Image A represents the first page of a non adapted version of the html lesson plan where links to all levels (6 levels) are found. While Image A1 represents the content of the second level, as it is composed of 11 pages, a user can access them by means of the navigational buttons on the left frame. On the other hand, image B represents the adapted version of the first page of the same lesson plan, where links to five levels are only found and the sixth one is removed as none of its pages has chunks to include. Image B1 represents the content of the same level that image A1 represents, but with only links to 7 pages not 11, where the other 4 pages the user hasn't met their prerequisites, in addition non of them is of non conditional type.

The adaptation engine in WHURLE acts as a filter to lessons' content, where conditional chunks whose prerequisites are met and non-conditional chunks are included. Thus, a user will access information that suits his/her knowledge level. Through the history links, which could be found on the toolbar at the far right side in Figure 4 (images: B and B1), users may access all visited lessons in a non-adaptable version (without excluding any chunk) through the pop up history window, as shown in Figure 5. The idea behind that approach is to build a kind of library for every individual user, composed of all visited lessons in a row format without adaptation, as he/she can refer back whenever he/she wants to maintain information about studied domains.

Lesson Name	Score	Finished Date
basic-lesson-level3	30	10-Sep-2002
intro-web-am-01	70	30-Jan-2003

Figure 5: when a user clicks on the history link, as in Figure 4-Image B or B1, a popup history window come into view, which include links to all visited lessons, in a non adapted version as in Figure 4-Image A and A1, in addition to the total score the user got at the end of each of them, also the date in which he/she finished each lesson.

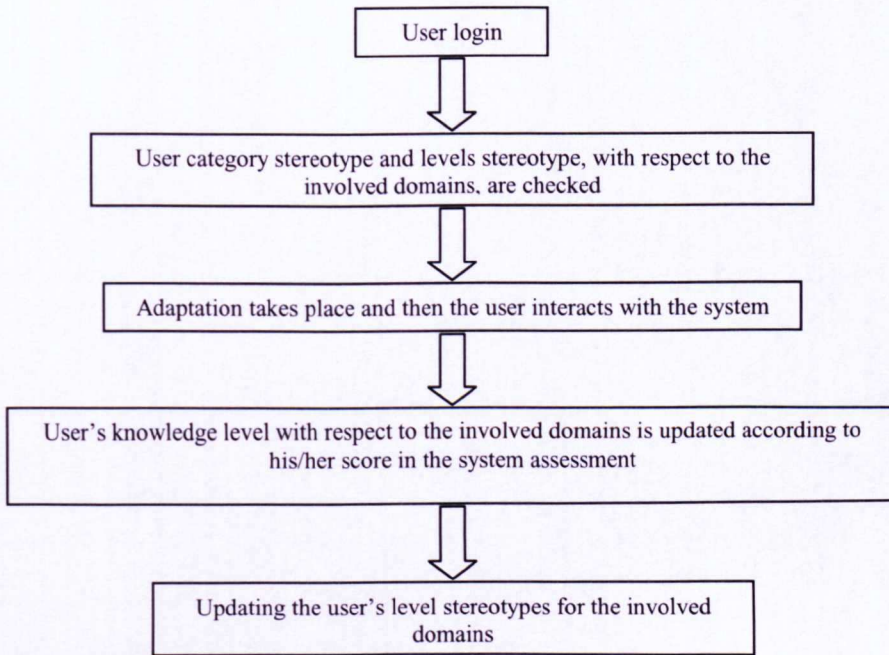


Figure 6. Once the user logs in, the system either adapts its material according to the existing user model, or creates a new one. At the end of each lesson, the user model is updated.

The hybrid model used within WHURLE is a cooperative (13) user model, because it collaborates with users in gathering information, as they are required to supply the system with personal information when they access the system for the first time, e.g. their occupation/category, preferences, and other entities. Figure 6 illustrates the involved steps in the adaptation process.

In Figure 6, when a user logs on to the system, he/she supplies his/her username and password to verify his/her authentication. Subsequently, the user's category is compared with the category of the lesson that he/she intends to access. Thus, if

categories do not match, the user will be declined. But if matching is defined, the user's knowledge level about each involved domain in the lesson plan is checked. Afterwards, the adaptation process takes place and the user starts to navigate through the adapted information. Hence, at the end of each lesson, the user may answer a quiz or take a test, and depending on his/her score, his/her knowledge level about each involved domain is determined. Therefore, the user is re-assigned to one of the level stereotypes for each involved domain.

In the case of new users, they have to supply the system with personal information, such as their name, category, etc. Also, they have to answer a quiz with respect to the lesson they like to access. According to this information and to the score they got in the quiz, their knowledge level about every involved domain is maintained, and thereby adaptation takes place.

The WHURLE system is implemented using the Cocoon (14) publishing framework from APACHE software foundation (which is a servlet-based open-source engine currently under heavy development and with a rapidly maturing feature set). The MySQL (15) database is used to store the user model information, lessons' sequences, knowledge scale, etc. The adaptation engine in WHURLE is implemented using XSP (16) (extensible Server Pages), which allows the writing of JAVA code inside an XML file, that is responsible for filtering lessons' content with respect to users' knowledge level. Also, it is responsible for managing session variables. Finally, esql (17), which is an XSP logicsheet, manages the communication between the system and the database.

5 Conclusion

The hybrid model is an attempt to reinterpret the most common techniques utilized in the world of user modelling. It combines these techniques together in a flexible way that can be used in a wide variety of educational contexts. Although the components of this model are derived from early researches, it has a number of novel features – particularly:

- It is capable of measuring and classifying users' knowledge with respect to every domain involved in any educational hypermedia framework, which embraces multiple different domains to teach, at the same time, users with different backgrounds, educational states, and goals.
- The systems' authors can choose any parameters that suit their application with respect to any of the model's components. For example, level stereotypes could be up to any number of classes and may include learning styles in addition to knowledge levels, categories could be up to any number of involved categories as those categories could be geographical, educational, etc., and also the way the overlay model is updated is totally left open to the authors.

We have demonstrated these features by describing how the model is implemented through the WHURLE system. Details about the technical implementations and user interface are beyond of the scope of this paper. Rather, we tried to have focussed on the core concepts underlying the WHURLE framework. Details about the navigational system and the interface implementation within WHURLE are given elsewhere (3).

References

- (1) Brusilovsky, P. Adaptive Hypermedia: from Intelligent Tutoring Systems To Web-based Education. *Proceeding of ITS 2000*, Montreal, Canada, 2000, 1-7.
- (2) Zakaria, M, R and Moore, A and Ashman, H and Stewart, C and Brailsford, T. The hybrid model for adaptive educational hypermedia. *Proceedings of AH2002, The Second International Conference on Adaptive Hypermedia*, Malaga, Spain, 2002, 580-585. Poster.
- (3) Brailsford, T and Stewart, C and Zakaria, M, R and Moore, A. Autonavigation, links and narrative in an adaptive web-based integrated learning environment. *Proceedings of www2002*, Hawaii, USA, 2002.
- (4) Boyle, C. and Encarcion, A. Metadoc: an adaptive hypertext reading System. *User Modeling and User-adapted Interaction*, 4, 1994, 1-19.
- (5) Negor, A and Scarano, V and Simari, R. User Adaptivity on the WWW through CHEOPS. *Proceedings of the 2nd workshop on Adaptive Hypertext and Hypermedia*, Pittsburgh, USA, 1998.
- (6) Brusilovsky, P and Eklund, J and Schwarz, E. A tool for developing adaptive electronic textbooks on the WWW. *Proceedings of WebNet'96 - World Conference of the Web Society*, San Francisco, CA, AACE, 1996, 64-69.
- (7) Valley, K. Learning Styles and Courseware Design. *Association of Learning Technology Journal*, 5(2), 1997, 42-51.
- (8) Carr, B and Goldstein, I. *Overlays, a theory of modeling for computer aided Instruction*. A Technical Report. AI Memo 406, MIT, Cambridge, MA, 1977
- (9) <http://www.w3.org/TR/xinclude>
- (10) http://www.oclc.org/dewey/about/about_the_ddc.htm#history
- (11) Brusilovsky, P. Efficient Techniques for Adaptive hypermedia. In: C. Nicholas and J. Mayfield, eds. *Intelligent Hypertext: Advanced techniques for the World Wide Web*. Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1997, 12-30
- (12) Brusilovsky, P. Methods and Techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, 6 (2-3), 1996, 87-129
- (13) Kay, J. (1995), The UM toolkit for cooperative user models. *User Models and User Adapted Interaction*, 4(3), 1995,149-196.
- (14) <http://www.apache.org/cocoon1>
- (15) <http://www.mysql.com>
- (16) <http://www.apache.org/xsp>
- (17) <http://www.apache.org/esql>

“Pluggable” user models for adaptive hypermedia in education.

M.R.Zakaria
School of Computer Science and IT,
University of Nottingham,
Nottingham, U.K.
mrz@cs.nott.ac.uk

A.Moore
School of Computer Science and IT,
University of Nottingham,
Nottingham, U.K.
axm@cs.nott.ac.uk

C.D.Stewart
School of Computer Science and IT,
University of Nottingham,
Nottingham, U.K.
cds@cs.nott.ac.uk

T.J. Brailsford
School of Computer Science and IT,
University of Nottingham,
Nottingham, U.K.
tjb@cs.nott.ac.uk

ABSTRACT

Most adaptive hypermedia systems used in education implement a single user model – inevitably originally designed for a specific set of circumstances. In this paper we describe an architecture that makes use of XML pipelines to facilitate the implementation of different user models.

Categories and Subject Descriptors

H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia – *architectures, navigation, theory, user issues.*

General Terms

Design, Experimentation, Human Factors.

Keywords

Adaptive hypermedia, education, system architecture, user modeling, XML

ADAPTIVE HYPERMEDIA IN EDUCATION

As on-line learning is becoming increasingly pervasive, so is an appreciation of the difficulties and limitations of this approach to education. The most important issue is that using the WWW to distribute resources - as though it were a distributed photocopying machine - is of limited educational value. Although this approach can be useful to support teaching, it is unlikely in itself to provide a sound learning experience. One approach to addressing this is to develop on-line collaborative systems that foster the creation of learning communities. Another approach - one that

we are concerned with here - is to leverage the techniques and technologies of adaptive hypermedia to deliver educational content that is appropriate to the needs of individual students.

Education is currently the major application of adaptive hypertext, and is an area that has been reviewed thoroughly by Brusilovsky [1]. A number of adaptive systems have been developed for educational use - such as AHA! [2]; CHEOPS[3]; Interbook [4]; and WHURLE [5]. These systems use a variety of techniques - operating at the level of the links, the content or both - to adapt the learning experience to suit individuals. One feature that they all have in common is that they construct a profile of individual users, and apply a set of criteria, the user model, to inform the adaptation. There are significant differences between the approaches to user modeling that these systems use [6]. Any user model is imperfect from an educational point of view, in so far as it makes pedagogic assumptions that are unlikely to be valid under all circumstances. Hence it would be highly advantageous - both for research purposes, and for practical implementation in teaching - to design adaptive systems in such a way that user models are independent modules that can be "plugged" into the system. In this paper we shall describe part of the WHURLE architecture that has been designed to do just this.

THE WHURLE SYSTEM

WHURLE is an XML-based adaptive learning environment [5]. In WHURLE content is stored as conceptually self-contained "chunks" of information - each chunk being described by a separate XML file.

The student experience - lessons - are defined by lesson plans, each of which contains a list of all of the chunks utilised in that lesson, the default structure of the information and various meta-information and configuration settings.

The WHURLE rendering engine is an XSLT stylesheet that provides a navigational overlay (navigation information being derived from the structure of the lesson) and a user interface that is specified as a skin.

ADAPTATION BY FILTER

The concept of XML pipelines, are fundamental to the WHURLE architecture. An XML pipeline is a series of events, generated at parse-time, that flow through a predefined sequence of filters or processors. Just as a Unix pipeline uses the output of one program as the input of another, an XML pipeline uses the output document of one process as the input document of another. In WHURLE, the lesson plan is constructed into a node-tree (using XInclude[7]) that contains all of the chunks that might be required in that lesson. This is ultimately passed - in an XML pipeline to

the display engine, an XSLT stylesheet that adds autonavigation and the user interface, and finally generates an HTML output document. This pipeline is thus the ideal place to implement a pluggable user model, as illustrated in Figure 1. XML pipelines are discussed in the Cocoon documentation, found at: <http://cocoon.apache.org/2.0/userdocs/concepts/index.html#basic-mechanisms>.

The output document of the lesson plan contains all of the content that could potentially be in the lesson. This is then filtered by an adaptation filter (ie an XSLT stylesheet that removes those chunks that are not required for the current user – as determined by whichever user model is currently in use).

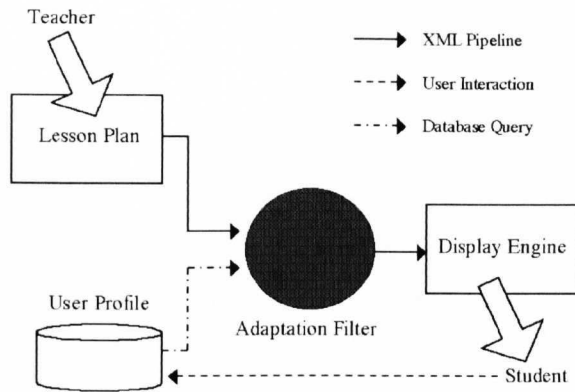


Figure 1. The adaptation filter in WHURLE. A teacher creates a lesson plan, which defines all possible content of the lesson. This is filtered, according to the user model, which makes use of information stored in the user profile. The output of the filter is rendered by the display engine, and student interactions update the user profile.

Although a larger node-tree is generated than will usually be delivered to any individual user, the computational overhead is minimal due to the processing model of XInclude [7]. Because the information is processed at the level of an XML information set, the only elements processed are the top-level included items, unless the child elements are required elsewhere in the pipeline.

Currently we use an adaptation filter that implements the “Hybrid Model” [8, 9], although we have also used WHURLE as a framework to deliver non-adaptive material by the simple expedient of passing the entire node-tree through to the rendering engine. In order to implement a different user model, the only part of the system that needs to be modified is the adaptation filter. It is, however, important to note that depending upon the criteria used for adaptation, some user models might require different meta-information to be stored in the chunks and/or lesson plans.

SUMMARY

The use of pluggable user models in WHURLE increases its flexibility, and also provides us with a potentially powerful research tool for future evaluation studies. User models are currently under development that operate on factors other than that of knowledge and ability; such as preferred learning styles. Also, WHURLE is to be a test system for user models developed by the Minerva:ADAPT project (<http://wwwis.win.tue.nl/~alex/HTML/Minerva/>). All of these applications are facilitated by the flexibility of this architecture.

ACKNOWLEDGEMENTS

We are grateful to the EU Minerva programme for funding under the ADAPT project. We should also like to thank Helen Ashman, and other members of the Web Technology Group in the University of Nottingham for their support and for many helpful discussions

REFERENCES

- [1] Brusilovsky, P. (1998). Adaptive Educational Systems on the World Wide Web: A review of available technologies. *Proceedings of the Workshop "WWW-Based Tutoring" at the 4th International Conference on Intelligent Tutoring Systems (ITS'98)*.
- [2] De Bra, P. & Calvi, L. (1998). AHA! An open Adaptive Hypermedia Architecture . *The New Review of Hypermedia* 4, 115 – 139.
- [3] Ferrandino, S; Negro, A & Scarano,V. (1997). CHEOPS: Adaptive Hypermedia on the World Wide Web . *Proceedings of the European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS '97)*, 210-219.
- [4] Brusilovsky, P.; Eklund, J.; & Schwarz, E. (1998). Web-based education for all: A tool for developing adaptive courseware. *Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference)* 30 (1-7), 291-300.
- [5] Brailsford, T.J.; Stewart, C.D.; Zakaria, M.R. & Moore, A. (2002). Autonavigation, Links and Narrative in an Adaptive Web-Based Integrated Learning Environment. *Eleventh International World Wide Web Conference*.
- [6] Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction* 6 (2-3), 87-129.
- [7] Marsh, J. & Orchard, D. (eds, 2002). XML Inclusions (XInclude) Version 1.0. W3C Candidate Recommendation.
- [8] Zakaria, M.R; Moore, A; Ashman, A; Stewart, C & Brailsford, T. (2002). The Hybrid Model for Adaptive Educational Hypermedia. *Proceedings of Second International Conference, AH2002*, 580-585.
- [9] Zakaria, M.R. & Brailsford, T.J. (2002). User Modelling and Adaptive Educational Hypermedia Frameworks for Education. *New Review of Hypermedia and Multimedia, (NRHM 2002)* 8,83-97.

Bibliography

ANON (2003). *Dewey Decimal Classification and Relative Index*. 22 edn. ISBN: 0-910608-70-9.

Anderson M, Jackson D (2000). *Computer Systems for distributed and distance learning*. *Journal of Computer Assisted Learning*, 16, pp 213-228.

Atkinson S (2001). *Cognitive Styles and Computer Aided Learning (CAL): Exploring Designer and User Perspectives*. Proceedings of PATT-11 Conference. pp 3-14.

Barker T, Jones S, Britton C, Messer D (2002). *The Use of a Co-operative Student Model of Learner Characteristics to Configure a Multimedia Application*. *User Modeling and User-Adapted Interaction*, 12, pp 207-241.

Benaki E, Karkaletsis V, Spyropoulos C (1997). *User Modeling in WWW: the UMIE Prototype*. Proceedings of the workshop "adaptive systems and user modeling on the world wide web", Sixth International Conference on User Modeling. Chia Laguna, Sardinia. pp 13-22.

Benjamins VR, Fensel D, Pe'rez AG (1998). *Knowledge Management through Ontologies*. Proceedings of Second International Conference on Practical Aspects of Knowledge Management. pp 5.1-5.12

Benyon D, Murray D (1993). *Adaptive Systems: from intelligent tutoring to autonomous agents*. *Knowledge- Based Systems*, 6(4), pp 197-219.

Benyon D, Stone D, Woodroffe M (1997). *Experience with developing multimedia courseware for the World Wide Web: the need for better tools and clear pedagogy*. *International Journal of Human-Computer Studies*, 47, pp 197-218.

Berners-Lee T (1992). *Hypertext Markup Language (HTML)*. <http://www.w3.org/hypertext/WWW/MarkUp/MarkUp.html>.

Berners-Lee T (1998). *Semantic Web Road Map*. <http://www.w3.org/DesignIssues/Semantic.html>.

Bernstein M (1998). *Patterns of Hypertext*. Proceedings of Hypertext '98, pp 21-29

Billsus D, Brunk C, Evans C, Gladish B, Pazzani, M (2002). *Adaptive Interfaces for Ubiquitous web access*. *Communications Of the ACM*, 45(5), pp 34-38.

Bonfigli M, Casadei G, Salomoni P (2000). *Adaptive Intelligent Hypermedia using XML*. Proceedings of ACM Symposium on Applied Computing, SAC 2000. Italy. pp 922-926.

- Bontcheva K (2001). *The Impact of Empirical Studies on the Design of an Adaptive Hypertext Generation System*. Proceedings of the Third workshop on adaptive hypertext and hypermedia, Eight International Conference on User Modeling (UM2001). Sonthofen, Germany. pp 201-204.
- Boyle C, Encarnacion A (1994). *Metadoc: an adaptive hypertext reading System*. User Modeling and User-Adapted Interaction, 4, pp 1-19.
- Brailsford T, Stewart C, Zakaria M, R, Moore A (2002). *Autonavigation, links and narrative in an adaptive web-based integrated learning environment*. Proceedings of World Wide Web, (www 2002). Hawaii, USA.
<http://www2002.org/CDROM/alternate/738/>
- Brailsford T, Moore A, Stewart C, Zakaria M, Choo BS, Davies P (2001). *Towards a framework for effective web-based distributed learning*. Proceedings of World Wide Web 2001, (WWW 2001). Hong Kong. pp 120 – 121.
- Brailsford T, Ashman H, Stewart C, Zakaria M, Moore A (2002). *User Control of Adaptation in an Automated Web-Based Learning Environment*. Proceedings of First International Conference on Information Technology & Applications (ICITA 2002). Bathurst, Australia. pp 252.14-262.14.
- Brajnik G, Tasso C, Vaccher A (1991). *A Flexible Tool for Assumption based user modeling*. Proceedings of Trends in Artificial Intelligence - 2nd Congress of the Italian Association for Artificial Intelligence (AI*IA). pp 445-449.
- Brown I (1998). *The Effect of WWW Document Structure on Students' Information Retrieval*. Journal of Interactive Media in Education, 12.
<http://www-jime.open.ac.uk/98/12/brown-98-12.html>
- Brusilovsky P (1995). *Intelligent Tutoring Systems for the World-Wide Web*. Proceedings of Third International WWW Conference. pp 42-45.
- Brusilovsky P (2000). *Adaptive Hypermedia: from Intelligent Tutoring Systems To Web-based Education*. Proceedings of ITS 2000. Montreal, Canada. pp 1-7.
- Brusilovsky P (2001). *Adaptive Hypermedia*. User Modeling and User-Adapted Interaction, 11, pp 87-110.
- Brusilovsky P (1996). *Methods and techniques of adaptive hypermedia*. User Modeling and User-Adapted Interaction, 6(2-3), pp 87-129.
- Brusilovsky P (1997). *Efficient Techniques for Adaptive hypermedia*. In: Nicholas C, Mayfield J (eds), *Intelligent Hypertext: Advanced techniques for the World Wide Web*. Springer-Verlag, pp 12-30.
- Brusilovsky P (2001a). *Adaptive Educational Hypermedia*. Proceedings of Tenth International PEG conference. Tampere, Finland. pp 8-12.

Brusilovsky P (1994). *Student model centered architecture for intelligent learning environments*. Proceedings of Fourth international conference on User Modeling. Hyannis, MA, USA. pp 31-36.

Brusilovsky P (1998). *Adaptive Educational Systems on the World-Wide-Web: A Review of Available Technology*. Proceedings of the workshop “www-based tutoring”, 4th International Conference on Intelligent Tutoring Systems (ITS'98). San Antonio, TX.

<http://www-aml.cs.umass.edu/~stern/webits/itsworkshop/brusilovsky.html>

Brusilovsky P, Eklund J, Schwarz E (1998). *Web-based education for all: a tool for development adaptive courseware*. Computer Networks and ISDN Systems, 30(1-7), pp 291-300.

Brusilovsky P, Maybury M (2002). *From Adaptive Hypermedia to the Adaptive Web*. Communications Of the ACM, 45(5), pp 31-33.

Brusilovsky P, Pesin L (1998). *Adaptive Navigation Support in Educational Hypermedia: an Evaluation of the ISIS-Tutor*. Journal of Computing and Information Technology, 6(1), pp 27-38.

Brusilovsky P, Ritter S, Schwarz E (1997). *Distributed Intelligent tutoring on the Web*. In: B. du Boulay and R. Mizoguchi (eds), Proceedings of 8th World Conference on Artificial Intelligence in Education (AI-ED'97). Amsterdam. pp 482-489.

Brusilovsky P, Schwarz E, Weber G (1996a). *A Tool for developing adaptive electronic textbooks on WWW*. Proceedings of WebNet'96, World Conference of the Web Society. San Francisco, USA. pp 64-69.

Brusilovsky P, Schwarz E, Weber G (1996b). *ELM-ART: An Intelligent Tutoring System on World Wide Web*. Proceedings of Third International Conference on Intelligent Tutoring Systems (ITS'96). pp 261-269.

Bryman, A, Cramer, D (1999). *Quantitative Data Analysis with SPSS Release 8 for Windows*. Routledge, ISBN: 0-415-20697-9.

Bush V (1945). *As We May Think*. The Atlantic Monthly 176(1), pp101-108.

Cabrera L, Fórtiz MJ, Llorca J (2001). *Hypermedia Systems: the Need for Cognitive Hypermedia Models*. Proceedings of Taller de Evolución del Software en VI Jornadas de Ingeniería del Software y Bases de Datos. Spain. pp 71-87.

Cagle K, Corning D, Dynstee T, Gudmundsson O, Mason M, Pinnock J, Spencer P, Tang J, Watt A, Jirat J, Tchistopolskii P, Tennison J (2001). *Professional XSL*. Wrox Press Ltd, ISBN: 1-861003-57-9.

Calvi L (1997). *Multifunctional (Hyper) Books: A Cognitive Perspective (or the User's Side)*. Proceedings of the workshop "Adaptive systems and user modeling on the World Wide Web", Sixth International Conference on User Modeling. Chia Laguna, Sardinia. pp 23-30.

Calvi L, Cristea A (2002). *Towards Generic Adaptive systems: Analysis of a case Study*. Proceedings of Second international conference on adaptive hypermedia and adaptive web-based systems, AH 2002. Malaga, Spain. pp 79-89.

Carr B, Goldstein I (1977). *Overlays, a theory of modeling for computer aided Instruction*. MIT, Cambridge, MA, Report number: 406.

Carro R, Pulido E, Quentin-Baxter M (2000). *How Adaptivity Affects the Development of TANGOW web-based courses*. Proceedings of International conference on Adaptive Hypermedia and adaptive web-based systems, AH 2000. Trento, Italy. pp 280-283.

Carro R, Pulido E, Rodríguez P (1999). *TANGOW: Task-Based Adaptive Learner Guidance on the WWW*. Proceedings of Second Workshop on Adaptive Systems and User Modeling on the Web. Banff, Canada. Pp 49-57.

Chappel R, Wilson D, Cahour B (1992). *Engineering User Models to Enhance Multimodal Dialogue*. Proceedings of Engineering for Human-Computer Interaction. Amsterdam. pp 297-313.

Chen S, Ford N (1997). *Towards adaptive information systems: individual differences and hypermedia*. Information Research, 3(2).
<http://informationr.net/ir/3-2/paper37.html>

Chris W (1998). *Towards an Ontology for Library Modalities*. Proceedings of AAAI-98 Workshop on Representations for Multi-Modal Human-Computer Interaction.
<http://www.cs.vassar.edu/faculty/welty/papers/multi-modal98.pdf>

Conklin J (1987). *Hypertext: An Introduction and Survey*. IEEE Computer, 20(9), pp 17-41.

Curtin J (2002). *WebCT and online tutorials: New possibilities for student interaction*. Australian Journal of Educational Technology, 18(1), pp 110-126.

Da Silva D, Durm R, Duval E, Olivie' H (1998). *Concepts and documents for adaptive educational hypermedia: a model and a prototype*. Proceedings of 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98. Pittsburgh, USA. <http://wwwis.win.tue.nl/ah98/Pilar/Pilar.html>

Davies P (1994). *The Scholar's Desktop: A Specification Briefing (version 2)*. TLTP Biodiversity Consortium

De Bra P (2002). *Adaptive Educational Hypermedia on the WEB*. Communications Of the ACM, 45(5), pp 60-61.

De Bra P (1996). *Teaching Hypertext and Hypermedia through the web*. Proceedings of WebNet'96 conference. San Francisco, USA. pp 130-135.

De Bra P, Aerts A, Houben G, Wu H (2000). *Making General-Purpose Adaptive Hypermedia Work*. Proceedings of AACE WebNet Conference. San Antonio, Texas, USA. pp 117-123.

De Bra P, Aerts A, Smits D, Stash N (2002a). *AHA! Meets AHAM*. Proceedings of Second international conference, AH 2002. Malaga, Spain. pp 388-391.

De Bra P, Aerts A, Smits D, Stash N (2002b). *AHA! Version 2.0, More Adaptation Flexibility for Authors*. Proceedings of AACE ELearn'2002 conference. pp 240-246.

De Bra P, Aerts A, Smits D, Stash N (2002c). *AHA! The Next Generation*. Proceedings of ACM Conference on Hypertext and Hypermedia, HT'02. pp 21-22.

De Bra P, Brusilovsky P, Houben G (1999). *Adaptive Hypermedia: From Systems to Frameworks*. ACM Computing Surveys, 31(4).

De Bra P, Calvi L (1998a). *AHA! An open Adaptive Hypermedia Architecture*. New Review of Hypertext and Multimedia, 4, pp 115-139.

De Bra P, Calvi L (1998b). *AHA: a Generic Adaptive Hypermedia System*. Proceedings of 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98. Pittsburgh, USA. pp 5-12.

De Bra P, Calvi L (1997). *Creating Adaptive Hyperdocuments for and on the Web*. Proceedings of AACE WebNet'97 Conference. Toronto, Canada. pp 149-154.

De Bra P, Houben G, Wu H (1999a). *AHAM: A Dexter-based Reference Model for adaptive Hypermedia*. Proceedings of the ACM Conference on Hypertext and Hypermedia. Darmstadt, Germany. pp 147-156.

De Bra P, Ruiters J (2001). *AHA! Adaptive Hypermedia for All*. Proceedings of Proceedings of the WebNet Conference. pp 262-268.

Dede C (1996). *Distance Learning to Distributed Learning: Making the Transition*. Learning and Leading with Technology, 23(7), pp 25-30.

Delestre N, Pécuchet J, Gréboval C (1999). *Why to use a dynamic adaptive hypermedia for teaching, and how to design it?*. Proceedings of WebNet'99. Honolulu.

Dubios, P (2000). *MySQL*. New Riders Publishing, USA. ISBN: 0-7357-0921-1.

Eklund J, Brusilovsky P (1998). *The Value of Adaptivity in Hypermedia Learning Environments: The Value of Adaptivity in Hypermedia Learning Environments: A Short Review of Empirical Evidence*. Proceedings of Second Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98. Pittsburgh, USA. pp 13-20.

- Eklund J, Brusilovsky P, Schwarz E (1997). *Adaptive Textbooks on the World WideWeb*. Proceedings of The Third Australian Conference on the World Wide Web (AUSWEB'97). Queensland, Australia. pp 186-192.
- Engelbart D (1995). *Toward Augmenting the Human Intellect and Boosting our Collective IQ*. Communications Of the ACM, 38(8), pp 30-33.
- Engelbart D, English WK (1968). *A research center for augmenting human intellect*. Proceedings of Fall Joint Computer Conference. San Francisco, USA. pp 395-410.
- Fensel D, Musen M (2001). *The Semantic Web: A Brain for HumanKind*. IEEE Intelligent systems, 16(2), pp 24-25.
- Ferrandino S, Negro A, Scarano V (1997). *CHEOPS: Adaptive Hypermedia on the World Wide Web*. Proceedings of European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97). pp 210-219.
- Finin T (1989). *GUMS - A General User Modeling Shell*. In: Kobsa A, Wahlster W (eds), *User Models in Dialog Systems*. Springer-Verlag, New York, pp 411-430.
- Fink J, Kobsa A (2000). *A Review and Analysis of Commercial User Modeling Servers for Personalization on the World Wide Web*. *User Modeling and User-Adapted Interaction*, 10, pp 209-249.
- Fink J, Kobsa A, Nill A (1998). *Adaptable and Adaptive Information Provision for All Users, Including Disabled and Elderly People*. *New Review of Hypermedia and Multimedia*, 4, pp 163-188.
- Fink J, Koenemann J, Noller S, Schwab I (2002). *Putting Personalization into practice*. Communications Of the ACM, 45(5), pp 41-42.
- Gilchrist A (2002). *From Aristotle to the 'Semantic web'*. *Library Association Record*, 104(1). <http://www.la-hq.org.uk/directory/record/r200201/article2.html>
- Gonschorek M, Herzog C (1995). *Using Hypertext for Adaptive Helpsystem in an Intelligent Tutoring System*. Proceedings of 7th World Conference on Artificial Intelligence in Education, AI-ED'95. pp 274-281.
- Grant S, Marshall A, Strivens J (1999). *LUSID: a Web-based student profiling system*. Proceedings of the active web, British HCI Group Day Conference. Staffordshire University, UK.
<http://www.visualize.uk.com/conf/activeweb/proceed/pap16/>
- Gruber T (1993). *A translation approach to portable ontology specifications*. *Knowledge Acquisition*, 5(2), pp 199-220.
- Guarino N (1998). *Formal Ontology and Information Systems*. Proceedings of FOIS'98. Trento, Italy. pp 3-15.

Halasz F, Schwartz M (1990). *The Dexter Hypertext Reference Model*. Proceedings of NIST Hypertext Standardization Workshop. <http://ei.cs.vt.edu/~mm/pdf/dexter.pdf>

Harold, E (1999). *XML Bible*. IDG Books Worldwide, ISBN: 0-7645-3236-7.

Hartly J, Sleeman D (1973). *Towards more intelligent teaching systems*. International Journal of Man-Machine Studies, 2, pp 215-236.

Helic D, Maurer H, Scherbakov N (1999). *Authoring and Maintaining of Educational Applications on the Web*. Proceedings of ED-MEDIA'99. Seattle, USA. pp 1792-1797.

Helic D, Maurer H, Scerbakov N (2001). *Knowledge Domains: A Global Structuring Mechanism for Learning Resources in WBT Systems*. Proceedings of WebNet 2001. Charlottesville, USA. pp 509-514.

Hohl H, Böcker H, Gunzenähuser R (1996). *Hypadapter: An Adaptive Hypertext System form Exploratory Learning and Programming*. User Modeling and User-Adapted Interaction, 6, pp 131-156.

Holt P, Dubs S, Jones M, Greer J (1991). *The State of Student Modelling*. In: Greer J, McCalla G (eds), *Student Modelling: The Key to Individualized Knowledge-Based Instruction*. Springer, pp 3-35.

Hothi J, Hall W (1998). *An Evaluation of Adapted Hypermedia Techniques Using Static User Modelling*. Proceedings of 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98. Pittsburgh, USA. pp 45-50.

Hughes G (1999). *AIMS - Academic Information Management System*. Proceedings of the active web, British HCI Group Day Conference. Staffordshire University, UK. <http://www.visualize.uk.com/conf/activeweb/proceed/pap26/>

Hunter J, Crawford W (1998). *JAVA Servlet Programming*. O'Reilly & Associates, ISBN: 1-56592-391-X.

Joachims T, Freitag D, Mitchell T (1997). *Webwatcher: A tour guide for the World Wide Web*. Proceedings of IJCAI-97 conference, pp 770-775.

Jording T, Michel S (1999). *Personalized Shopping in the Web by Monitoring the Customer*. Proceedings of the active web, British HCI Group Day Conference. Staffordshire University, UK. <http://www.visualize.uk.com/conf/activeweb/proceed/pap6/>

Kampa S, Miles-Board T, Carr L (2001). *Linking with meaning: Ontological hypertext for scholars*. University of Southampton, Southampton, Report number: 0-854327-37-1.

Kass R (1989). *Student Modeling in Intelligent Tutoring systems*. In: Kobsa A, Wahlste W (eds), *User Models in Dialog Systems*. Springer, pp 386-410.

Kass R, Finin T (1988). *A General User Modelling Facility*. Proceedings of SIGCHI conference on Human factors in computing systems. Washington, D.C, USA. pp 145-150.

Kay J (1995). *The UM Toolkit for Cooperative user modelling*. User Modeling and User-Adapted Interaction, 4(3), pp 149-196.

Kay M (2000). *XSLT-Programmer's Reference*. Wross press LTD,

Kinshuk, Patel A (1997). *A Conceptual Framework for Internet based Intelligent Tutoring Systems*. In: Behrooz A (eds), Knowledge Transfer (Volume II). pp 117-124.

Klein M, Methlie L (1995). *Knowledge-based Decision Support Systems with Applications in Business*. second edn. John Wiley & Sons Ltd, England. ISBN: 0-471-95295-8.

Kobsa A (1990). *Modeling the user's conceptual knowledge in BGP-MS, a user modeling shell system*. Computational Intelligence, 6, pp 193-208.

Kobsa A (2001). *Generic User Modeling systems*. User Modeling and User-Adapted Interaction, 11, pp 49-63.

Kobsa A, Koenemann J, Pohl W (2001). *Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships*. The Knowledge Engineering Review, 16(2), pp 111-155.

Kobsa A, Pohl W (1995). *The User Modeling Shell System BGP-MS, a user modeling shell system*. User Modeling and User-Adapted Interaction, 4(2), pp 59-106.

Lashkari Y, Metral M, Maes P (1994). *Collaborative Interface Agents*. Proceedings of the twelfth National Conference on Artificial Intelligence. pp 444-450.

Laurillard, D (1993). *Rethinking University Teaching a Framework for the Effective use of Educational Technology*. Routledge, London. ISBN: 0-415-09288-4.

Machado I, Martins A, Paiva A (1999). *One for All and All in One A Learner Modelling Server in a Multi-Agent Platform*. Proceedings of the Seventh International Conference on User Modeling, (UM'99). pp 211-221.

Maes P (1994). *Agents that Reduce Work and Information Overload*. Communications Of the ACM, 37(7), pp 30-40.

McKnight C, Dillon A, Richardson J (1996). *User-Centered Design of Hypertext/Hypermedia for Education*. In: Jonassen D (eds), Handbook of Research for Educational Communications and Technology. pp 622-633.

Mclaughlin B (2000). *Java and XML*. 1st edn. O'Reilly, ISBN: 0596000162.

MICROSOFT (2001). *MCSE Training Kit: Microsoft Windows 2000 Advanced Server Clustering Services*. Microsoft Press, ISBN: 0735612935.

Moore A, Brailsford T, Stewart C (2001). *Personally tailored teaching in WHURLE using conditional transclusion*. Proceedings of The twelfth ACM conference on hypertext and hypermedia. Denmark. pp 163-164.

Moore A, Stewart C, Zakaria MR, Brailsford T (2003). *WHURLE - an adaptive remote learning framework*. Proceedings of International Conference of Engineering Education. Valencia, Spain. <http://www.etsid.upv.es/icee2003/pdf/5614.pdf>

Murray T (1999). *Authoring Intelligent Tutoring Systems: An Analysis of the State of the art*. International journal of Artificial Intelligence in Education, 10, pp 98-129.

Murray T, Shen T, Piemonte J, Condit C, Thibedeau J (2000a). *Adaptivity in the MetaLinks Hyper-Book Authoring Framework*. Proceedings of the international workshop on Adaptive and Intelligent Web-Based Education Systems, ITS 2000 Canada. Osnabrück: Technical Report of the Institute for Semantic Information Processing. pp. 61-72.

Murray T, Shen T, Piemonte J, Condit, C, Thibedeau J (2000). *Adaptivity For Conceptual and Narrative Flow in Hyperbooks: the Metalinks System*. Proceedings of Adaptive Hypermedia 2000, (AH2000). Trento, Italy. pp 155-166.

Negro A, Vittorio S, Simari R (1998). *User Adaptivity on WWW through CHEOPS*. Proceedings of 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98. Pittsburgh, USA. pp 57-62.

Nelson T (1965). *A File Structure for The Complex, The Changing and the Indeterminate*. Proceedings of ACM 20th National Conference. pp 191-210.

Nelson T (1967). *Getting out of our system*. In: Shecter G (eds), *Information Retrieval: a Critical Review*. Thompson Books, Washington D.C, pp 191-210.

Nelson T (1980). *Replacing the printed word: a complete literary system*. Proceedings of IFIP congress. pp 1013-1023.

Nelson T (1995). *The Heart of Connection: Hypermedia Unified by Transclusion*. Communications of the ACM, 38(8), pp 31-33.

Nelson T (1997). *Transcopyright: Dealing with the dilemma of digital copyright*. Educom Review, 32, pp 32-35.

Nelson T (1999). *Xanalogical Structure, Needed Now More Than Ever: Parallel Documents, Deep Links to Content, Deep Versioning, and Deep Re-Use*. ACM Computing Surveys, 31(4).

http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/60.html

Ohene-Djan J (2000). *A Formal Approach to Personalisable, Adaptive Hyperlink-Based Systems*. PhD Thesis: Mathematical and Computing Science, Goldsmiths College - University of London

Ohlsson S (1987). *Some Principles of Intelligent Tutoring*. *Artificial Intelligent and Education*, 1, pp 203-238.

Ohlsson S (1994). *Constraint-Based Student Modeling*. In: Greer J, McCalla G (eds), *Student Modelling: The Key to Individualized Knowledge-Based Instruction*. Springer, pp 167-189.

Pardi W (1999). *XML in Action Web Technology*. Microsoft Press, ISBN: 0-7356-0562-9.

Patel A, Kinshuk (1997). *Intelligent Tutoring Tools in a Computer-Integrated Learning Environment for Introductory Numeric Disciplines*. *International Journal of Innovations in Education and Training*, 34(3), pp 200-207.

Pearcey M, Pywell D, Tattersall D (1999). *Dynamic web-based Information Management*. Proceedings of the active web, British HCI Group Day Conference. Staffordshire University, UK.
<http://www.visualize.uk.com/conf/activeweb/proceed/pap2/>

Piguet A, Peraya D (2000). *Creating web-integrated learning environments: An analysis of WebCT authoring tools in respect to usability*. *Australian Journal of Educational Technology*, 16(3), pp 302-314.

Pressy S (1926). *A Simple Apparatus Which Gives Tests and Scores - and Teaches*. *School and Society*, 23(586), pp 373-376.

Pressy S (1927). *A Machine for Automatic Teaching of Drill Material*. *School and Society*, 25(645), pp 549-552.

Quentin-Baxter M (1999). *Quantitative Evidence for Differences Between Learners Making Use of Passive Hypermedia Learning Environments*. *ACM Computing Surveys*, 31(4es).
http://www.cs.brown.edu/memex/ACM_HypertextTestbed/papers/52.html

Ragnemalm E (1995). *Student diagnosis in practice; Bridging a gap*. *User Modeling and User-Adapted Interaction*, 5(2), pp 93-116.

Rich E (1989). *Stereotypes and User Modeling*. In: Kobsa A, Wahlster W (eds), *User Models in Dialog Systems*. pp 35-51.

Rich E (1999). *Users are individuals: individualizing user models*. *International Journal of Human-Computer studies*, 51, pp 323-338.

- Rutledge L, Hardman L, Ossenbruggen J, Bulterman C (1999). *Adaptable Hypermedia with Web Standards and Tools*. Proceedings of the active web, British HCI Group Day Conference. Staffordshire University, UK.
<http://www.visualize.uk.com/conf/activeweb/proceed/pap18/>
- Schwab T (1989). *Methoden zur Dialog-und Benutzermodellierung in adaptive Computersystemen*. PhD Thesis: Fakultät Informatik, Universität Stuttgart.
- Shute V, Psotka J (1996). *Intelligent Tutoring Systems: Past, Present, and Future*. In: Jonassen D (eds), *Handbook of Research for Educational Communications and Technology*. First pp 570-600.
- Skinner B (1954). *The science of Learning And the Art of Teaching*. Harvard Educational Review, 24(2), pp 86-97.
- Smith J, Weiss S (1988). *Hypertext: Introduction to the Special Issue*. Communications Of the ACM, 31(7), pp 816-819.
- Smyth B, Bradley K, Rafter R (2002). *Personalization techniques for online recruitment services*. Communications Of the ACM, 45(5), pp 39-40.
- Soloway E, Bielaczyc K (1995). *Interactive Learning Environments: Where They've Come From & Where They're Going*. Proceedings of CHI 95 Conference Companion. pp 347-348.
- Specht M, Kravcik M, Klemke R, Pesin L, Hüttenhain R (2002). *Adaptive Learning Environment for Teaching and Learning in WINDS*. Proceedings of Second international conference, AH 2002. Malaga, Spain. pp 572-575.
- Specht M, Kravcik M, Pesin L, Klemke R (2001). *Authoring Adaptive Educational Hypermedia in WINDS*. Proceedings of ABIS-Workshop 2001. Dortmund, Germany.
http://www.kbs.uni-hannover.de/~henze/ABIS_Workshop2001/final/Specht_final.pdf
- Spiro R, Feltovich P, Jacobson M, Coulson R (1991). *Cognitive Flexibility, Constructivism, and Hypertext: Random Access Instruction for Advanced Knowledge Acquisition in III-Structured Domains*. Educational Technology, 3(51), pp 24-33.
- Tabachnick B, Fidell L (2001). *Using Multivariate Statistics*. 4 edn. A Pearson Education Company, ISBN: 0-321-05677-9.
- Tsinakos A, Margaritis K (2000). *Student Models: the transit to distance education*. European Journal of Open and Distance Learning (EURODL), 11.
- Valley K (1997). *Learning Styles and Courseware Design*. Association of Learning Technology Journal, 5(2), pp 42-51.
- Wahlster W, Kobsa A (1989). *Users Models in Dialog Systems*. In: Kobsa A, Wahlster W (eds), *User Models in Dialog Systems*. pp 5-34.
- Weber G (1996). *Episodic learner modeling*. Cognitive Science, 20, pp 195-236.

Weber G, Kuhl H, Weibelzahl S (2001). *Developing Adaptive Internet Based Courses with the Authoring System NetCoach*. Pre-workshop proceedings of the third workshop on adaptive hypertext and hypermedia, Eight International Conference on User Modeling, (UM2001). Sonthofen, Germany. pp 41-54.

Weber G, Möllenberg A (1994). *ELM-PE: A Knowledge-based Programming Environment for Learning LISP*. Proceedings of World Conference on Educational Multimedia and Hypermedia (ED-MEDIA'94). Vancouver, Canada. pp 557-562.

Weber G, Specht M (1997). *User modeling and adaptive navigation support in www-based tutoring systems*. Proceedings of User Modelling '97. pp 289-300.

Weinstein P (1998). *Ontology-Based Metadata: Transforming the MARC Legacy*. Proceedings of Third ACM International Conference on Digital Libraries. Pittsburgh, USA. pp 254-263.

Wilson BG (1995). *Metaphors for instruction: Why we talk about learning environments*. Educational Technology, 35(2), pp 25-30.

Wu H, Houben G, De Bra P (1999). *User modeling in adaptive hypermedia applications*. Proceedings of the Interdisciplinaire Conferentie Informatiewetenschap. Amsterdam. pp 10-21.

Zakaria M, Brailsford T (2002). *User Modelling, and Adaptive Hypermedia Frameworks for Education*. New Review of Hypermedia and Multimedia, 8, pp 83-98.

Zakaria M, Moore A, Ashman H, Stewart C, Brailsford T (2002). *The Hybrid Model for Adaptive Educational Hypermedia*. Proceedings of Second international conference, AH 2002. Malaga, Spain. pp 580-585.

Zakaria M, Moore A, Stewart C, Brailsford T (2003). *Pluggable user models for adaptive hypermedia in education*. Proceedings of The Fourteenth ACM Conference on Hypertext and Hypermedia (HT'03). Nottingham, UK. pp 170-171.