

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Developing an extramural e-learning environment to bridge the digital divide

A dissertation presented in partial fulfilment of the requirements for the degree of Doctor Of Philosophy in Computer Science at Massey University, Palmerston North, New Zealand.

Russell Johnson

2005

Abstract

The research presented in this thesis conceptualises a strategy for designing e-learning systems to bridge the digital divide between those who have access to – and know how to use – high performance information technology, and those whose do not. It describes the prototyping of a system to test this conceptualisation, and the subsequent evaluation of the prototype in a realistic setting.

From a review of existing research, eight guidelines were synthesised for developing effective extramural e-learning environments. In addition, three broad user-centred strategies were identified as showing promise as possible ways to implement such an environment. These strategies emphasised localised over centralised functionality, specialised over general-purpose tools, and user-initiated adaptability over system-initiated adaptivity. It was hypothesised that by following the design guidelines and combining these three strategies – without making any presumptions about technological platform – a workable way could be found to meet all the requirements for an extramural e-learning environment that offers a significant improvement over correspondence-based courses.

Incremental prototyping was used to evaluate and refine the main elements of the design specification and then to integrate them into an operational system. This prototyping confirmed that the method proposed for developing a computer-based learning environment was workable. The prototype was then installed and tested, first over a LAN, and then over a rural telephone-based communication system where it was tested it with users.

The system performed very favourably under these conditions. The volunteers' response to the learning computer was enthusiastic, contrasting what they could accomplish with it to the difficulties they faced with conventional systems. It was concluded that the user testing gave strong support to the thesis that distributive, specialised and adaptable strategies can be successfully combined to provide a widely-accessible and usable computer-based learning environment.

Acknowledgements

There are a number of people and organisations whose advice, encouragement and support during this research I want to acknowledge.

First and foremost I would like to thank my principal supervisor, Assoc. Prof. Elizabeth Kemp, for the time and effort she has put into keeping me focussed throughout this project, but especially during the final writing up of this thesis. I would also like to warmly acknowledge the contributions of my co-supervisors, Assoc. Prof. Ray Kemp and Peter Blakey. I want to express my appreciation for the considered opinions, the constructive feedback, and, above all, the space to develop my research ideas, that I received from all of them.

There are a number of busy people from a range of disciplines who took time out to share with me their experience and knowledge of computer-based learning during the initial stages of my research. I would particularly like to acknowledge Assoc. Prof. Kinshuk, Bill Anderson and Trevor Billany, in this regard. I also acknowledge the influence of the TILE research group, led by Professor Chris Jesshope, in helping to provide an initial framework for this research.

I want to thank the school and residents of Akitio for their enthusiastic participation in this research, and for providing a real environment in which to test out my ideas.

I acknowledge the financial support of the Foundation for Research, Science and Technology through a Bright Futures Top Achiever Doctoral Scholarship.

On a personal note, I would like to thank Cheryl for her support and encouragement throughout this long process.

I would also like to acknowledge my parents, especially my late father, whose own academic aspirations were cut short by injury, illness and family responsibilities, and for whom I carried the flag on this march.

Publications

Publications associated with this research are:

Johnson R., Kemp E., Kemp R., and Blakey P. (2004). Evaluating Immediate: A Tool for Distance Learning. *Proceedings of International Conference on Computers in Education 2004*. Melbourne Exhibition Centre, November 30th to December 3rd. 2004, Australia. pp. 997-1006.

Johnson R., Kemp E., Kemp R., and Blakey P. (2003). The Virtual Learning Machine: Addressing the Needs of Distance Learners Outside the Information Superhighway. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5. pp. 271-73.

Johnson R., Kemp E., Kemp R., and Blakey P. (2003). The Virtual Learning Machine: Integrating Web and Non-Web Technologies. (V. Devedzic, J. M. Spector, S. D. G, and Kinshuk, eds.) *Proceedings of Third IEEE Conference on Advanced Learning Technologies (ICALT)*. IEEE, Athens, Greece. 9-11 July. pp. 328-329

Johnson, R., Kemp R, Kemp, E., and Blakey, P. (2003). Designing a flexible learning environment: learning from books. *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*. IOS Press. Amsterdam. AIED 2003, Sydney, Australia. July 22-24. pp. 434-436.

Johnson R., Kemp E., Kemp R., and Blakey P. (2002). From electronic textbook to multidimensional learning environment: overcoming the loneliness of the distance learner, Volume 1, *Proceedings of 2002 International Conference on Computers in Education (ICCE 2002)*, Auckland, Dec 3-6, IEEE Press pp. 632 - 636

Brief Table of Contents

Chapter 1	Introduction	1
Chapter 2	Distance education and computer-based learning overview	9
Chapter 3	Web-based e-learning: an evaluation	35
Chapter 4	Conceptualisation of an extramural e-learning system	67
Chapter 5	Towards a specification for an extramural e-learning system	83
Chapter 6	Prototyping the extramural e-learning system	113
Chapter 7	Use and evaluation of IMMEDIATE	175
Chapter 8	Conclusion	211
References		223
Appendix A	References for e-Learning Systems Reviewed	241
Appendix B	Conceptual view of learning elements and study modes	251
Appendix C	User Interfaces to Internet-based systems	257
Appendix D	Learning Shell Requirements Specifications	263
Appendix E	Learning Shell prototyping – classes and components	283
Appendix F	Learning Shell – selected class interfaces and source code	297
Appendix G	Learning Shell – screen shots	351
Appendix H	Communications Management	357
Appendix I	Course authoring and management application	385
Appendix J	Evaluation – handy hints, scenarios for user testing	395
Appendix K	Evaluation: Information sheet, questionnaire, and interviews	405

Detailed Contents

Chapter 1	Introduction	1
1.1	E-Learning for all	1
1.2	An issue for research	3
	1.2.1 Research objectives	4
1.3	Research Methodology	5
1.4	Thesis structure	8
Chapter 2	Distance education and computer-based learning overview	9
2.1	Distance education: an historical review	9
	2.1.1 Four threads in distance learning	10
2.1.2	The technology of distance education	12
	2.1.3 The university and distance education	14
	2.1.4 Summary	17
2.2	Computers in education: Milestones in computer-based learning research	18
	2.2.1 Teaching machines	18
	2.2.2 Thinking machines	21
	2.2.3 Discovery learning	24
	2.2.4 Communities of learning	26
	2.2.5 Summary	28
2.3	Networked computers and distance education	29
	2.3.1 Constraints on use of computers in distance education	31
2.4	Conclusion	33
Chapter 3	Web-based e-learning: an evaluation	35
3.1	Terminology	35
3.2	Overview of recent e-learning research	36
3.3	Evaluation criteria	37
	3.3.1 A focus on extramural study in the public sector	37
	3.3.2 End-user requirements	37
	3.3.3 Functionality	38

3.3.4	Usability	40
3.3.5	Accessibility	41
3.3.6	A student-centred learning environment	42
3.4	Assessment of web-based courseware	42
3.4.1	Courseware functionality	42
3.4.2	Usability concerns	46
3.4.3	Inaccessibility	47
3.4.4	Summary	48
3.5	Recent developments in e-learning	49
3.5.1	Anywhere, anytime study	49
3.5.2	Individualisation	51
3.5.3	Collaboration and Help	55
3.5.4	Interaction	56
3.5.5	Specialisation	57
3.5.6	Conclusion	58
3.6	Designing for extramural e-learning	59
3.6.1	Prioritising the student interface	59
3.6.2	Adaptation in a learning environment approach:	61
3.6.3	Guidelines for designing extramural e-learning environments	62
3.7	Concluding the literature review – a hypothesis	63
Chapter 4	Conceptualisation of an extramural e-learning system	67
4.1	A computer for learning	67
4.1.1	Everyday activities imbuing the conceptual model	68
4.1.2	Making learning possible	69
4.1.3	A reusable educational resource	70
4.2	Specialised interface	70
4.2.1	E-learning metaphors	70
4.2.2	A modular approach	72
4.2.3	Rendering the interface invisible	72
4.3	Individualised interface	74
4.3.1	Adaptation strategy	74
4.3.2	Integrated Support, Communication and Collaboration	75
4.4	Integrated network user interface	77
4.4.1	Network characteristics	78
4.4.2	A networked extramural e-learning system	79

4.5	Conclusion	80
Chapter 5 Towards a specification for an extramural e-learning system		83
5.1	Network User Interface	84
	5.1.1 Alternative network user interfaces	84
	5.1.2 Integrated Interface	89
5.2	Network Architecture	90
	5.2.1 Architectural style	91
	5.2.2 Communication Style.	92
	5.2.3 Connection style	93
5.3	Development platform	94
	5.3.1 Multi-platform	95
	5.3.2 PC-based	95
	5.3.3 Windows-based	96
5.4	A Specialised User Shell	97
5.5	Modular construction	100
	5.5.1 System Components	102
5.6	Adaptable environment	103
	5.6.1 Student Model	104
	5.6.2 Integrated Learning Support	104
	5.6.3 Individualised Support	105
5.7	IMMEDIATE specification	109
5.8	Summary- Prototype Focus	110
Chapter 6 Prototyping the extramural e-learning system		113
6.1	Functional requirements for student user	114
6.2	Component-based approach	115
	6.2.1 Study modes	115
	6.2.2 Learning elements	117
	6.2.3 Delphi code modules	117
	6.2.4 Reusable learning components	119
	6.2.5 Learning Shell assembly	120
	6.2.6 Refinements to promote consistency and modularity	123
	6.2.7 Feasibility of component-based approach	124
6.3	System level support	124

6.3.1	System Tree	125
6.3.2	System level operations	125
6.3.3	Reference model specification	127
6.3.4	Implementation of data structures	127
6.3.5	Refinements to system modules	129
6.3.6	Embedded operating system	130
6.4	Interface Design	131
6.4.1	Learning Shell look and feel	132
6.4.2	Interface Refinement	136
6.5	Integrated learning support and communications	141
6.5.1	The system database	142
6.5.2	Communications and Group Work	143
6.5.3	Extramural Support	145
6.5.4	Coarse-grained learning help	148
6.5.5	Individualisation	149
6.6	Communications management	150
6.6.1	Repository Manager	152
6.6.2	Asynchronous messaging	153
6.6.3	Procedures for Resource Updating over the network	160
6.6.4	A challenging task	162
6.7	Authoring application	163
6.7.1	Course Authoring and Re-use	163
6.7.2	Authoring mechanism	164
6.7.3	Implementation	165
6.7.4	Evaluation of authoring tool	167
6.8	The IMMEDIATE prototype	168
6.8.1	Modifications to the prototype	170
6.8.2	Debugging	171
6.9	Conclusion	172
Chapter 7 Use and evaluation of IMMEDIATE		175
7.1	Goals of evaluation phase	175
7.2	The Evaluation Strategy	176
7.2.1	Focus on student as user	176
7.2.2	Testing for functionality and accessibility	177
7.2.3	Usability testing	177

7.2.4	Field testing	178
7.2.5	Purposive sampling	179
7.3	Installation and evaluation	179
7.3.1	Test environment	179
7.3.2	Installation of the system	180
7.3.3	Pilot study	182
7.3.4	Organising the field test	185
7.4	Evaluation Results	195
7.4.1	Functionality and accessibility	195
7.4.2	Usability	199
7.5	Conclusion	209
 Chapter 8 Conclusion		211
8.1	Summary of research	211
8.1.1	Literature review	212
8.1.2	Conceptualisation and specification of a computer for learning	213
8.1.3	Evaluation through prototyping and user testing	215
8.2	Contribution to knowledge	216
8.3	Future Work	220
 References		223
 Appendix A References for e-Learning Systems Reviewed		241
A.1	Paper References for Initial System Comparison	242
A.2	Web References for Initial System Comparison	247
 Appendix B Conceptual view of learning elements and study modes		251
 Appendix C User Interfaces to Internet-based systems		257
 Appendix D Learning Shell Requirements Specifications		263
D1	Distance learner scenarios	264
D2	Use Cases	273

D3	Sequence Diagrams	278
D4	Extramural Support: Query Specification	279
D5	Extramural Support: Dialogue Specification	281

Appendix E Learning Shell prototyping – classes and components 283

E1	Prototyping basic component types	284
E2	Component dependencies	285
E3	Inheritance hierarchy	287
E4	Class Hierarchy	288
E5	Delphi Object Hierarchy	294

Appendix F Learning Shell – selected class interfaces and source code 297

F1	Controller object – interface and selected source code	298
F2	Course Explorer object – class interface, selected source code	306
F3	System Dictionary – constants	311
F4	System Utilities – class interface	315
F5	Resource Model Manager – interface and selected source code	316
F6	Student Model Manager – class interface	317
F7	System Model Manager – class interface and selected code	318
F8	System Tree Manager – class interface	320
F9	Sample reference model implementation – System Tree	321
F10	Sample learning component implementation	325
F11	Extramural Support – class interface and selected source code	327
F12	Concept Map – class interface and selected source code	338
F13	System Models – Models directory and selected model files	344

Appendix G Learning Shell – screen shots 351

Appendix H Communications Management 357

H1	FTP Server – Screen shots	358
H2	Repository Manager – Screen shots	360
H3	Repository Manager – interface and selected source code	362
H4	Learning Shell: Update Resources – interface and selected code	374
H5	Learning Shell: Update Extramural Support – interface, selected code	379

H6	Learning Shell: Messaging – interface	382
Appendix I Course authoring and management application		385
I1	Authoring application – screen shots	386
I2	Add Learning Material – screen shot and class interface	393
Appendix J Evaluation – handy hints, scenarios for user testing		395
J1	Handy Hints sheet	396
J2	Initialisation (Scenario 1)	397
J3	Start-up, browse the Study guide (Scenario 2)	399
J4	Work on an Assignment (Scenario 3)	400
J5	Access a Lecture, Ask for Learning Support (Scenario 4)	401
J6	Monitor group discussion, get assignment feedback (Scenario 5)	402
J7	Exploring a Topic (Scenario 6)	403
J8	Completing an interactive tutorial (Scenario 7)	404
Appendix K Evaluation: Information sheet, questionnaire, and interviews		405
K1	Information sheet for participating volunteers	406
K2	Questionnaire for participants	408
K3	Outline for semi-structured interviews with each participant	409
K4	Interviews with participants	411

Figures and Tables

Figures

Figure 1.1	Research methodology	5
Figure 2.1	Innovations in distance learning technology.	12
Figure 3.1	Courseware environment marked by visual clutter.	47
Figure 4.1	Each study mode contains a unique set of learning elements	73
Figure 4.2	Learning computer shifts weight of system to learner	78
Figure: 4.3	Networked extramural e-learning – conceptual view	79
Figure 5.1	Three basic types of web documents.	85
Figure 5.2	Enhanced web browser components.	86
Figure 5.3	Integrated Interface	89
Figure 5.4	IMMEDIATE accommodates diverse communication media.	91
Figure 5.5	Gartner Spectrum of Network Styles.	92
Figure 5.6	Modes implemented from pre-installed components.	101
Figure 5.7	Learning Shell System components.	102
Figure 5.8	System Tree models table of contents and student's progress.	103
Figure 5.9	Knowledge-based system components.	108
Figure 5.10	IMMEDIATE network components and configuration.	110
Figure 6.1	Start-up scenario	116
Figure 6.2	Interface components of the Learning Shell	118
Figure 6.3	Message List component encapsulates multiple forms	119
Figure 6.4	Code to wrap Feedback learning element as Delphi component.	121
Figure 6.5	Shell assembled by dragging components onto Desktop form.	121
Figure 6.6	Desktop encapsulates all Shell components	122
Figure 6.7	System Tree with additional concept level.	125

Figure 6.8	Sequence diagram for Change Mode Use Case.	126
Figure 6.9	Reference models stored as text files in Models directory.	128
Figure 6.10	Learning Shell directory structure.	129
Figure 6.11	Content of "s01t02le.txt" in Resources directory.	129
Figure 6.12	Component <i>show</i> method gets data filepath from Controller.	130
Figure 6.13	Learning Shell architecture.	131
Figure 6.14	Nielsen's usability heuristics (Nielsen, 1994).	132
Figure 6.15	Desktop with Course Explorer open.	134
Figure 6.16	Help screen for Lecture component.	135
Figure 6.17	Mode components are colour-coded.	137
Figure 6.18	Separate learning components may be docked together.	140
Figure 6.19	The integrated System Database model.	143
Figure 6.20	Group Work component with Desktop Menu open.	144
Figure 6.21	Top level methods for interactive dialogue with user.	146
Figure 6.22	Concept Map for "conceptual frameworks".	147
Figure 6.23	Repository folder structure.	151
Figure 6.24	The Repository Manager.	153
Figure 6.25	Repository Manager architecture.	153
Figure 6.26	Messaging Protocols.	156
Figure 6.27	(a) Two-step transmission protocol.	157
Figure 6.27	(b) Transmission protocols accommodate overlapping users.	157
Figure 6.28	SQL queries for composing message lists to be transmitted.	159
Figure 6.29	Learning Shell update resources screen.	161
Figure 6.30	Interface for adding and updating learning materials.	166
Figure 6.31	SQL query rewritten to run outside Delphi environment.	170
Figure 6.32	IMMEDIATE prototype implementation.	171
Figure 7.1	"Attend Lectures" scenario.	186
Figure 7.2	Logging on to the Learning Shell.	187

Figure 7.3	User Options form. "Select new topic" selected.	187
Figure 7.4	Course Explorer. Clicking Help icon opens Help screen.	188
Figure 7.5	Navigating to Lectures Mode in Topic 1.2.	188
Figure 7.5	Help screen for Lecture component.	189
Figure 7.5	Right-clicking background with mouse opens Desktop Menu.	189
Figure 7.6	Key Ideas opened via Desktop Menu.	190
Figure 7.7	Asking the tutor (and my group) for help with "metaphor".	190
Figure 7.8	Selecting "conceptual models" in Concept Map.	191
Figure 7.9	After Extramural Support has explained "conceptual models".	191
Figure 7.10	Starting Self-Assessment.	192
Figure 7.11	Self Assessment questionnaire component.	192
Figure 7.12	Key Ideas, Course Explorer are updated after Self-Assessment.	193
Figure 7.13	Exit after updating messages.	193
Figure 7.14	User interactions traced through log file (e.g. from Scenario 4).	196
Figure 7.15	Comparison of scenario completion times.	197
Figure 7.16	Comparison of help access frequency.	197
Figure 7.17	Extract from Repository Manager log.	199
Figure 7.18	Extract from FTP Server log.	200

Tables

Table 3.1	Contrasting properties of adaptable and adaptive systems.	52
Table 3.2	Forms of adaptation in learning systems.	52
Table 4.1	Individualisation dimensions supported by the learning computer	75
Table 4.2	Requirements for network components	80
Table 5.1	Some higher-level protocols implemented over TCP/IP.	93
Table 5.2	Possible methods for rendering the interface invisible.	98
Table 5.3	The four dimensions of the Integrated Help system.	107

Table 6.1	Basic learning element categories	118
Table 6.2	System level operations defined from use cases.	126
Table 7.1	Summary of Participant Profile Questionnaires.	183
Table 7.2	Scenarios covered all e-learning enhancements and dimensions.	185

Chapter 1

Introduction

1.1 E-Learning for all

One of the more remote students of the NZ Correspondence School in recent years lives in Nepal. To return his completed lessons for assessment and pick up his new ones, this student must trek for two weeks through the Himalayan Mountains.¹ His determination to overcome such obstacles in order to advance his education, exemplifies the aspirations of millions in the underdeveloped countries to find a road out of poverty through education. In fact, at least since the industrial and democratic revolutions of the late 18th Century, "Education for all!" has been a rallying call of movements for social progress, and is enshrined in the UN Charter for Human Rights.

The role of the NZ Correspondence School in helping this Nepalese student to achieve his learning goals highlights how learning institutions in the more developed countries can assist in raising educational levels world-wide through distance learning programmes. Through such programmes, the greater educational resources of the developed countries can be made available to poorer countries to help expand the numbers of teachers, agronomists and others with key skills for economic and social development. Distance education has become a major part of development initiatives in the Third World (Cook, 1998).

Distance education also plays a major role in narrowing the gap in educational opportunities between town and countryside in all countries, and in facilitating study by those unable to attend conventional learning institutions by reason of geographic location, job, disability or age. Others are attracted to distance forms of learning because of the greater flexibility they offer in terms of acquiring new skills and knowledge throughout one's lifetime. In fact, distance learning today encompasses fields as diverse as public education from primary through to tertiary level, government-funded and in-house job skills training, and cultural self-awareness programmes.

¹ Recounted in personal interview with NZ Correspondence School teacher, October 2003.

Demand for the provision of distance education at the university level remains strong. In December 2003, it was reported that Massey University, New Zealand's major distance university education provider, had signed up record numbers of students. In that year, "Massey had extramural² students sitting exams as far afield as Istanbul, Saudi Arabia and Brazil. They also had distance learning students studying in prison, on board naval ships and on overseas army postings" (Allen, 2003).

In recent years, a major shift in distance learning has been underway, with computer-based courses delivered over the Worldwide Web increasingly replacing traditional correspondence courses. With rapid advances in computer and communication technology making possible real-time video-conferencing and "virtual" classrooms (Tiffin, 2002), some even question the future of "bricks-and-mortar" learning institutions. In fact, what has become known as e-learning is more and more promoted as the future of education. According to this view, the communication potential of internet-linked computers is turning the world into a global village (Cisco, 2000; Drucker, 2000). However, if one steps back and looks at the world as it really is today, then a more sobering perspective emerges.

The increased reliance on technology in education is widening the gap between the haves and have-nots. According to an Auckland communication studies lecturer, online information services such as e-learning are focussed upon 150 to 200 "wired cities", while ninety-seven percent of the world's population have no Internet access and almost two-thirds of households have no telephone, let alone the broadband delivery and multimedia computer required for web-based learning (Hope, 2002).

Nor is this just a developed/underdeveloped country hiatus. Microsoft has acknowledged that the digital divide between people who have access to and know how to use technology and those who do not, is a growing problem in developed and developing countries alike. In 2004, the software giant announced that it will spend tens of millions of dollars on community-based technology learning centres around the world (Herman, 2004).

There is tremendous unevenness in social and economic development within developed countries themselves, including between the large urban centres and the

² Meaning study "outside the walls" of the university. "Extramural study" is used in New Zealand to refer to correspondence-based, university-level, distance learning and its computer-supported derivatives. The term is used in the thesis wherever it is necessary to differentiate this particular form of university distance education from university distance learning centres or from the more general domains of "distance learning" and "e-learning".

more remote and sparsely-populated rural districts. In New Zealand, for example, an OECD developed country centred on farming, forestry and fishing, an economically strategic minority of the population lives in relatively-isolated rural communities, where poor communications infrastructure remains a major issue impeding Internet use (Searle, 2001)³.

Distance learning was begun to provide schooling for people living in areas remote from normal educational opportunities. Paradoxically, internet-based delivery of distance education creates a 2-tier system, determined by access to high performance technology, in which those most in need of it can least avail themselves of its benefits.

Poor usability limits learning

At first glance, the simplest way around the problem of poor telecommunications infrastructure is to make use of the postal system. Distance students without fast, reliable Internet access could be mailed a removable storage disk containing their lessons. However, this solution has the disadvantage of maintaining a 2-tier system by sacrificing the communication strengths and speed of the Internet for some.

Moreover, whether the lessons are delivered by mail or the Internet, it is assumed that the remote student will be able to install and effectively use the learning material unaided. Anecdotal and other evidence suggest that this often may not be the case.

It is widely recognised that software systems tend to be more complex than they need to be (Hurley, 1998; Cooper et al., 1999), and to be biased towards the priorities of the developers rather than the users (Gentner et al., 1990). Educational software is not exempt from this trend (Bork, 2001a; Murray et al., 2000). In fact, it poses an additional problem in e-learning because most of the student's effort may be diverted into learning how to use the system rather than learning the content of the course (Smulders, 2003). Poor usability can severely limit and curtail learning.

1.2 An issue for research

This situation poses the question: Is there a universally-applicable way to apply

³ By way of illustrating the scale of the urban/rural digital divide: In September, 2004, an attempt to download the 75 MB Service Pack 2 upgrade of the Windows XP operating system from the Microsoft web site, via a rural modem, was abandoned after the system calculated that, at the actual data transfer rate, it would require 22 hours and 11 minutes to complete. A 240MB developer's version of the upgrade was downloaded from the Microsoft site via Massey University's fast Internet connection in less than 8 minutes!

information technology to university-level distance education, so that it offers all extramural students an advantage over traditional correspondence lessons?

In such a solution, technology would be applied to distance learning without widening the digital divide. Firstly, it would work as well where infrastructure is poor as where it is well-developed; in the countryside as well as the large urban centres. Singapore and India, for example, are two countries which play a prominent part in the modern IT industry. E-learning technologies developed for a relatively-prosperous city-state like Singapore – where a developer can assume widespread access to modern computers and broadband communications – will most likely prove unworkable in India, where eighty percent of the population still live in impoverished rural villages. The converse, however, is probably not true. An e-learning technology that will work across India should also work in Singapore.

Secondly, the e-learning environment would be as usable by a student studying alone in an isolated locality, as one studying under skilled supervision in a learning centre. One that has been designed and tested in a regular campus environment may prove too complex for the distance student, who may be a relative computer novice and have no-one more experienced to turn to for help. On the other hand, a system that has been designed for and tested with extramural students in remote localities, should be usable anywhere.

Therefore, the question being posed here is an important one that merits further research.

1.2.1 Research objectives

The objectives for this research are:

- To identify the ways in which computer systems can assist distance learning and to what extent this potential has been realised in existing e-learning technologies.
- To identify threads of current research relevant to overcoming the shortcomings of e-learning systems for extramural study, drawing from the broad spectrum of Computer Science including Artificial Intelligence, Software Engineering and Human Computer Interaction.
- To develop new ideas and extend existing ones for fulfilling the potential for computers to effectively support distance learning at the university level.
- To build and test a prototype which implements these key ideas.

1.3 Research Methodology

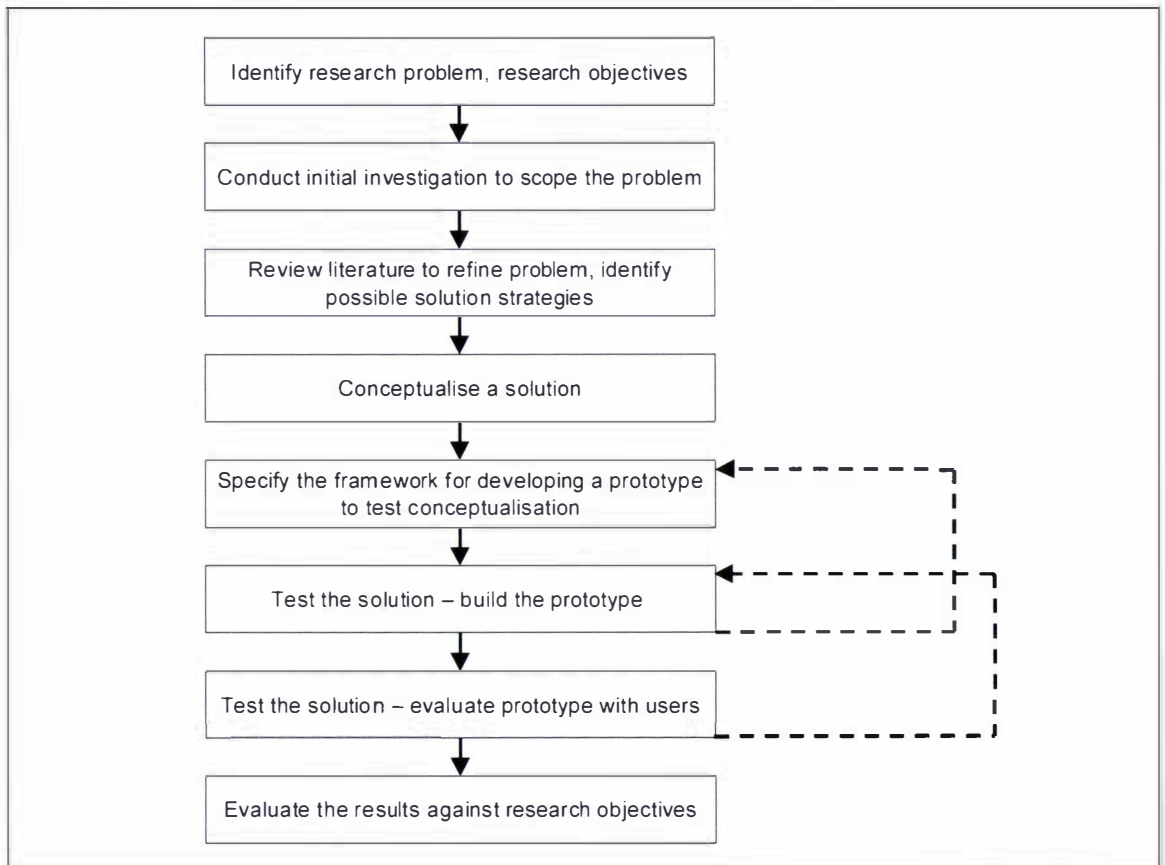


Figure 1.1: Research methodology

In this section, the methodology used in this research is summarised. While the eight stages of this methodology are logically separate (Figure 1.1), the actual process is strongly iterative, especially during the later testing phases where prototyping and evaluation experiences motivate a fresh look at higher level design issues and concepts.

Identify the research problem

The research problem identified is: Is there a universally-applicable way to apply information technology to distance education, so that it offers the student an advantage over traditional correspondence lessons? To be universal, such a method should narrow the digital divide, working as well where infrastructure is poor as where it is well-developed. The motivation for addressing this question is outlined in the previous sections.

Initial investigation

In order to gain a better understanding of the scope and scale of the problem domain

and set a framework for the subsequent phases of the research, especially the literature review, informal interviews and discussions have been conducted with experts, practitioners and users from a range of related disciplines. These include:

- A number of teachers with experience in preparing and coordinating distance learning courses at the university and pre-university levels;
- Several former and current distance learning students;
- An educationalist with research and practical experience in applying technology to learning;
- A professional instructor in the authoring of web-based university courses;
- Two researchers in the application of Artificial Intelligence to computer-based education;
- Two researchers in usability and Human-Computer Interaction.

In addition, some of the available computer-based educational systems, or their associated web sites, have been explored to get a “hands-on” feel for them.

Literature review

The overall objectives of the literature review phase of this research are to further refine the problem under investigation and hypothesise a solution. The review proceeds in two distinct steps.

In the first step, the two major strands which are brought together in computer-based distance learning are examined:

- distance education in its various forms; and
- the educational applications of computers.

The purpose of this historical review is to identify the potential benefits (and downsides) of computer-based distance learning systems, to determine what limits need to be placed on the scope of the domain for this research project and, within that scope, to establish a set of criteria for evaluating e-learning systems.

The second step in the literature review is to evaluate the existing technology, especially the web-based course authoring tools or learning management systems which are widely used for online education and training. These are evaluated against the criteria developed in the previous step, and their major shortcomings listed.

Then research and development addressing issues posed by these shortcomings are

reviewed. The principal objectives for this step are:

- to identify some promising strategies for meeting the criteria for effective computer-based distance learning systems; and
- to synthesise some guidelines for designing and building such systems.

Finally, on this basis, a possible solution to the refined problem is hypothesised.

Conceptualisation

To explore the solution that has been hypothesised a working prototype is needed. The objective of this step in the research methodology is to conceptualise an e-learning system that embodies the main principles and ideas to be evaluated. Conceptualising the prototype also involves drawing analogies from education and other everyday activities with which to imbue the design, as well as addressing some high level information system analysis and design considerations.

Specification framework

Once the solution has been conceptualised at a high level, and before a system can be built, decisions need to be made about what technologies will be used to implement the prototype. This means investigating the major platform and design options in relation to issues such as network architectures and interfaces, operating environments and development platforms, and database structure and search strategies. The objectives are to identify the key elements of the conceptualisation that need to be evaluated through prototyping, and to determine the technological framework for the more detailed specification of the system that will emerge from the prototyping process itself.

Testing through prototyping

Incremental prototyping (Pressman, 1997, pp. 285-288) is used to further explore and refine the key elements of the design, which are then codified in a series of documents. The approach taken is to develop and evaluate each element individually, through a process of repeated iterations. Then the constituent elements are progressively integrated into an overall working system, which in turn is evaluated under laboratory conditions. The primary objective of this stage is to test the technical feasibility of the ideas under consideration, and make possible the subsequent evaluation with users.

Testing through evaluation with users

In this phase, the prototype is installed and field-tested with volunteer users under

realistic conditions. The objectives are

- To demonstrate that the system is fully functional under such conditions;
- To evaluate the system for accessibility and usability.

The usability experiment is prepared by a pilot study. The principal form of data collection is semi-structured interviews with each of the participants. Observation and log files are also used. The results are then collated and analysed.

Evaluate hypothesis in light of the testing phases

The lessons from the prototyping and evaluation phases are discussed. To what extent the research objectives have been met and the research problem solved is assessed in light of the results.

1.4 Thesis structure

The structure of the thesis closely follows the steps in the research methodology.

Chapter 2 and 3 present the results of the literature review. Chapter 2 outlines the historical background to distance learning and computer-based education. Chapter 3 summarises current research in the field, and the conclusions that have been drawn from it regarding a solution to the research problem.

In Chapter 4 the conceptualisation of a universal computer-based distance learning system is presented. Chapter 5 specifies the technological basis on which the conceptualisation is prototyped.

Chapter 6 summarises the extensive prototyping process, with particular attention to the novel aspects of the system. Chapter 7 reports on the work needed to prepare the field evaluation of the prototype, and the results of the usability experiment.

Finally, Chapter 8 concludes the thesis by summarising what has been achieved during the project, looks at some of its broader implications, and outlines some proposals for future work.

Chapter 2

Distance education and computer-based learning: an overview

Research into computer-based distance education occurs at the intersection of two broader fields of learning research and practice: distance education in its various forms, and the educational applications of computer systems. In this chapter, the results of an extensive literature review of both these areas are presented. The major steps in the evolution of each are summarised, with special attention to their role and place in education at the university level. On the basis of the literature review, some conclusions on the potential for computers to enhance distance education are drawn.

2.1 *Distance education: an historical review*

The Oxford Dictionary defines learning as acquiring knowledge of something “by study, experience or being taught” (Fowler et al., 1974), where study is “devotion of time and thought to acquiring information especially from books” (ibid.). All learning occurs in a social context (Wenger, 1998, p3.) and is therefore never a completely isolated activity.

Education, in the sense of an organised system of learning, involves guidance (Alessi et al., 1991, p. 6) Laurillard (1993) defines teaching at the university level as “mediated learning, allowing students to acquire knowledge of someone else’s way of experiencing the world” (p. 29). In the classical classroom situation the emphasis of learning is on *being taught*, that is, on learning activities being directly guided by a human teacher. In distance education, the emphasis is on learning *by study*, especially from books. Nevertheless it is still guided by a human teacher, albeit indirectly, via some form of paper-based, or electronic, lessons delivered to the student by a means such as mail, broadcast media, or the Internet.

Distance learning, therefore, is a social relationship between humans which is mediated by technology. In computer-based distance learning, computers and communication networks provide the facilitating medium.

2.1.1 Four threads in distance learning

Modern distance learning can be traced back to a growing demand for popular education spurred by the industrial and democratic revolutions that gathered steam across Europe and North America in the late 18th and early 19th Centuries. The London Corresponding Society, founded in the 1790s, used the early mail system to disseminate the ideas of Tom Paine (Higgins, 1998). “The pioneers of distance education used the best technology of their day, the postal system, to open educational opportunities to people who wanted to learn but were not able to attend conventional schools” (CDLP, 2004, p. 4). One of these early pioneers was Isaac Pitman who began teaching shorthand by the penny post in England as early as 1840, and later in the United States (PBS, 2004).

Since that time, a diverse range of institutions and motivations have been involved in the delivery of various forms of distance teaching to an equally diverse range of students. Within this diversity, at least four major historical threads can be identified:

- correspondence schooling of isolated rural children, usually provided by the state;
- vocationally-oriented training, often provided on a for-profit basis by private organisations;
- public lecture series presented by touring university academics; and
- home-based, self-improvement study, initially oriented towards women, and often provided by voluntary organisations.

In North America and Australasia, distance learning became institutionalised through the correspondence primary schooling of children living in remote rural areas beyond the reach of itinerant teachers (Higgins, 1998). The New Zealand Correspondence School was established in New Zealand in 1922 with one teacher and 100 pupils (NZSC, 2004). In the United States, correspondence courses began in the late 19th Century and were institutionalised as early as the 1930s (Aniebonam, 2000).

The success of the correspondence system in the United States rapidly spawned private organisations which marketed “job-skills” distance training packages on a for-profit basis. According to Noble (1999), such businesses proliferated during the early 20th century. He writes that by “1924 these commercial enterprises, which catered primarily to people who sought qualifications for job advancement in business and industry, boasted of an enrolment four times that of all colleges, universities, and professional schools combined” (para. 12). The best-known example of these distance

education businesses is the International Correspondence Schools founded in Pennsylvania in 1891 to train mineworkers, which now has millions of students world-wide (PBS, 2004).

As early as 1920 the United States Department of Defense (DoD) began utilising distance education technology in the training of its personnel (ibid.). In the ensuing decades it has been a major driver in the development of this technology, especially of the computer-based training systems now widely adopted in the commercial world. In 1997, it launched the Advanced Distributed Learning initiative to "develop a DoD-wide strategy for using learning and information technologies to modernise education and training and to promote co-operation between government, academia and business to develop e-learning standardisation" (Dodds, 2003, para. 1).

Uglove (2002) and Clark (1999a) cite examples of noted scientists and other academics during the 18th and 19th Centuries touring Britain delivering public lectures. These lectures presaged university extension courses – in which short educational programmes are offered to the general public – and distance learning centres – in which students study together in a classroom remote from the university and supported by visiting lecturers, tutors and, more recently, satellite video, telephone and internet hook-ups.

Learning centres, including satellite campuses, are an important component of distance education in Japan, the United States and many other countries, and are a focal point for e-learning research as evidenced by Hayashi et al. (2001) and Tiffin (2002). In developing countries like India, they are a vehicle for providing distance students with access to technology that would not otherwise be available to them (Cook, 1998, p. 20).

Learning centres are an important aspect of the Open University of the United Kingdom, founded in 1969 to provide mature students with the opportunity for tertiary study to the degree level (Norton, 1994). "The British Open University has broken traditional barriers to education by allowing any student to enrol regardless of previous educational background or experience. It currently has more than 200,000 students and has enrolled more than 2 million people. It is recognised throughout the world as a prototype for current day non-traditional learning" (PBS, 2004, p. 3).

The Open University draws upon the tradition of self-improvement home-based study which began in the 19th century in part to provide educational opportunities to women barred from attending male-only universities. An example is the volunteer-run Society to Encourage Studies at Home in Boston, Massachusetts (PBS, 2004). Today, home-

based study provides a vehicle for more general "second-chance" adult education. In New Zealand, the Technical Correspondence School was founded in 1946 to provide resettlement training for returned servicemen and women following World War II. It expanded in 1963 into national apprenticeship training and now, as the Open Polytechnic, offers many open-admission courses (TOPNZ, 2004). Massey University, which has offered degree and diploma courses by correspondence since the 1960s, has about 21,000 extramural students studying by mail, phone, e-mail or the Internet each year (Massey Extramural, 2004).

By the 1990s, almost three million tertiary students were enrolled world-wide in the eleven largest public, distance teaching universities (Higgins, 1998). In New Zealand, up until the late 1980s distance learning was dominated by four large, public institutions. These institutions accounted for well over 90% of all distance education enrolments and provided New Zealand with a comprehensive selection of school-level, polytechnic, teacher education and university programmes (Prebble, 2001a). Today, however, almost all universities and polytechnics offer some form of distance study as an adjunct to their internal courses – in part to meet a demand for more flexible forms of learning – although on a lesser scale than that offered by Massey University and the Open Polytechnic.

2.1.2 The technology of distance education

Until the 1920s, distance education was almost exclusively based upon the postal system. While book-based lessons delivered by mail remain a core component of distance learning, technological innovations have been progressively incorporated (Figure 2.1).

1840s	Correspondence lessons through the mail
1920s	Radio broadcasts by correspondence schools
1960s	Open university public television programming
1980s	Video-based teleconferencing via satellite to remote classrooms by universities and military institutions
1990s	PC-based learning in the home and workplace delivered by CD-ROMs and the Internet
2000s	Online virtual universities via broadband Internet

Figure 2.1: Innovations in distance learning technology. Based on PBS (2004).

Radio broadcasts were introduced beginning in the 1920s (PBS, 2004). However, it was not until 1964 that the Schools of the Air were introduced in Queensland, Australia,

to directly teach numbers of similar-aged children via High Frequency radio broadcasts (Higgins, 1998).

The first television broadcasts were introduced into distance education in the United States in the 1930s. But again it was not until the late 1960s that the Open University of the United Kingdom systematised the use of public television to broadcast lessons to a wide audience. Television remains a popular vehicle for delivering distance education, particularly in the developing countries, because of the ubiquity and cheapness of the technology. The China TV University had 530,000 students in degree programmes in 1994 (Higgins, 1998). India has delivered agricultural training to rural villagers via television since 1967 (Chaudhary, 1992). And Cuba has launched the "University for All" which is "part of a campaign to widen the availability of education both among student youth and working people of all ages. It consists of nationally televised courses on various subjects such as English, geography, and art appreciation" (Militant, 2001).

By the 1980s, the United States military was pioneering the use of teleconferencing via satellite, using 1-way (and later 2-way) video and 2-way audio to deliver lectures and interact with students simultaneously to multiple sites. This technique is now widely used by universities in the United States, Japan and elsewhere to broadcast lectures to satellite campuses and learning centres, and is widely preferred by students as the "next best thing to being there" (Aniebonam, 2000; PBS, 2004). More recently, the World Bank has sponsored distance education via satellite in Africa and Latin America (Cook, 1998).

In Queensland, during the 1980s the Education Department began to experiment with technology in an effort to bring many of the interactive and multidimensional benefits of classroom-based teaching to distance education. This led to the revision of the curriculum and the development of teaching "packages", incorporating multimedia teaching aids such as video and audio tapes which could be studied virtually anywhere, anytime. This created the elements of an open or flexible learning approach to teaching and learning for its students (Higgins, 1998).

The invention of the personal computer, the CD-ROM, and the world-wide web meant that, by the 1990s, computers and computer networks had become the principal alternative to the mail system for the delivery of distance education. Healey et al. (1998) cite Canadian examples where Web-based distance education has enabled students in isolated rural schools to study senior high school subjects like chemistry, despite no qualified teacher being available locally.

Most recently, "virtual universities", resting on broadband communications networks, have emerged as the broker for learning produced by several or many universities (Jones and Pritchard, 1999), with mixed success. Virtual universities are discussed further in the following section.

2.1.3 The university and distance education

The formation of the National University Continuing Education Association at the University of Wisconsin at Madison in 1915 marked the beginning of university involvement in distance education in an institutionalised way (PBS, 2004). Over the ensuing decades, universities have extended into distance learning along three axes: running courses at satellite universities and learning centres, often via video hook-up; participating in open-learning, adult education extension programmes; and providing individual, correspondence-based lessons for home-study that are usually an extension of courses taught internally by the university, i.e. extramural study.

Distance education at the university level has special characteristics flowing from the unique role of the university in society. The Dictionary of Ideas (Norton, 1994) defines the university as "an institution of higher learning" and points to its long tradition dating back to Salerno in the 9th Century. The Universal Dictionary (Wyld, 1932) traces "university" to the Mediaeval Latin word meaning "the whole" and suggests it referred to a university being a "combination of all the Faculties", and the "idea of the whole of learning being taught." The American Heritage Dictionary (American Heritage, 1992) includes "the buildings and grounds ... and the body of students and faculty of such an institution" in its definition. All these definitions point to a university amounting to more than the sum of its courses. Higher learning is more than just instruction. It encompasses the entire intellectual life of the university, including study, debate, criticism, experimentation and research led by faculty, out of which university teaching and learning is synthesised. Distance education provided by universities draws strength from this *community* of learning and its greater possibilities for drawing students into the *whole* of learning, including the possibilities for postgraduate research and study.

Universities have developed strategies and techniques to try to realise more of this potential for distance learners. Massey University has used contact courses to bring its extramural students together into temporary learning communities and is exploring methods for replicating this approach independently of the classroom using computer technology (Prebble, 2001b). In 2000 Massey University's Information Technology and Distance Education Taskforce proposed that the University should "provide an

environment of on-line support for any and every paper offered by the University" (Massey University, 2000, p.12).

The approach of utilising computer networks to provide distance students with a richer learning environment, which more closely approximates the traditional university experience, is sometimes referred to as a "virtual university" (Brusilovsky et al., 2001). However, "virtual university" is more often used in the literature in the sense of a completely electronic institution which replaces a "bricks and mortar" university. In this virtual university all relations with the remote student are mediated through a computer network, the university may not have any actual physical existence, and the student's study programme may be sourced from multiple providers (Jones and Pritchard, 1999; Universal, 2001). Examples are the University of Texas Telecampus (UTTC, 2004) and the University of Phoenix (UP, 2004).

It is difficult to see how a purely electronic entity can reproduce the whole of learning - to imbue into distance education the intellectual life associated with an *institution* of higher learning with its "buildings and grounds" and "body of students and faculty". There is a real possibility that a virtual university will produce virtual rather than real understanding, as Chan (2003) discusses in relation to his experiences teaching with virtual laboratories.

It is noteworthy that the prestigious and purely distance education-providing Open University of the United Kingdom is based around a permanent, research-oriented faculty and campus. By way of contrast, a four-year attempt to establish a British "e-university" at a cost of £62 million was essentially scrapped in 2004. Critics labelled it a "fiasco" arguing that "the project overestimated the impact of the Internet, and that the most successful e-operations are built on the back of successful face-to-face ventures" (Education Review, 2004).

While Cheese (2003) argues that e-learning can potentially transform education at every level, he notes that many universities have resisted the e-learning trend. The main effect has been on the corporate training world while e-learning has had "little impact" in education, especially at the university level. "Universities don't see themselves merely as education 'content providers.' Universities have a proud tradition of combining learning, research, teaching, and professional development. If you look at higher education as a whole, it's not necessarily obvious how to implement e-learning" (ibid, para. 4) He adds that universities are also suspicious of the corporate side of e-learning: "As the flag-bearers of the concept of free knowledge and research, universities resist the commoditisation of knowledge" (ibid, para. 8).

In 2002, Massachusetts Institute of Technology launched a major new initiative which drew a clear distinction between its role as a learning institution and as a content-provider. It decided to make its entire core teaching materials freely available on the Internet for educators, enrolled students and self-learners. By June 1, 2005, the content of 1100 courses had been published online as part of the MIT OpenCourseWare project (MIT, 2005).

Universities are not the only or even the main providers of post-secondary distance learning today. They face growing competition from a range of more commercially-oriented providers, both public and private, that offer job-oriented programmes without the research focus of the universities.

The boundaries between university and non-university tertiary institutions are becoming increasingly blurred as all providers compete for paying students, and for government and industry funding. Polytechnics and even company training divisions are re-branding themselves as universities (Huynh et al. 2003), and universities are offering industrial training courses previously associated with polytechnics.

There is a general trend in tertiary learning away from a broad-brush, liberal education towards more narrowly-based training geared to current business and industry requirements. A major Australian government-sponsored study "The Business of Borderless Education" (Cunningham et al., 2000), for example, writes of the trend towards the "for-profit university" (p. 15), the "earner-learner" (p. xiii), and the "marketisation or commodification of knowledge" (p. 22), and sums up the shift as from "just-in-case" education to "just-in-time" training (p. xiii). Lennox (2000) calls it creating, distributing and updating knowledge in a "just-in-time, just enough" fashion (p. 2).

The potential for packaging computer-based distance learning into marketable education products for the "earner-learner" has driven much of the research and development in this field. In fact, "The Business of Borderless Education" authors comment: "Those 'hyped' the industry often simply conflate the marketisation of education with the emergence of online education and training" (pp 22-23). And Drucker (2000) writes: "Online continuing education is creating a new and distinct educational realm, and it is the future of education. There is a global market here that is potentially worth hundreds of billions of dollars."

Noble (1997, 1998a, 1998b, and 1999) and Taylor (2000) reflect a broad disquiet among academic staff and students in face of this drift of universities towards becoming what Noble calls "digital diploma mills". Noble (1997) quotes Educom president Robert Heterich as observing: "Today you're looking at a highly personal

human-mediated environment [in university teaching]. The potential to remove the human mediation in some areas and replace it with automation - smart, computer-based, network-based systems - is tremendous. It's gotta happen" (para.24). Noble's conclusion is that the adoption of computer-based learning technologies by some universities is "but a vehicle and a disarming disguise" for the commercialisation of higher education (Noble, 1997, para. 5).

Notwithstanding this growing commercialisation of tertiary education and the particular impetus it has given to e-learning, world-wide distance education *at the university level* is still overwhelming provided on a non-profit basis by the public universities and will continue to be so for the foreseeable future. It is also clear that computers, linked together through the public communications system into extensive networks, are playing an increasingly important role in the delivery of extramural courses.

2.1.4 Summary

Distance learning has a long and diverse history based around correspondence lessons using the postal system and augmented increasingly by information technology. From the beginning, private, for-profit, industry-oriented programmes have been a significant component, although the primary driver has always been providing public education to all those unable to attend normal classes. At the tertiary level, the line between these two trends has become increasingly blurred. However, on a world-wide basis, university-level distance education remains overwhelmingly provided by public, non-profit institutions.

Distance learning is an umbrella term encompassing many different settings, each with its own unique requirements – occupational training by business, military or public institutions; home schooling of primary or secondary students; "open university" cultural or self-improvement courses; group learning at a satellite university or learning centre; or individual extramural university study. Students may be distance learners because its flexibility suits them, or because they have no other option open to them.

It is therefore necessary to assess distance learning technologies in a very specific context. Factors such as the character and aims of the teaching organisation and its students, the subject matter and the breadth and depth to which it will be taught, the age and learning level of the target students, the accessibility of the technology, and the location of the students, have to be taken into account.

University-level distance learning has been most successful when it has been built on the back of successful face-to-face teaching and a permanent research-oriented

faculty. Traditional universities are thus better placed to deliver effective distance education than purely electronic virtual universities.

The uptake of technology varies greatly between the different forms of distance learning, with the greatest uptake in training applications, and the least in extramural university study. At the least, this suggests that there are still issues to be resolved before the broader sweep of university education can fit with e-learning technologies as readily as does occupational training.

2.2 Computers in education: Milestones in computer-based learning research

Since the 1950s the potential for computers in education has attracted the attention of learning psychologists, educationalists, and computer scientists. Research has evolved along two major axes: using computer technology as a practical tool to improve the productivity of teachers; and using computers to better understand how people learn and to develop more effective methods of teaching (Park et al., 1987). Milestones in computer-based learning research have reflected both innovations in computer technology and shifts in the underlying pedagogy. In this section, the major milestones are reviewed chronologically.

2.2.1 Teaching machines

During the foundation years of computer science in the 1950s and 1960s, the behaviourist approach to learning developed principally by J. B. Watson and later B. F. Skinner prevailed among psychologists (Hill, 1997, pp. 33-34). Based upon laboratory studies of animals, Skinner's behaviourism viewed learning as the modification of observable, external behaviour through an appropriate mix of stimuli, responses, feedback and reinforcement (Reeves, 1994).

Behaviourism focussed upon what people *do* rather than what they *think*. As such it rested upon a theory of knowledge as immutable truths external to the mind, which only needed to be rote-learned by students. It emphasised learning as training students to make the correct responses to questions and inspired a teaching method known as programmed instruction. "Programmed instruction is characterised by clearly stated behavioural objectives, small frames of instruction, self-pacing, active learner response to inserted questions, and immediate feedback regarding the correctness of a response. Individualised instruction in essence replaces the teacher with systematic or programmed materials" (Clark, 1999b, para. 1). Skinner believed that programmed

instruction should be linear, presenting the same sequence of steps to each learner, regardless of their responses (Hill, 1997, p. 81). In 1958, he designed an interactive mechanical teaching machine which implemented this method.

Most early developments in computer-based learning reflected Skinner's concept of the teaching machine, notably PLATO (Programmed Logic for Automatic Teaching Operations) (DeCloque, 2000).

Several studies have shown that the programmed instruction approach can be an effective learning method in areas suited to rote-learning (Park et al., 1987). Even advocates of more open-ended, discovery types of learning such as Underwood et al. (1990) acknowledged that such "practise is vital if skills are to reach the level of automaticity necessary to allow the individual to focus attention on higher level problems" (p. 22) although they criticise drill-and-practise software for delivering "little more than computerised worksheets" (p. 30). The approach has been widely used, often in a games format, to provide an interactive environment for practising skills, and revising material that has already been taught (Alessi et al., 1991; Ch. 3).

An important modification to the linear programmed instruction method was branching. Branching enabled the learner's response to determine what material they received next from the teaching machine (Hill, 1997, p. 81). Branching enabled the system to accommodate different learning levels. Students could skip steps they already knew, or study remedial material on information already presented (Clark, 1999b). As a basic programming construct, it is straightforward to implement in software. It has been observed that many adaptive tutoring programs that emerged in the 1980s were in fact "no more than courseware with sophisticated branching" (Alessi et al., 1991, p. 452).

Behaviourism no longer predominates among learning theorists, and few educationalists or e-learning researchers would advocate the programmed learning approach today. Nevertheless, the observation in Reeves (1992) that, despite its widespread debunking, many computer-based learning systems continue to use instructional models directly derived from behavioural psychology, retains validity more than a decade later. At least two papers presented at ICCE 2003 discussed projects which combined the latest computer technologies with programmed instruction pedagogy. One paper outlined a hand-held PDA-based system for teaching history to 6th grade school children (Ishizuka, 2003), while another discussed a web-based system for drilling students in basic IT concepts (Tominaga, 2003).

There are many systems still available on CD-ROM or over the Web which use programmed instruction to drill school students in basic science, spelling, mathematics,

etc. Equations, for example, is a chemistry primer available on CD-ROM (Keall, 2001). And NuMaths, (Queensland College, 2004) can be accessed over the Web.

Even in courseware, the legacy of the programmed instruction approach can be seen in the form of the fill-in-the-blank and multi-choice revision quizzes and tests. Nevertheless, its relevance to the kind of broad-brush education associated with university-level distance learning would seem to be quite limited.

PLATO and TICCIT

Before the advent of the personal computer in the 1980s, prototyping and testing educational software was more problematic than it is today. Research in computer-based instruction was based around a handful of large-scale projects backed by government or corporate funding. PLATO and TICCIT are generally recognised as the trailblazers of modern computer-based learning systems (Park et al., 1987; Alessi et al., 1991; DeCloque, 2000).

From its beginnings in the 1960s, PLATO developed during the 1970s into a mainframe-based timesharing system that could serve hundreds of students at different localities simultaneously. PLATO's developers were guided by practical rather than theoretical concerns (DeCloque, 2000). While it reflected the prevailing programmed instruction approach, PLATO went beyond the usual drill-and-practice philosophy by including tutorial material, inquiry, dialogue, simulation, computer games, and problem solving to help students develop an understanding of the subject matter (Gledhill, 1981, p. 153). PLATO pioneered the integration of text and graphics, course-authoring software, and on-line conferencing and communication (Woolley, 1994). It was thus a proving ground and a benchmark for the PC-based courseware authoring and delivery systems that emerged in the 1980s and their web-based counterparts of the 1990s. These systems incorporated many of the features, and the practical orientation, pioneered by PLATO.

In 1967, the MITRE Corporation began developing a multimedia learning system by combining the recently developed mini-computer with television technology (DeCloque, 2000). TICCIT, as it was called, was built around an instructional design approach called Component Display Theory which introduced the concept of learner-controlled instruction and individualisation (Kearsley, 2003; Alessi et al., 1991, p. 1). Special keys enabled the student to exercise some control over both the content and the learning strategies used for study. TICCIT's major legacy is being the first multimedia computer-based learning system, and the first to be designed around a specific instructional theory (DeCloque, 2000).

2.2.2 Thinking machines

Programmed instruction is concerned with whether the student gives the correct answer to a question. However, many learning scenarios do not lend themselves to this approach. In problem-solving, for example, the teacher may be as interested in the intermediate steps - the reasoning process by which the student reaches an answer - as in the answer itself. Moreover, on noting flaws in a student's reasoning, the teacher can adjust the content and method of their instruction to address that student's learning difficulties, which a teaching machine cannot do. "Traditional CBI programs (mostly of the tutorial, drill, simulation or game varieties) embed their pedagogy within the content of the lesson. That is very specific instructional techniques are selected and tailored to the particular content of the lesson. The most immediate result of this is that such lessons provide limited adaptation to student needs" (Alessi et al., 1991, p. 461).

During the 1970s, there was an increasing interest in improving learning strategies and outcomes by incorporating into educational software the ability to model and adapt to the individual student's understanding. Well before this point, cognitive psychology had supplanted behaviourism as the dominant trend among learning psychologists (Clark, 1999c). Cognitive psychology emphasises the inner workings of the mind in learning. In their efforts to better understand how the brain worked, the cognitive psychologists drew parallels with the information processing capabilities of computers. They became interested in using computers as tools for modelling the mind and exploring their learning theories, "so that we can say, only partly in jest, that psychologists study three kinds of organisms: humans, animals, and computers" (Hill, 1997, p. 116).

This interest drove research into developing computer software that could teach like a human tutor, which have become known as intelligent, or adaptive, tutoring systems (ITS). Murray (1999) defines intelligent tutoring systems as "computer-based instructional systems with models of instructional content that specify what to teach, and teaching strategies that specify how to teach... Instructional models allow the computer tutor to more closely approach the benefits of individualised instruction by a competent pedagogue" (para. 1).

ITSs typically contain a student model, a teaching model and the course contents as separate components. The student model contains information about the extent and level of a student's understanding of the course. By separating the teaching model from the course contents, the teaching strategy and content can be adjusted to suit the student's understanding and ability. SCHOLAR (Carbonnel, 1970), a system for teaching South American geography developed at the Massachusetts Institute of

Technology, provided the initial impetus to intelligent tutoring research (Woolf, 1990; Park et al., 1987). However, the major driving force has come from scientists at two Pittsburgh universities – the Carnegie Mellon University and the University of Pittsburgh – including major contributions on human problem-solving (Newell et al., 1972; Chi et al., 1981), computerised tutors (Anderson et al., 1995) and automated voice recognition (Mostow, 1994).

Research into intelligent tutoring systems is part of exploring computers as "thinking machines." As such it has overlapped with the development of expert systems – computer programs which provide specific problem-solving skills by manipulating symbolic representations of knowledge to simulate the behaviour of human experts (Harmon et al., 1985, p. 5). Both tutoring and expert systems have had most success when focussed upon specific topics within well-structured domains such as mathematics, programming or medical diagnosis. Knowledge in such domains can be readily expressed as sets of facts, rules and relationships defined by formal logic. "The database of SCHOLAR is a complex but well-defined structure in the form of a network of facts, concepts and procedures" (Park et al., 1987, p. 18). A number of adaptive tutoring systems have been built as extensions of expert problem-solving systems. GUIDON, for example, was developed in 1979 on top of the medical diagnosis expert system MYCIN (*ibid.*). The LISP Tutor developed in 1983, contains an expert system which traces student's solutions, offering advice where necessary (Corbett et al., 1990). The LISP Tutor's authors cite research showing that "while the tutor is more effective than 'learning on your own,' it is not as effective as a human tutor" (*ibid.*, p. 84.).

SQL-Tutor (Mitrovic et al., 2000), PAT (Ritter et al., 1998), and RIDES (Munro et al., c.1995) are examples of systems which have progressed to the point of being used in regular school and training environments. However, most ITSs appear to have run into problems that have made them unviable beyond their initial prototype environments. Inhibiting factors include:

- difficulties in building the mental model of the student (Underwood et al., 1990, p. 20);
- over-ambitious goals such as trying to outperform a teacher (Patel et al., 1997);
- the complexity of the authoring task (Murray, 1998b; Alessi et al., 1991, p. 452);
- constant shifts in the technological platform rendering tutors obsolete (Patel et al., 1997; Murray, 1998b); and

- a focus on developing learning theory, rather than on building successful instructional systems (Laurillard, 1993, p. 76).

While the practical goal of artificial intelligence in education is to produce computer-based instruction that is more adaptive (Park et al., 1987), this has not always been the priority of the researchers themselves. For instance, the initial motivation of the team that produced the LISP, Geometry and PAT Algebra tutors was learning more about skill acquisition rather than producing practical classroom results (Anderson et al., 1995).

During the last few years, researchers have addressed these issues along three major axes:

- The development of authoring tools, with which domain experts (i.e. teachers) can build computerised tutors, analogous to the authoring shells that have been developed for expert systems. Murray (1999), Munro et al., (c.1995), Nakabayashi et al. (1996), Ritter et al. (1998), Hsieh et al. (1999) and Martin et al. (2003) discuss examples of authoring tools.
- The construction of web-based tutoring systems, either by porting standalone tutors to the web or by building native web applications. Ritter (1997), Vassileva (1997) and Mitrovic et al. (2000) report on existing tutoring systems being re-engineered for the web, while Okazaki et al. (2003) and Merceron et al. (2003) describe new web-based tutors.
- Extending the techniques pioneered by ITS research into broader learning applications. The editors of "Artificial Intelligence in Education" describe their domain as "applied cognitive science" in which "the focus [is] on developing computational models of relevant aspects of learning and teaching processes (Hoppe et al., 2003, p. v)." Of particular current interest is the application of cognitive modelling to web-based learning. The areas that have been explored include: adaptive courseware (Weber et al., 2001; Vassileva, 1997), interactive electronic textbooks (Brusilovsky 1999; Murray et al., 2000), context-sensitive adaptive help systems (Benyon et al., 1993), the integration of intelligent tutoring tools into broader learning environments (Patel et al., 1997 ; Gehne et al., 2001), class monitoring aids for teachers (Merceron et al., 2003; Despres, 2003; Mazza et al., 2003), supporting virtual communities for collaborative learning (Greer et al., 1998; Vizcaíno et al., 2002; Tiffin, 2002), and web search strategies (Mitsuhara et al., 2002; Hasegawa et al., 2002).

If adaptive tutoring systems have so far had only a limited practical impact in the classroom, nevertheless, many of their lessons have been incorporated into subsequent research on computer-based learning. Moreover, by demonstrating that computer software could teach skills without the immediate supervision of a human tutor, ITSs have also demonstrated the potential for computers to assist students to learn in their own time and *at a distance* from a teaching institution.

2.2.3 Discovery learning

The launching of the personal computer (PC) and the graphical user interface (GUI) for the mass-market in the 1980s revolutionised computing. Hitherto, computers had been for the most part the domain of highly skilled personnel operating large computer systems for companies, government departments or research laboratories. With the advent of the PC, computers became a practical alternative to books, tapes and videos as a medium for *popular* instruction. Students could use them as a learning tool at home or at work, as well as at school.

The PC made possible the widespread introduction of computer-based training into the workplace. By the mid-1980s, systems were emerging that combined laser discs and computer software to provide an interactive video training environment (Dodds, 2003; Alessi et al., 1991), later superseded by digital CD-ROM and DVD technology. Training applications incorporating sound, graphics, animation, or video, could be downloaded onto workstations via a removable disk or company local area network (LAN). Multimedia courseware authoring tools like Authorware (Macromedia, 2005), which built on the groundwork laid by PLATO and TICCIT, were developed to cater for this demand. Clark (1999d) considers this marks the beginning of individualised computer-based training that goes beyond programmed instruction.

The possibilities offered by PC-based learning programs for self-paced, individualised and interactive education was embraced by advocates of a constructivist or discovery approach to learning who "favour hands-on, self-directed activities oriented towards design and discovery" (Wenger, 1998, p 279). Mayes (1993), for example, wrote of "the important shift in recent years brought about by the power of modern computers to provide general purpose learning environments in which the learner can seek information in the pursuit of understanding. In such an environment the learner can explore, discover, create, edit, ask questions, seek information, and communicate with other learners" (para. 4). Among educationalists, a constructivist pedagogy, in which "teachers are seen as facilitators or coaches who assist students to construct their own conceptualisations and solutions to problems", has had numerous advocates since the

1930s (Clark, 1999e, para. 3). Many consider this approach particularly relevant for the more advanced level of learning required at a university (Jonassen et al., 1993). In fact, by the 1990s this approach was so widely accepted that Mayes (1993) could write: "There is a sense in which 'we are all constructivists now'" (para. 5).

Programmed teaching and computerised tutoring approaches both involve close control of the learning process by the computer (Hill, 1997, p. 129). This "instructivist" approach is criticised by "constructivist" educationalists for emphasising learning as a process of knowledge acquisition - where "one only has to organise knowledge in an appropriate form to match the conceptual state of the individual learner and learning will occur inevitably" - rather than as a process of knowledge construction (Mayes, 1993, para. 3).

Perhaps the simplest form of discovery learning is to explore a book at will. And in its simplest form, computer-based discovery learning is represented by an electronic book which the student can also explore at will, such as the online Encarta encyclopaedia (Microsoft, 1997). An active area of research in recent years has been the development of more interactive and individualised electronic books (Sinista et al., 1999; Anastaides, 2003). In particular, efforts have been directed towards electronic books which not only adapt but also guide the individual student towards desired learning outcomes (Brusilovsky et al., 2001; Murray et al., 1998).

To support a more constructivist learning environment, and provide students with a range of learning options, efforts have been made to integrate a variety of learning tools and technologies into a single package (Patel et al., 1997; Jesshope et al., 2000).

The World Wide Web has created new opportunities for discovery learning. Learnz2001 (2001), for example, enabled school students to participate in a science field trip to the Antarctica from inside their classroom. The Web presents students with a huge database of knowledge to explore beyond anything which a single learning institution can provide (Web-Based Education Commission, 2000). At the tertiary level, there have been efforts to create web-based "virtual laboratories" (Chee, 2001; Cheng et al., 2001).

Computer-based simulation offers particular opportunities for learning by exploration and doing. Virtual environments that simulate industrial processes and machinery have also been an important aspect of interactive computer-based training. Computers and simulation software offer a company a cheaper and safer option for the training of novice operators and technicians than risking expensive machinery. The British Polymer Training organisation, for example, offers injection moulding training

programmes, which enable trainees to set up and troubleshoot processes on a simulated moulding machine (BPTA, 2001).

Much of the impetus for research into computer-based training in general, and simulators in particular, has come from governments wanting to develop computer-based systems to train military personnel (Dodds, 2003; Seidel et al., 1995). RIDES (Munro et al., c.1995), developed at the University of Southern California for the United States Air Force, is an authoring tool that enables non-programmers to develop tutoring software which simulates machine processes.

2.2.4 Communities of learning

Initially, most computerised training was via standalone programs, although as PCs began to be linked into LANs within companies in the early 1990s, some course management functions were centralised (Dodds, 2003). In addition, most training software could be installed and run from removable storage disks and thus could be utilised for distance study. But none of these systems addressed the sometimes “lonely”, “anonymous” and “boring” experience of computer-based training for the distance learner (Lennox, 2000). It has taken the World Wide Web (WWW) to open up to the remote learner the *communication* and *collaboration* potential provided by networked computers.

As early as the 1960s, the United States Defense Department funded a project to connect university computer scientists and engineers together via their computers and telephone lines (Bonelli, 1998, p. 4). By 1990, hundreds of thousands of users were communicating over the public computer network which has become known as the Internet. However, with its “entangled web of Unix, text-based commands”, sharing documents across the Internet remained inaccessible to the lay user until a graphical, hypertext navigation tool called the World Wide Web was introduced to the Internet in 1992 by software engineer Tim Berners-Lee (ibid., p. 9). This made accessing documents on remote computers by means of the Internet as easy as using a graphical operating system.

The WWW had important practical implications for university education. It meant that instructors could place lecture notes and links to supporting materials on a course web site where students might read them, or download them onto their own computer. In addition, the communication and conferencing capabilities of the Internet meant that students could use their computers to communicate with their teachers, discuss with each other and even work together on a common project.

The biggest implications have been for distance education. "Web courseware installed and supported in one place can be used by thousands of learners all over the world that are equipped with any kind of Internet-connected computer" (Brusilovsky, 1999, para. 1).

By the mid-1990s, systems for the authoring and managing of courses delivered over the Web were emerging. These included WebCT, developed at the University of British Columbia (Goldberg, 1997), and Blackboard/CourseInfo, developed by graduate students at Cornell University in 1997 (Kubarek, 1999). Today there are thousands of courses world-wide that are being supported or taught using such software, frequently referred to in the literature as "web-based courseware" or "Learning Management Systems" (LMS).

The emergence of the WWW also posed new opportunities and challenges for research into how people learn and how to improve learning through computers. Commenting on the contributions to ITS2000 the conference organisers noted: "Most efforts focus on ITS on the Web. The challenge is not simple, as one is expected to demonstrate not just how the Web is used for teaching, but how it should evolve to facilitate human learning" (Cerri et al., 2002, p. vi). It has had particular importance for those who emphasise the social over the individual aspects of learning.

A common thread from programmed instruction through to PC-based simulation has been the emphasis upon the student learning as an individual through one-to-one interaction with the computer. However, learning is essentially a social rather than an isolated, individual activity. Wenger (1998) writes of placing "learning in the context of our lived experience of participation in the world" (p. 3), and of consciously promoting collaborative "communities of practice" in which people learn through shared experiences. One of the most important strengths of studying internally on a campus is the opportunity for students to participate in the complex and multifarious learning relationships that constitute the university.

As computers began to be linked together into LANs and then into wider networks via the telephone system, the potential to foster more collaborative forms of distance learning within military and business training programmes was noted (e.g. Seidel and Chatelier, 1995). The ubiquity of the WWW and the PC raised the possibility of building similar learning communities among those studying at home as well.

Collaborative learning strategies which take advantage of the communication and co-operation capabilities of networked computers, and the technologies which enable them, have now become a major focus of computer-based learning research. This has

particular implications for computer-based *distance* learning. Web-based courseware systems incorporated asynchronous email, bulletin board, and threaded discussion tools and synchronous chat tools as standard features. WebCT's developers, for example, "added tools that facilitate student communication and collaboration" (Goldberg, 1997, para. 4). GENTLE (Dietinger et al., 1998) offered annotation tools, discussion forums, and online chat.

More recently, research has been directed towards both the technical and pedagogical challenges of transforming the basic communication facilities of the WWW into more sophisticated collaborative learning tools (e.g. Jong et al., 2003; Pinkwart et al., 2002; Greer et al., 2001; Ishikawa, 2002). A more detailed evaluation of web-based learning research is the subject of Chapter 3.

2.2.5 Summary

In this section the major milestones in computer-based learning research since the 1950s have been reviewed historically, from programmed instruction on mainframe computers, through adaptive tutoring and discovery learning on the PC, to collaborative learning on the Web. While each new milestone has represented an advance in terms of technology and pedagogy, the earlier approaches have not been entirely superseded.

PLATO has provided the benchmark for those researchers interested in developing practical computer-based teaching tools. Strongly influenced by programmed instruction, its legacy can be seen in modern web-based and PC-based training and authoring systems.

Researchers whose primary interest has been the use of computers to better understand and improve the learning process have enjoyed far less practical success in terms of systems that have moved beyond the prototype stage. Nevertheless, the techniques they have pioneered are drawn upon by those seeking to improve the learning effectiveness of courseware.

At the university level, it would appear that the discovery and collaborative learning are most broadly applicable, although some ITSs have been successfully used at this level in areas such as programming. Computer systems which can integrate all these approaches into a single environment appear to hold particular promise for distance learning.

2.3 Networked computers and distance education

So far in this chapter, the evolution of distance education and of computer-based learning research have been reviewed. In this section, some lessons are drawn from this review concerning the potential for networked computers to enhance distance education, with particular emphasis on university distance education. It concludes by discussing some of the factors which limit the realisation of this potential.

A traditional university education is above all a *social* process involving a complex set of interactions among students, and between students and instructors, on many levels, formally and informally. One of the most important strengths of studying on-campus is the opportunity for collaborating in the complex and multifarious learning relationships that constitute the university. For computer-based distance education to be most effective, it must find ways to draw extramural students into this university community of learning.

The special requirements of university distance learning flow from the student's *isolation* – from other students, from teachers, and from support materials and resources. In distance learning, there is no human tutor looking over the student's shoulder to help her/him out of a blind alley. There is no lecturer's door to knock on, nor group of students to sit down with to work through an assignment. These social aspects of university learning must be provided by other means such as better preparation of learning materials and finding alternative means of accessing tutors and other students.

In computer-based distance learning the goal is to utilise the capabilities of computer systems to provide the functionality needed to bridge the gap between a traditional socialised university education and the isolation of the extramural student. From the literature review, computers and computer networks have been identified as having the *potential* to enhance distance learning through the following features and functions:

1. *Allowing learning material to be accessed from almost anywhere, at any time.* They can expand the opportunities to study, learn, acquire new skills and develop a broader outlook in places where universities are not otherwise accessible. By providing access to large banks of information, a library is placed at the door of every distance learner.
2. *Facilitating communication*, thereby reducing remote students' isolation. Students can use their computers to consult human tutors and other students either synchronously, as with communication by telephone, or asynchronously, as with communication by post. Computer-based communications are fast and are largely

distance-independent. The work of external students can be assigned, submitted and graded in the same time frame as for internal students. The distance student and their teacher are brought closer together.

3. *Promoting co-operative work*, by enabling distance learners to join-in group discussions and work collaboratively with others on problems and projects.
4. *Integrating the various media used to deliver distance education (video, audio, telephone, mail, graphics, and text) into a single multi-media environment*, making it easier to deliver more of a multi-dimensional learning experience to the remote learner. A student can even participate in a field trip without leaving home.
5. *Simulating real world situations, processes and problems*, with which the student can interact and learn by exploration and practice.
6. *Integrating materials from a variety of sources and locations*. Computers can provide a student with more learning resources than might be available in a single classroom or learning institution.
7. *Allowing the presentation of material to be adapted to suit the individual student*. Computers can enable the individual distance learner to tailor their learning environment to take into account physical disability, language, and navigational and learning preferences. They can track a student through a study session, and offer context-sensitive references and help. They can remember how far a student has progressed through a course and return to that point when the student starts their next session.
8. *Acting like a human tutor*, by dynamically adapting what, how and when material is presented to suit the individual student's current knowledge level. Computers can be programmed to help a student even when a human tutor is not available to answer questions and to provide immediate feedback when needed.
9. *Helping the teacher to author teaching material*. Computers can provide templates to guide teachers in preparing material that best serves a particular learning goal. They can store learning materials in forms that facilitate their re-use by other teachers and courses.

In short, computer systems have the potential to provide distance learning *environments*, which incorporate passive, teacher-centred – or active, student-centred – learning styles; group or individual work; interaction; and simulation.

Notwithstanding this potential to enhance distance learning with computer technology, there are also real limits on what can be achieved today. These limits are discussed in the next section.

2.3.1 Constraints on use of computers in distance education

There is an extensive literature espousing the advantages of e-learning over traditional face-to-face teaching methods. For example, an online brochure promoting a major provider of Internet-based network engineering training argues that e-learning "eliminates barriers of time, distance, and socio-economic status, allowing people to take charge of their own lifelong learning...[and] is also a highly effective tool for reaching disadvantaged and at-risk constituencies world-wide..." (Cisco, 2000, p. 5). And, Drucker (2000) asserts that "online continuing education" is time-efficient, cost-efficient and has an interactive advantage over the typical classroom that makes it the future of education.

A major United States government study stresses the potential of the WWW for discovery and collaborative learning beyond what is possible in a single classroom. It also observes that this potential is often not fully utilised because the technology is used to mimic the "top-down, lecture or text-driven model of instruction" rather than "exploring its interactive potential" (Web-Based Education Commission, 2000, p59).

However, there are important constraints on the use of networked computers in education for the foreseeable future, especially as it applies to university extramural study. These exist on the technical, socio-economic and pedagogical levels.

Technical

Aniebonam (2000) highlights several disadvantages of computer-based distance learning systems due to the complexity of the underlying technology, which he contrasts to the simplicity of the book and mail system. Performance is limited by disparities in communications infrastructure and computer platforms. Computer networks are slower, less reliable and more vulnerable to accidental or malicious misuse than standalone computers. Computer viruses flourish in the environment of the Internet.

These performance problems are exacerbated when a global view is taken of distance education. Access to telephone service is still a major issue in most developing countries and parts of Eastern Europe (Cook, 1998). In 1999, two-thirds of the world's households had no telephone service, and 97 percent had no Internet access (Hope,

2002). Not even the more developed countries are exempt. Wright (2002) reports that for many parts of the New Zealand countryside: "Serious Internet use is out of the question, with unbearably slow connections which frequently drop out plaguing many in the rural sector."

Hope (2002) argues that web-based applications are relevant only to a minority of 150-200 "wired cities". While various initiatives are under way to improve telephone service and extend broadband coverage in rural areas in New Zealand and elsewhere, there will remain a significant number of would-be extramural students for whom serious Internet use is out of the question for the foreseeable future.

A related issue is the question of computer literacy. Delivering distance education via the Internet assumes that the recipient is able to use a relatively complex system unaided. Given the older demographic of "second-chance" learners and the relatively narrow base of experienced computer users viewed on a world scale, this is not a safe assumption.

Socio-economic

Online learning does not address the problem of the poor and women being neglected in education (Bork, 2001b). In socio-economic terms, technological disparities and high entry and maintenance costs deepen the divide between the haves and have-nots, town and countryside, developed and developing countries. A networked computer is beyond the reach of most of the world's population, and will remain so for the foreseeable future. "[F]ew of the poor, even in wealthy countries, have access to the Internet. So it increases the gap between rich and poor" (Bork, 2001a, p9). In 2002 it was reported that the Maori university, Te Wananga o Aotearoa, was providing mobile phones to its distance students because a majority had no access to landlines or lived in remote areas and needed them to contact tutors (One News, 2002).

In this context, computerisation actually increases the entry cost to distance education and reduces access for the disadvantaged. Broadband requirements for the delivery of multimedia education further disadvantages those who cannot access such a service. It is notable that mass-based distance education initiatives in developing countries have been focussed upon television rather than the Internet (Cook, 1998).

Pedagogical

Computerisation cannot compensate for bad teaching and poor preparation. It may even encourage it (Stirling, 2001). Computer-based courses are the hardest to prepare,

because computers cannot “think on their feet”. Course materials can be difficult and time-consuming to prepare, leading to a reliance on presenting lessons as text to be read (Aniebonam, 2000). The good teacher must be programmed into the course. The technological limitations of computer systems can begin to shape and limit the pedagogy.

The focus on what computers can deliver rather than what learners need means “that students may need to reconfigure their minds to suit the technology. The tail is now wagging the dog” (Taylor, 2000). For instance, there has been a growing emphasis on assessment by automated tests at the expense of the essay (Mills, 1997, p. 185). It can encourage the “myopic tendency to view education as training”, and knowledge as a “non-reflexive facility with large quantities of information” (Taylor, 2000). Mayes (1993) observes that presenting learning material through multimedia to try to make it more attractive or more palatable is doomed to failure because it rests on the mistaken notion “that vividly presented information, with which the learner is asked to interact in some way, will [inevitably] lead to better learning” (para.6).

Computerisation cannot substitute for the teacher altogether. Teaching is above all a human relation between teacher and student. As such it is a dynamic process which can never be fully captured in software. Educational software may increase the productivity of individual teachers. It should certainly aim to extend a lecturer's scope (Patel et al., 1997). But it can never fully replace the human teacher. Computer-based learning tools should be seen as a means to better teaching and not as an end in themselves (Roseth, 2001). By “using them as tools it remains for the teacher to make the important decisions about educational policy” (Underwood et al., 1990, p. 21).

A successful strategy for utilising networked computers for distance education needs to take into account all the pitfalls and design to minimise or avoid them.

2.4 Conclusion

This chapter began with an historical review of distance learning from its 18th Century beginnings up to the present day. While there have been a variety of threads and motivations in distance learning, at the core of most have been correspondence-based lessons. These have been increasingly supplemented by learning activities made possible through technological innovations – from radio broadcasting to video-conferencing. Networked computers have opened new possibilities for more closely integrating these activities into a multimedia learning environment.

The evolution of computer-based learning research and development has also been extensively reviewed. It was noted that this evolution has been shaped both by advances in computer technology and shifts in the influence of various theories of how people learn and consequently what matters in education. Current research emphasises collaborative and discovery approaches to learning, especially at the university level, but the earlier cognitive modelling and programmed teaching approaches retain significant influence.

Networked computers have enormous potential to enhance the learning environment for distance students. But realising this potential is not a given, and in many real world scenarios a reliance upon computer delivery can have a negative impact. Furthermore, technology that may be appropriate for one area, such as supplementing the teacher in a primary school classroom or providing work-based training, may be inappropriate for another such as home-based university study. For these reasons it is necessary to clearly delineate both the domain and the target audience for an e-learning system.

The following chapter will develop some broad criteria for evaluating the suitability of educational software for university-level distance learning. These criteria will then be used for evaluating web-based courseware.

Chapter 3

Web-based e-learning: an evaluation

This chapter presents an evaluation of web-based e-learning from the perspective of its suitability for extramural e-learning. The primary focus is the tools for authoring, administration and delivery of web-based courses. This involves first clarifying some terminological issues, before developing a set of evaluation criteria based upon the literature reviewed in Chapter 2, including clearly defining the target domain. To the general criteria for evaluating software systems - *functionality* and *usability* - *accessibility* for the remote user is added.

Next, web-based e-learning is assessed against the functionality, usability and accessibility requirements of the extramural student. Some issues with courseware are highlighted, which need to be addressed if e-learning is to offer a universal and practicable alternative to traditional correspondence-based study. Recent technological innovations and research relating to these shortcomings are discussed and assessed. Drawing some strands together from that discussion, the key requirements for computer-based delivery and support of extramural study are summarised, and the fundamental issues that need to be resolved to achieve them are outlined. Finally, a method for achieving a practical solution to these problems is hypothesised.

3.1 Terminology

Within the literature, computer-based educational software has been referred to by a myriad of labels and acronyms - computer-based instruction (CBI), computer-assisted instruction (CAI), computer-assisted intelligent instruction (CAII), computer-based learning (CBL), computer-based distance learning (CBDL), courseware, web-based courseware, e-Learning, web-based learning (WBL), intelligent tutoring system (ITS), computer-based learning environments, computer assisted collaborative learning, and so on. While some of these terms can be and are used interchangeably, they in large part reflect shifts in technologies, emphases and underlying theories during several decades of evolving research and development in this field. During the 1990s the sharpest distinction was drawn between the adaptive ITSs with their narrow subject focus, and the broad-brushed, one-size-fits-all, courseware authoring and delivery systems.

More recently, a major distinction has been drawn between Learning Management Systems (LMS) and Learning Content Management Systems (LCMS). The Brandon Hall website, for example, explains that: "The primary objective of a learning management system is to manage learners, keeping track of their progress and performance across all types of training activities. By contrast, a learning content management system manages content or learning objects that are served up to the right learner at the right time" (Brandon Hall, 2005, para. 2). From this perspective, the primary target users of an LMS are "training managers, instructors, administrators", while for an LCMS they are "content developers, instructional designers, project managers" (ibid, para. 5).

Web-based courseware systems like WebCT were developed by universities to administer and support their internal and external courses on a one-size-fits-all basis. "Web-based courseware" is used here interchangeably with LMS and is the focus of the evaluation in this chapter. LCMS, which use learning object technology to deliver modules of learning material tailored to a particular individual, have been primarily developed to support corporate training and to date have had little or no uptake by universities. Learning objects are discussed in Section 3.5.4.

3.2 Overview of recent e-learning research

Computer-based learning continues to be a central arena of research for computer scientists. Several major international conferences are held on an annual or biennial basis including ICCE (Chee et al., 2003), ICALT (Devedzic et al., 2003), ITS (Cerri et al., 2002) and AIED (Hoppe et al., 2003). From a review of the proceedings of these conferences and of other forums for e-learning research, and from a wider review of relevant web-sites, a number of general observations can be made.

- While the collaborative and discovery (constructivist) approaches to learning predominate in current research, all of the historical trends reviewed in Chapter 2 remain represented in the field. This reality perhaps supports an observation by Nunes et al. (2003) that education designers draw upon their prior knowledge and experience even when it is inappropriate, and that even though objectivism (behaviourism) is considered heresy by many constructivists, it "prevails even today in many universities" (p. 497).
- Almost all new research is geared towards the World Wide Web, so much so that, under the label of "e-learning", computer-based education has become practically synonymous with web-based education.

- Web-based course authoring and management systems like WebCT or Blackboard are now widely-accepted commercial products that have become the predominant e-learning technology in the training and education fields and the de facto standard for e-learning at the university level. A considerable portion of current research involves conceptualising, prototyping and otherwise exploring methods for improving courseware's functionality and performance.
- While intelligent tutoring systems have enjoyed much less practical success than courseware, they remain the major alternative point of reference within computer-based learning research. Moreover, a large part of efforts aimed at improving the learning features of web-based courseware draws on the techniques and lessons that have been learnt from several decades of intelligent tutoring research.

3.3 Evaluation criteria

3.3.1 A focus on extramural study in the public sector

In this thesis the focus is upon extramural study. Extramural study involves the delivery of university-level degree and diploma programmes directed to students unable to attend normal classes. These programmes are an extension of internally-taught courses, and are organised around the correspondence schooling tradition of individual home study. As a public sector activity, the priority is upon ensuring that all those wishing to study can do so on an equitable basis, ahead of the commercial criteria that necessarily govern for-profit distance education.

While extramural learning offers advantages of anywhere, anytime study, there is no requirement to outperform traditional classroom-based teaching. Primary emphasis is on delivery to individual students. However, extramural courses also try to incorporate more of the collaborative and interactive benefits associated with face-to-face teaching through the use of email and telephone communication and the organisation of contact courses, especially for papers with more of a practical dimension (Prebble, 2001b).

3.3.2 End-user requirements

As an end-user computing application, good design of computer-based distance education software means treating the users as part of the system and designing their requirements into the interface (Moran, 1981). There are at least three types of end users of computer-based distance education systems: teachers as course authors, teachers as course administrators, and distance students.

As a learning application the most important end user of the system is the student. Therefore, while the requirements of all three user types need to be integrated into the design, primary emphasis should be placed upon the student view. In general, the evaluation of software systems from the end-user's perspective aims to establish that it is both useful and usable (Preece, 1994, p. 603).

Evaluation typically proceeds along two dimensions - functionality and usability. When evaluating a computer-based extramural study system from the student end-user's view, it is proposed to add a third dimension - accessibility.

3.3.3 Functionality

The evaluation of a software system in terms of its functionality is concerned with what the system does and how well it does it. A computer-based distance education system is a vehicle for teaching. Teaching, in turn, aims to make student learning possible (Laurillard, 1993; p. 13). Thus, evaluation of the functionality of such a system is primarily concerned with its pedagogical effectiveness, with students as *learners*. Poor computer-based distance education may be little more than self-directed learning in which the students muddle through by themselves as best they can. Good computer-based distance learning, on the other hand, should be permeated by conscious learning outcomes determined by the course author. Therefore, the student user's requirements cannot be defined by the student alone. They must incorporate the pedagogical functions and features desired by the teacher user as well.

"There are many different kinds of learning theory. Each emphasises different aspects of learning, and each is therefore useful for different purposes. To some extent these differences reflect a deliberate focus on a slice of the multidimensional problem of learning, and to some extent they reflect more fundamental differences about the nature of knowledge, knowing, and knowers, and consequently about what matters in learning" (Wenger, 1998, p. 4). When teaching an internal paper, a lecturer has a considerable degree of flexibility in determining the learning dimensions and styles they consider to be most appropriate for the subject and students they are teaching. They may incorporate both passive and active styles of learning, which may involve students studying alone or working with others.

Bork asserts that university education has traditionally relied upon the more passive modes of learning: "Within almost every university in the world, the major courses of the first 2 years are delivered almost entirely by a lecture-textbook method, often with large numbers of students... [This] has been the dominant method since the 1700s,

both in schools and universities" (Bork, 1996, para. 15).

Nevertheless, even the undergraduate student has opportunities to participate in workshops and clinics, seminars and tutorials, laboratory sessions and library browsing, and formal and informal group work, as well as to seek out individual help from lecturers and tutors. This creates a richer and more multi-dimensional learning experience – a community of learning – than is possible by the "lecture-textbook method" alone. At least six broad dimensions to this community of learning can be identified:

1. *Learning by textbook*, in which students learn by studying a prescribed set of readings in a prescribed order, as with a course textbook. This is also the basic form of distance correspondence learning.
2. *Learning by lecture*, in which learning material is presented by the teacher, often with accompanying notes and illustrations on a board or slide, which the student must absorb linearly, as in a lecture theatre;
3. *Learning by exploration*, in which students learn by following their own lines of thought through the learning material, as they might browse a library book, or other reference and demonstration materials;
4. *Learning by collaboration*, in which students learn by discussing and working on problems together, as they would in a seminar, or through informal discussion and joint work on an assignment;
5. *Learning by doing*, in which students learn by experimenting and practical problem-solving, as in a laboratory, or by writing an essay or arranging a piece of music; and
6. *Learning by tutorial*, in which a tutor interacts with a student individually to provide personalised coaching.

Furthermore, experienced teachers pragmatically adjust their teaching methods and content to suit the subject and the students. "Applying standard recipes to all students in all contexts simply does not work," explains one educational expert (Marland, 1997, p. 5). It would therefore be an advantage for computer-based extramural study, for the system to have the flexibility to accommodate multiple approaches to teaching and learning.

If undergraduate university study has been predominantly two-dimensional, then the traditional correspondence method of extramural learning has been one-dimensional, based around the textbook. To match the correspondence method, therefore, a

computer-based system needs only to provide all necessary educational material to support anytime, anywhere learning by prescribed textbook.

But if extramural e-learning is to offer an advantage over correspondence-based study, then it needs to support other dimensions of university learning that have been identified here so as to provide more individualised, interactive and collaborative study opportunities converging towards those available to the internal student. This in turn means realising more of the nine potential ways networked computers can enhance extramural education as discussed in the previous chapter (Section 2.3). By utilising these capacities, the system should enable teachers to integrate a wider range of teaching strategies and styles into their courses and facilitate more of the multi-dimensional learning associated with an on-campus university education.

3.3.4 Usability

There are two types of learning associated with computer-based education – learning the course subject matter, and learning to use the courseware itself. The latter should obtrude as little as possible upon the primary learning process. This means designing the learning system for usability. The functionality should be presented in such a way that it is readily usable by the student. The focus here is on students as *end users* of a computer system.

In the first instance, the usability of a computer-based distance learning system needs to be evaluated against well-established criteria applicable to any interactive software application, such as the "eight golden rules" of Shneiderman (1998; pp. 74-75) or Nielsen's usability heuristics (Nielsen, 1994).

Usability takes on added importance in the extramural environment, where students may be inexperienced computer users working in isolated conditions without the support network available to internal students. The problem of overloading computer systems with superfluous features and functions that overwhelm the inexperienced user (Hurley, 1998) takes on added weight in this context. Under these circumstances, it is necessary to take a holistic approach which considers the usability of the entire computer environment – hardware, operating system and the educational software itself.

An extramural e-learning system must not only meet all the distance student's functional requirements but must overcome their instinctive distrust of being taught by a computer (Yellen, 1999). This places greater importance on transparency of system actions and retaining the locus of control with the user wherever possible. Nowhere is

this more important than in the provision of end-user training and help. Here, the "just-in-time, just enough" training approach (Lennox, 2000; Donovan et al., 1999) would seem to be the most appropriate and least obtrusive.

3.3.5 Accessibility

On top of the usual stresses of university study, the distance student must confront issues such as:

- physical isolation from learning materials, teachers and other students;
- dependence upon unreliable public communication networks;
- complications imposed by the student having to provide their own hardware and software; and
- additional disruptions and distractions posed by living two lives – studying and working, or studying and raising a family, for example.

The situation of the extramural student studying from home via the Internet is roughly analogous to that of the remote worker telecommuting from home via a virtual private network. The sharpest difference, however, is that the remote student must function without the technological, financial and peer support given the remote worker by his/her employing organisation.

There are unique aspects to the functional and usability requirements of computer-based extramural study that are best summed up by adding a third dimension, *accessibility* – access to learning materials wherever and whenever required; access to help wherever and whenever required, and access to the necessary hardware and software.

An extramural learning application needs to address issues such as difficulties in communicating with other students – and in locating and accessing resources – across a public communications network, and variations between students' computer platforms. The requirement should be that all the functionality of a distance learning system is as accessible to a student studying from a more remote locality, e.g. a hill-country farm, a rural village in a developing country, or an army post abroad, as it is to a student working in an on-campus computer laboratory.

3.3.6 A student-centred learning environment

In this section some criteria have been enunciated for making a qualitative evaluation of a computer-based system regarding its suitability for extramural study. In essence, these criteria are that it should sufficiently fulfil functionality, usability and accessibility requirements so that the learning circumstances and opportunities of extramural students more closely parallel that of campus-based learners. If an extramural study programme is to be delivered by computer, then the entire system – hardware, operating system, network services, and courseware – should be conceived of, and be assessed as, part of the learning system. It should provide a holistic environment that promotes student study and learning.

3.4 Assessment of web-based courseware

The initial phase of the research reported in this thesis involved a wide-ranging review of educational software. This included a more detailed investigation of about thirty systems, based upon perusing articles and papers by the developers, considering comparative reviews, visiting promotional web sites, and exploring sample software (Appendix A). This review quickly established that there existed a vast array of programs aimed at coaching very specific skills, often at a quite elementary level. But in terms of on-line teaching at the university level, the options were much narrower. If one-off prototypes are ignored, then the choice reduces down to a custom-built web site or one built using any of a small number of established commercial course-authoring products.

Some courseware systems such as Top-Class (Lennox, 2000) have been specifically developed for the authoring and management of industry training programmes. Others like WebCT (Goldberg, 1997) and Blackboard/CourseInfo (Kubarek, 1999) have been developed through university research projects and were initially geared towards supporting internal university courses. All have been promoted as suitable vehicles for university distance learning. In this section, their suitability for extramural learning is critically appraised using the criteria developed in Section 3.2.

3.4.1 Courseware functionality

How far does web-based courseware enhance extramural study? As a practical teaching tool, it realises several of the learning features and functions potentially supported by networked computers, although with some additional limitations in the extramural environment.

The main courseware products have been extensively reviewed and compared on web-sites, in magazine articles, and in research papers (Appendix A). Brusilovsky and Millar present a useful overview in which they classify existing courseware from a number of different perspectives. From the *delivery* point of view they distinguish three levels of sophistication – from a base level that relies on static HTML and media pages with static links, through to those systems that store course information and content in a database and generate most of the pages on the fly. A system with a database core provides more functionality and is more easily managed (Brusilovsky et al., 2001). More recent experience from Massey University's use of courseware would suggest that, while more use is being made of database technology to manage individual students, most learning material is still being presented as a labyrinth of static HTML documents, through which the student must navigate.

Courseware is typically installed on a central server and delivers course content as web pages viewed through a multi-platform browser on the user's computer. This places greater restraints on functionality at the student interface compared to applications written for a single platform or as a standalone system. Additional features and functions can be enabled by providing special software which is downloaded with the web page, or separately, and executes on the user's computer.

Perhaps courseware's strongest contribution has been in helping teachers to author and update learning material. Initially, web-based university courses were do-it-yourself (DIY). Keen faculty members had to learn to build web sites from scratch, make the best of authoring tools included in web browsers, or get what help they could from university computing support services. Together with course email lists, they provided on-line resources and support for internally-taught courses and external correspondence courses. While in practise most DIY web sites were quite rudimentary, based upon static HTML pages, in theory the only limit on their features was the programming abilities of the developer and what can be downloaded via a web browser to the student's computer. The Open University of the United Kingdom warns that sites built on a DIY basis "are like nuclear power stations: they might seem to be a great idea at the time, but maintenance can be prohibitively expensive" (Watt, 2000, para. 5).

One approach to reducing this "nuclear power" risk, represented by the CATALYST project at the University of Washington, is to provide a support service including just-in-time training and a suite of tools for faculty wishing to develop course web sites (Donovan et al., 1999). Another is to extend to the Web the concept of course authoring software, represented by products like Macromedia's Authorware, that had been mainly developed to author and deliver CD-ROM-based occupational training.

Courseware tools represent an advance for teachers over DIY through: providing templates to help them structure their course material, and to reduce or eliminate the need for programming skills; integrating communication functionality directly into the course structure; assisting staff with enrolment and other administrative tasks; and providing feedback, through online testing and tracking of students' progress through their courses.

Standard courseware integrates various media like video and text into a single browser-based environment. Its server-centred architecture means that it cannot easily integrate materials stored at different locations into a student's course, although it can provide links that the student can pursue independently.

Generally, interactions with the student are quite limited, relying heavily on the presentation of learning materials to be read or watched, in combination with internet-based discussion tools.

To Brusilovsky (1999), courseware remained in essence multimedia-embellished HTML pages. Nielsen (2001) considered it no more multi-dimensional and interactive than its 1970s PLATO predecessor. Jesshope and his colleagues wrote that, until web-based systems could interact dynamically with the individual student, all they would do is provide a repository of information which replaces the disk space on stand-alone computers or hardcopies in traditional correspondence courses (Jesshope et al., 2000).

Interestingly, Nielsen, a usability expert, warns against trying to reproduce the textbook electronically. "Large amounts of text just do not work very well on a computer screen simply because it is painful and slow to read" (Nielsen, 2001, para. 5). He considers that a book is better for transmitting large amounts of information.

The courseware teaching systems characteristically cover their subject matter with a broader brush than adaptive systems do. Murray (1999), as a result, criticises commercial courseware for its shallow representation of content and pedagogy. The student is presented with the material and then left to pursue things further on his/her own.

Characteristically, courseware is neither adaptive nor adaptable from the student user's point of view. From the system administrator's viewpoint, commercial courseware such as Top-Class (Lennox, 2000) or GENTLE (Dietinger et al., 1998) has the capacity to restrict individual student's access to some learning materials, or determine the order in which they may view it, in line with the requirements of education providers.

Brusilovsky considers it a challenging research goal to develop advanced Web-based

educational applications that offer some amount of adaptivity and intelligence. These features are especially important for distance learning applications "since distance students usually work on their own (often from home). An intelligent and personalised assistance that a teacher or a peer student can provide in a normal classroom situation is not easy to get. In addition, being adaptive is important for Web-based courseware because it has to be used by a much wider variety of students than any 'standalone' educational application...[C]ourseware that is designed with a particular class of users in mind may not suit other users" (Brusilovsky, 1999, para. 1).

In contrast to the individualised character of adaptive tutors, however, courseware systems can more easily accommodate collaborative study. WebCT, for example, is intended to facilitate communication and collaboration (Goldberg, 1997).

Support for multi-dimensional learning

While web-based courseware is not explicitly built around any particular approach to learning, in practise it relies heavily on the more passive lecture and textbook methods of study, what Brusilovsky and Miller call learning by reading and watching (Brusilovsky et al., 2001). Even then, learning by lecture is generally only supported in the sense of being able to store and display multimedia files. A course delivered in this ways offers very little advantage to the extramural student over the study outlines and texts of traditional correspondence learning. In fact, Cheung (2001) provides evidence that students may prefer hardcopy-based study to this form of e-learning.

Of the more active dimensions of university learning, only learning by collaboration is realised to any extent. Courseware facilitates co-operation and communication between students studying alone from separate locations. However, the anywhere, anytime requirement of extramural study means that the utility of synchronous collaborative features such as conferencing and chat may be severely restricted. Many extramural students must rely upon the asynchronous functions like email for collaboration.

Courseware does not support one-on-one tutorial learning beyond electronic correspondence with a human tutor.

None of the systems reviewed matched the potential of the textbook for exploratory learning. While a course web-site allows the freedom to explore, it is harder to navigate around, slower to leaf through, and easier to get lost in than a book. Some courseware systems have features built into them which militate against learning by exploration, by

imposing a predefined learning path upon students or restricting access to particular learning materials (Lennox, 2000; Dietinger et al, 1998).

3.4.2 Usability concerns

Whatever its functional deficiencies, a more fundamental barrier to the successful application of courseware to extramural learning is concern about its usability. Courseware does not meet key usability requirements for extramural study such as avoiding function and feature overload and providing just-in-time and just-enough user help.

For the extramural student, especially the less experienced computer user, using courseware is a far from trivial task. This is not a problem of the learning system alone. Even where the student interactions with the courseware have been carefully thought out, or the student's path through course material is constrained, it remains a component of a larger computer environment. The courseware interface is presented within the web browser interface, which in turn is presented within the operating system interface. Accessing particular learning materials may cause new windows to be opened. The result is a visual clutter in which the student can become lost, confused, or distracted (Figure 3.1).

The online help system poses the same layers of complexity to the user. Each open application provides its own help database which the user must try to query. There is a real likelihood of their being overwhelmed by too much information.

In short, courseware adds additional layers of complexity to a computer system whose complexities the extramural student may already be struggling with. It requires a level of computing experience beyond what can be assumed of an extramural student working alone at home.

Courseware reflects a more general problem in e-learning that the design of the student interface has not been prioritised (Kruse, 2002; Murray, 1999). Kruse considers the interface between students and computers to be the "single most neglected topic in the field of e-learning" and a major reason for students expressing a preference for classroom-based over computer-based instruction (Kruse, 2002, para. 1). Bork criticises the tendency of educational software developers to incorporate the standard Windows and web browser interface features and functions, such as menus and tool bars, even though these items have no connection with the learning task. They amount to "visual garbage" which distracts the student from this task (Bork, 2001a).

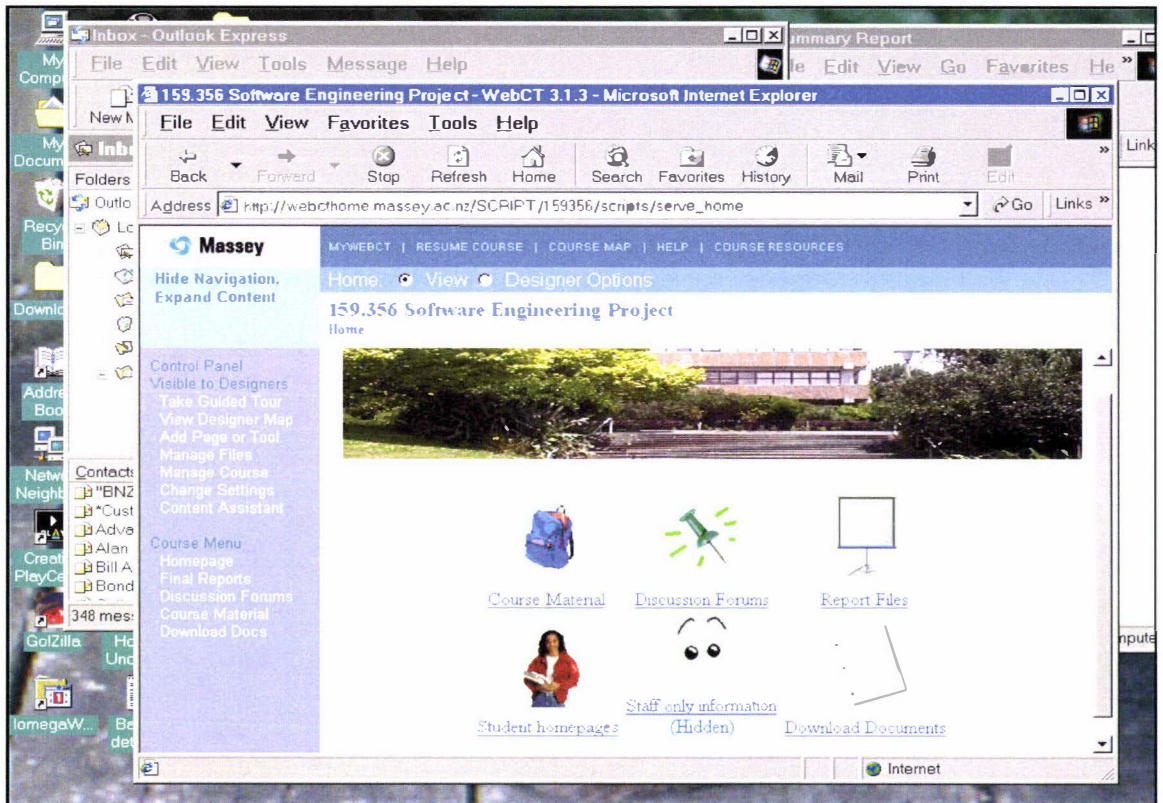


Figure 3.1: Courseware environment marked by visual clutter and distractions.

Because of their promotion as authoring and management tools, courseware systems emphasise the requirements of educational organisations and businesses, ahead of the requirements of students. From this perspective, the most important end user – the distance student – is viewed as the *object* rather than the *subject* of the system, or the back-end rather than the front-end.

3.4.3 Inaccessibility

The essential requirement for a wholesale shift from correspondence-based to computer-based extramural study is that all the learning material be available anywhere, anytime. Courseware is unable to achieve this.

Theoretically, courseware is *accessible* from every home or workplace with a telephone connection. In practise, however, the vast majority of households in the world do not have telephone service. Even among those that do, the quality of the service for those living outside the most developed urban areas may mean painfully slow and unreliable Internet connections (2.3.1).

Poor communications infrastructure is fatal for courseware, because it relies upon the learner maintaining a live connection with the server on which the course is stored. First, some extramural students may not be able to download their lessons. Second, a

slow response time frustrates and distracts the student and inhibits learning (Nielsen, 2001). Third, a slow connection restricts the content of the course. If the course makes extensive use of the multimedia capabilities of the technology, as some proprietary applications like the Cisco networking courses (Cisco, 2002) do, then it simply will not work under these conditions. Fourth, it gives an unfair advantage to those students with access to high speed communications technology.

Historically, *extramural university study* has existed in the first instance to provide tertiary educational opportunities where they would not otherwise exist, viz. in developing countries and in the more remote parts of developed countries. But it is precisely in these areas where web-based courseware is least effective.

3.4.4 Summary

From this review it is concluded that web-based courseware systems have been primarily designed as learning management systems to support company training programmes or internal university courses. The emphasis is upon cross-platform capabilities, and upon the requirements of education and training providers such as authoring and course administration. They provide a complex, multimedia environment that requires a fast and reliable network service and assumes that user training and support will be at hand when needed. Neither can be assumed in the extramural environment.

Courseware is a useful option by which experienced computer users with good Internet access can study a course without having to sit in a class or lecture theatre. This form of distance education is often called *flexible learning*.

As a tool which enables teachers to author e-learning material without recourse to a skilled programmer, and which facilitates collaboration and communication among students, web-based courseware has contributed greatly to the explosion in online education.

From the perspective of the extramural student, courseware's limited interactivity and individualisation offers little advantage over email-and-book-based extramural study. Moreover, courseware's management from a central web server means that learning materials and features may not be accessible by the distance student "anywhere, anytime" because of unreliability and slowness in the telecommunications system. And the complexity of the computer environment in which the user interacts with the learning material, and insufficient attention to interface design, poses usability concerns that further limit the learning functions and features available to the extramural student.

3.5 Recent developments in e-learning

The growing interest in web-based education and training is spurring new research and development into improving systems for the authoring and delivery of online learning. It has also prompted some old ideas to be revisited.

This section reviews some significant trends in this research that have a bearing on improving the functionality, usability or accessibility of e-learning software for extramural study.

3.5.1 Anywhere, anytime study

One-way Broadband

Cook (1998) writes that the biggest problem in providing Internet access in the developing countries is “that of ‘the last mile’ (or first mile) linking between the towns and the villages” (p. 22). Advances in communication technology – based upon variations of wireless microwave or satellite technologies – have opened up new possibilities for widening the bandwidth beyond what the local telephone infrastructure can offer for delivery of learning material into rural areas. However, this is by no means a solved issue, with many of these new technologies also failing at “the last mile”, even in the developed countries.

One line of attack uses wireless networking technology to provide a 2-way internet service, which is typically several times faster than that available over standard copper wire lines. A wireless modem has the advantage that it can be used anywhere within a wireless network's coverage, although its bandwidth is below that recommended for streaming audio/video, very large downloads or viewing graphic-intensive Web sites (NZWireless, 2004).

Unfortunately, node-to-node links in a wireless network require a line-of-sight connection over a relatively short distance. This is difficult and expensive to provide in remote and hilly terrains.

Hand-held wireless systems using mobile cell phone technology have also been explored (Jo et al., 2001, Boada et al, 2003). Mobile phone networks face similar performance issues as wireless networks. Moreover, they exacerbate the usability issues relating to e-learning because of poor performance and inadequate user interface (Quinn, 2002).

Satellite transmission technology is opening up promising new possibilities for high bandwidth delivery to individual distance students as telecommunication and media

companies drive research into linking the Internet with digital television services (Blyth, 2001). Two-way broadband satellite transmission is an important technology that is used by learning centres and schools. Satellite communications have the added advantage of being a global technology that does not require a well-developed local communications infrastructure. Thus satellite broadcasts have been used for some years to deliver distance education on an international basis to learning centres in Africa, Latin America and elsewhere in the developing world (Cook, 1998, p. 29).

For individual users, satellite *reception* is as straightforward as satellite television reception. Satellite *transmission* is more problematic and expensive¹. To be cost effective, home users would have to be linked through some other technology such as a landline to a shared transmitter. Such transmitters are not generally available in rural areas. Moreover, links to them face the same performance issues as for rural telephone or wireless communications.

A more practicable approach for the individual extramural distance student would be to employ a hybrid approach which uses satellite reception for rapid downloading of learning material and a terrestrial link to the transmitter for receiving communications from the student machine, including system information necessary to maintain the connection and ensure the integrity of the data transferred. Early efforts along these lines managed download speeds four times faster than a dial-up (Arora et al, 1996). Pioneer NZ internet service provider, IHUG, has since developed a commercial version of this technology providing download speeds of up to 40 times that of dial-up connection. In this way, rural customers could be offered a one-way broadband service at an affordable price that uses a standard dial-up internet connection to link to a central satellite transmitter, and special satellite reception hardware and software to receive downloads from the Web via the transmitter (IHug, 2004).

While video-conferencing over the Internet is technically straightforward, the results for the rural home user, employing standard copper wire connections, mirror those described in a computer journal as "teleconferencing where you have a Webcam on your PC and look at a postage stamp-sized fuzzy image changing twice per second" Honeyball (2002a, p. 172). However, integrating satellite broadcasts into e-learning not only provides a more effective medium for streaming video to a rural home computer, but opens up possibilities for using parallel television broadcasts to complement computer-based learning materials.

“A truckload of tapes”

To ensure that a student's course material is accessible anywhere, anytime, Gehne et al. (2001) advocate distributing part or all of the system's functionality to the student's machine, so that the system will work off-line.

The TILE system duplicates web server functions on the student's computer, and requires that the student machine and the central server be periodically synchronised online (Gehne et al., 2001). If necessary, teaching materials can then be distributed by mail using portable media like CD-ROMs. Gehne et al. cite computer pioneer E.E. Dykstra ("never underestimate the bandwidth of a truck-load of tapes", p.2) in support of this approach.

A similar tack is taken in Dietinger et al. (1998) and Bork (2001a).

3.5.2 Individualisation

Efforts to individualise educational software systems have proceeded along two axes – system-initiated adaptivity and user-initiated adaptability (Oppermann et al., 1997). The major differences between adaptable and adaptive approaches are listed in Table 3.1.

Adaptation may be implemented across a range of dimensions as illustrated in Table 3.2. A learning system may employ adaptation across all, some or none of these dimensions. Some features, e.g. the help system, may be adaptive, while others, e.g. the interface look and feel, may be adaptable.

In arguing for adaptive interfaces, Benyon and D. Murray asserted that the drawback with the adaptable approach is that “the user must learn functions which are tangential to their main task. Although some routines will only have to be set once, others involve learning specialised commands. Tailoring facilities are typically very general, do not take fine-grained, individual differences into account and do not cater for a user's task needs, either perceived or implicit” (Benyon et al., 1993, p. 198).

Efforts to imbue courseware with adaptive capabilities have drawn upon three decades of experience in the development of intelligent ITSs. One method of developing adaptive web-based systems has been to customise existing standalone ITSs so that they can be accessed from anywhere via the Internet. SQL-Tutor (Mitrovic and Hausler, 2000) and PAT (Ritter, 1997) are examples of standalone tutors that have been successfully ported to the Web. However, the narrow subject focus of such tutors

¹ In an October 2004 personal interview, the principal of a rural primary school reported that 2-way satellite internet had been offered to his school for \$1500 per month. In a February 2001 letter, a satellite

makes them an unsatisfactory choice for teaching whole courses. Efforts have therefore expanded along the lines of:

- integrating tutors and other adaptive learning tools into broader courseware; and
- the development of adaptive systems that can teach whole courses.

Property	Adaptability	Adaptivity
Primary research discipline	HCI	AI
Manner of adaptation	Static	Dynamic
Locus of control	User	System
Flexibility of user view	More	Less
Complexity of user view (cognitive load)	More	Less
Hidden complexity (implementation)	Less	More

Table 3.1: Contrasting properties of adaptable and adaptive computer systems

Dimension
Interface "look and feel" - how user interacts with the system
Sequencing – what part of the course is presented next
Learning style – how the material is presented to the student
Learning level – the degree of difficulty of the material presented
Learning help – what help is presented to the user and when

Table 3.2: Forms of adaptation in learning systems.

Another approach, which is discussed under the subheading of "Collaboration and Help", is to incorporate software agents that collaborate with and assist the user in the learning task.

Interactive tools to foster multidimensional learning

The United Kingdom-based Byzantium project sought to achieve more practical results for artificial intelligence in education by limiting the goal to producing intelligent tutoring tools that extend the scope of a lecturer rather than replacing him or her (Patel et al., 1997). Patel et al. (1997) propose that Web-based versions of such tools be integrated into broader courseware.

Audiograph (Jesshope et al., 1998) and MANIC (Stern et al., 1998) support learning-by-lecture through providing synchronised audio and slide presentations. MANIC supports individualisation by adapting presentation sequencing to suit each student.

A number of web-based tools have been developed, at least at the prototype level, to offer better support to active learning dimensions such as learning by doing, by exploring, by tutorial and by collaboration. But even when they have been successfully prototyped, lack of standardisation makes it difficult to integrate them into commercial courseware systems. The "Model for Distributed Curriculum" (Murray, 1998a) proposed standards by which an e-learning system could dynamically locate relevant tutorials on the World-Wide Web. More recent standardisation efforts have focussed upon reusable learning objects, which will be discussed later in this chapter.

InterBook (Brusilovsky et al., 1998) and Metalinks (Murray et al., 2000) are electronic textbooks which guide students while allowing them to access material in the order of their choice. In this way they provide an environment for students to learn by exploring. This is especially so of Metalinks, which enables the individual student to adapt navigation paths and the presentation of material to their own learning level and preferences.

Adaptive courseware

Adding adaptivity to educational software may come at the expense of other learning goals. For instance, with ITSs the locus of control is typically with the system rather than the user.

Overall ITSs have had a narrower teaching focus than web-based courseware. They have taught a particular problem-solving skill in a well-defined and structured topic. WITS however, is a course-oriented intelligent tutoring environment which gives

students the more coarse-grained treatment aimed at the average student that they would receive from a class teacher (Callear, 1999).

Other researchers have explored ways to build adaptation directly into the Web technology itself so as to produce adaptive courses. Adaptive courseware systems employ a user model and an adaptation strategy to generate an individualised user view. Two early efforts were the Dynamic Course Generator (DCG) (Vassileva, 1997) and ELM-ART (Weber et al., 2001). DCG generates individual courses according to the learners' goals and previous knowledge and dynamically adapts the course according to the learning progress of the student (Vassileva, 1997). ELM-ART is an intelligent, interactive, educational system to teach LISP which incorporates tools that enable students to communicate with each other and with their tutor (Weber et al., 2001).

Both DCG and ELM-ART were developed to teach computing-related subjects. They have not made the transition into broader educational use.

Interbook, an offshoot of ELM-ART, is a tool for authoring adaptive electronic textbooks. Interbook uses a depth-first strategy, i.e. encouraging the learner to demonstrate thorough knowledge of one topic before moving on to another topic.

ITSs are typically not adaptable by the user. An alternative adaptable, breadth-first approach represented by Metalinks will be discussed in the following section.

The customisable (adaptable) interface

Benyon et al. (1993, p. 197) wrote that the practical results from the adaptive approach had been "rather disappointing and problems have proved far harder to deal with than was first expected." Even today, more than ten years after that article was written, adding adaptive features to courseware remains at the level of research prototypes.

T. Murray (1999) and Bork (2001a) argue that individualisation goals can be achieved more simply by paying greater attention to design of the student interface. Murray writes that since basic graphics authoring is considered a solved problem, most ITS researchers have not prioritised student interface design, but have restricted their systems to predefined screens and styles. This "severely constrains the types of tasks and interactions that an ITS can have with the student" (Murray, 1999, para. 28).

Murray and his colleagues (Murray et al., 2000) present evidence from the development of the Metalinks project that adapting the content to the individual student may be better and more simply achieved through deeper design of the interface, keeping the locus of control with the student. They argue that underestimating the importance of interface design (adaptability) in learning systems can lead to

overestimating the importance of artificial intelligence capability (adaptivity).

Metalinks uses a breadth-first approach to allow the student to traverse the course at the level they choose. It was designed to accommodate intelligent software features but was implemented in a less sophisticated version. This version was then evaluated for evidence that students would benefit from adding more advanced features. The researchers were somewhat surprised by the results, which indicated that well-designed interface features and non-intelligent tools satisfied many of the user needs that they expected would have to be addressed by adding intelligent software capabilities. "This work supports the notion that good interface design and passive but powerful user features can sometimes provide the benefits that are ascribed to more sophisticated or intelligent features (which can be more presumptive, controlling, or intrusive than passive features)" (Murray et al., 2000, para. 42).

The authors felt this experience also underscored a more general lesson that inadequacies in the design of the interface or usability of educational software will overshadow any benefits from more sophisticated features it may possess (Murray et al., 2000).

3.5.3 Collaboration and Help

Collaborative agents

Benyon and D. Murray (1993) point to the development of interface agents as a more promising (and less ambitious) approach to imbuing systems with adaptive capabilities. With this approach, interfaces employ agents that can interact with users and with other agents to assist users to accomplish specific tasks individually, or co-operatively across a network. "Essentially agents are adaptive systems, but systems which are specialised and know about only a very small part of the world" (Benyon et al., 1993, p. 200).

Collaborative agents are an adaptation of intelligent tutoring technology to support collaborative forms of learning beyond the simple email, chat rooms and threaded discussion lists of commercial courseware.

Collaborative learning has not generally been the focus of adaptive tutors, which are geared to the individual student. Vassileva argues that while an adaptive system "truly 'cares' for the learner", by tailoring the course to them, it can reinforce the problem of the "lonely learner". She holds that adapting to other's ideas and experiences is part of the learning process, so, "maybe sometimes the learner should adapt, not the

environment?" (Vassileva, 2001, Slide 12).

I-Help (Greer et al., 2001), uses intelligent agents to attempt to overcome the learner's isolation by organising and monitoring computer-based collaboration within a community of learners. I-Help attempts to find the most suitable member of such a community to respond to another member's query. A variant to this approach (Ishikawa et al., 2002) channels questions posted to a courseware bulletin board to other students or teachers currently online to answer.

VINCENT (Paiva and Machado, 1998), which is built around an animated software agent that acts as an intelligent helper to assist trainees to complete their courses, intervenes when students get off-track. Lopez et al. (2002) and Rasseneur et al. (2002) use virtual student software agents to foster collaboration and discussion among networked students.

Natural language querying

Another approach to the lonely learner problem is to provide a means by which students can ask the system questions about a subject and get a meaningful response, even if a human tutor is not available online. The Flexible Structured Coding Language approach enables querying of a database, using a natural language-like syntax, which can return known facts to the student (Heinrich et al., 2001). Heinrich and Kemp propose extending this approach by using artificial intelligence techniques to enable the system to infer additional facts from this database to pass on to students (Heinrich et al., 2000).

3.5.4 Interaction

Specialised languages

Bork criticises the "very limited forms of interaction" possible with courseware using HTML-based browsers (Bork, 2001a, p. 59). He counterposes a graphical scripting language with which experienced teachers can design highly interactive, natural language-based tutorials (Bork, 2001a, p. 64; Bork et al., 1992).

Specialised languages for simplifying the programming of learning material are not a new idea. They were an integral part of the PLATO and TICCIT projects. This approach has now been taken up more widely in the form of developing specialised languages for authoring web-based learning material using the Extensible Mark-up Language (XML). XML provides a cross-platform format for defining a wide range of documents

as text files, using a set of author-defined HTML-like tags. The file can then be parsed by an XML parser and passed to a specialised browser or other application to execute. The result may be a standard web page, a web page displaying specialised content such as mathematical formulae, or an interactive program running on the user's computer (Harold et al., 2002, pp. 3-10).

E-learning researchers expect that using XML in this manner will offer a standardised, multi-platform means of providing interactive and individualised learning material. Gerdt et al. (2002) and Bergstedt et al. (2002) are applications of an XML-based approach.

Standardisation through learning objects

Learning objects are reusable and easily modifiable educational modules designed to be assembled into larger educational units such as activities, lessons, or whole courses. These objects may be distributed across a number of repositories from which they are retrieved and integrated into learning management systems for delivery to the student (Brooks et al., 2003). Through learning objects it is hoped to produce reusable learning materials that are more interactive and individualised than courseware can currently provide.

A range of organisations have been involved in a major international effort to develop standardised, reusable learning objects based upon XML (Tansey, 2003). The Sharable Content Object Reference Model (SCORM) is a suite of technical standards that enable web-based learning systems to find, import, share, reuse, and export learning content in a standardised way built upon XML. This content can then be picked up by learning management systems and course authoring tools that conform to the SCORM standards. The base unit of a SCORM-compliant course is a learning object (web page) authored using specialised tags to define the learning content. Courses are individualised by the course management system through tracking individual learner's progress and sequencing and branching between learning objects accordingly.

SCORM's principal sponsor has been the United States Department of Defense and is strongly oriented towards the provision of skills training within organisations with a large geographic spread (SCORM, 2002).

3.5.5 Specialisation

Bork has been a tireless advocate of providing simpler interfaces for courseware than that typically associated with Windows-based and browser-based applications:

"Nothing should be on screen that is not directly relevant for learning the material at hand. Many years ago we made the operating system invisible for such use. That seems to be a desirable direction for the future. Learning does not need the complexities of today's interfaces" (Bork, 2001a, p. 64).

Bork (2001a) also questions the assumption that the general-purpose computer is the best tool for learning applications. He points out that embedding computers into other more specialised devices is "a common occurrence" these days (p. 61). NIMIS (Hoppe et al., 2000) draws upon the "ubiquitous computing" concept of using "special purpose networked devices (rather than uniform computing equipment) embedded in natural environments, be it at home, at work, or in educational settings" to develop a "'computer-integrated' primary school classroom" (para 1).

Embedding computers in everyday environments does not in itself resolve usability issues. There are some good examples such as the Smart Drive washing machine² where, through careful interface design, embedding a computer in an appliance has improved usability. However, Cooper and Saffo discuss a wide range of cases where embedding a computer has added complexity to everyday tasks such as using a camera, in part because the computer makes possible the addition of a myriad of unneeded features (Cooper et al., 1999).

3.5.6 Conclusion

In this section aspects of recent research into improving the performance of e-learning systems have been reviewed. Perusing this research, three broad strategies stand out which contrast with the approach taken by conventional web-based courseware and show promise in addressing the functionality, usability and accessibility issues confronting computer-based extramural study. These are:

1. *Specialisation vs. generalisation.* That special-purpose computer tools (interfaces, programming languages, hardware...) offer a way of providing a simpler and more usable learning environment than general-purpose ones (web browser, HTML, PC...) by helping to render the computer invisible to the learning process.
2. *Adaptability vs. adaptivity.* That user-oriented interface design, incorporating customisable and collaborative strategies, can more easily and simply achieve many of the individualisation and interactivity goals of adaptive systems, while maintaining the locus of control with the student.

3. *Localised vs. centralised functionality.* That by placing more of the system's functionality on the student's computer and then using the internet or alternative media to update the student's computer, the greater functionality associated with standalone systems, and the collaborative benefits of networked computers, can be incorporated into an e-learning system that better meets the anywhere, anytime requirements for extramural study than server-centred systems.

Of particular interest is whether, by combining some or all of these strategies, a workable means can be found for meeting the requirements for extramural e-learning.

3.6 Designing for extramural e-learning

In this section some important lessons for designing extramural e-learning systems as learning environments are discussed in the framework of the literature reviewed.

3.6.1 Prioritising the student interface

In this chapter it has been emphasised that there are special user requirements for computer-based extramural study that cannot be subsumed under those for e-learning in general. It is especially important that all the system's learning functions are usable and accessible by an extramural student studying in less than ideal circumstances, e.g. an inexperienced computer user working from home in a remote district with substandard telecommunications. Developers of any computer-based learning must consider students both as learners and as users, that is, they must design both for form and content (Smulders, 2003).

Poor usability inhibits learning and may prevent it altogether. A review of the literature on computers and education supports the assessment that interface issues have received limited recent attention from educational software researchers.³ Research has been focussed upon the student as learner, at the expense of the student as user, for a range of reasons. These reasons include researchers' focus on the learning process and on developing and evaluating learning theories, and an assumption by e-learning researchers that interface design is a solved problem and therefore less important than the learning technology itself. It has been reinforced by the emphasis of courseware on the authoring and management requirements of educational and training providers as the front-end of the system. In this sense, courseware emphasises reusability over

² <http://www.fisherpaykel.co.nz/laundry/washers/washers.cfm>.

usability.

Other factors inhibiting usability include the interface limitations and the complexities of the web browser environment itself, and a more general design problem of software engineers' interfaces tending to emphasise their own domain (i.e. how the system works) over the end user's task domain (i.e. what it actually does) (Gentner et al., 1990).

The dominance of the e-learning field by web-based courseware flows from its utility as a practical course authoring and management tool for instructors. Its strength lies in its holistic (whole course) and reusability approach to computer-based education, in contrast to the more restricted domains and one-off character typical of the more adaptive and interactive systems.

However, courseware's limitations in meeting the functionality, usability and accessibility requirements of extramural study, make it less than ideal as a candidate for replacing traditional correspondence-based courses. While a number of these limitations have been at least partly addressed at the research level, the results from this research have yet to be successfully integrated into real courseware products. Moreover, some of the concerns seem inseparable from the technological limitations flowing from the multi-platform, thin-client architecture of the Web itself. The ubiquity of the Internet has encouraged researchers to accept the graphical browser interface as *the* basis of the interface between the learner and the learning system. In seeking to meet the requirements for extramural e-learning, therefore, it may be necessary to look beyond the technology of the World Wide Web.

Nielsen (2001) emphasises that online learning is inherently not very motivational and not as effective as studying on campus, and that a textbook works better than a computer for presenting large amounts of information for students to read. Computer-based extramural learning should complement the textbook and extend the teacher, and add new multi-dimensional capabilities a book cannot provide. It does not need to replace them. Extramural study is an adjunct or extension to a university – with its "buildings and grounds" and its "body of students and faculty" – rather than an alternative to it.

The introduction of internetworked personal computers has shaken up education in all its forms, and will continue to do so. Learning theory research which seeks out new

³ For instance, of the more than 220 papers accepted for two 2001 conferences on computer-based learning, only *one* is focussed upon the issue of the design of the learner interface (Lee, 2001; Kinshuk, Jesshope and Okamoto, 2000)

ways that computers can improve learning outcomes in comparison to conventional teaching methods are an important area of scientific endeavour. Extramural e-learning researchers, however, do not need to be that ambitious. Their goal need not be to outperform the classroom, but only to find ways to outperform correspondence-based and courseware-based study. From this perspective it is an advantage if a system supports a more pragmatic and flexible approach that can accommodate multiple learning paradigms and styles.

Effective extramural e-learning does not require the wholesale development of new technologies. At a minimum it means finding workable ways of utilising existing technologies so that they are usable by, and accessible to, the extramural student. It means recognising that, while meeting the learning needs of the student are paramount, this will not be possible without also prioritising the requirements of the student as a user. From the standpoint of the extramural learner, the student view *is* the learning system.

3.6.2 Adaptation in a learning environment approach

A learning environment approach to computer-based distance education implies that the system can provide some level of adaptation to the individual learning task and/or student. Some of the areas in which individualisation may be appropriate include:

- The adaptation of content to the individual learner e.g. what topics and sub-topics are presented to the student, and at what level of complexity.
- The adaptation of the form in which the content is presented to the individual learner e.g. as a lecture, tutorial, or a text to be read.
- Support to the student in terms of help and training in the use of the system itself, help in understanding the subject matter; and assistance in searching the entire network for related subject matters.

However, it may not always be desirable or feasible to individualise learning material. A considerable portion of conventional university education is, as Callear (1999) put it, coarse-grained and aimed at the average student. And the design should accommodate the point made in Vassileva (2001) that, from a pedagogical point of view, sometimes the learner should adapt in order to facilitate collaborative work and discussion between students.

3.6.3 Guidelines for designing extramural e-learning environments

The following principles for designing computer-based distance learning systems as interactive and collaborative learning environments have been synthesised from this review of computer-based learning research:

1. *Identify the users clearly.* Systems with a focus on ease of course administration, on supporting internal papers, or providing in-house training, may have quite different requirements from those oriented to supporting university-level distance learning.
2. *Prioritise interface design.* Designing as a multi-dimensional learning environment means focussing upon the content, interactions, presentation styles, and user support at the student interface. Treat the student view as the front end of the system and design from there. This requires designing the interface as a learning environment, rather than as the student interface to a teaching application. A learning environment integrates all the functions and features that support the student's learning tasks. A specialised interface may be more effective for learning than a general purpose one.
3. *Recognise the university itself as the most important learning community.* The goal should be to extend the reach of the university as a community of learning rather than replace it, simulate it rather than compete with it. This means emphasising collaborative, interactive and multi-dimensional learning functions, which complement the roles of the teacher and the textbook rather than replace them.
4. *Design as an information system.* A computer-based distance learning system should not be conceived of as a single piece of software but as an information system incorporating the users and their requirements. In an information system approach, automation (computerisation) is not assumed to be better. A decision is made on what to computerise and what not to. Some aspects of a course may be better implemented using alternative technologies or a human teacher.
5. *Design for reusability on three levels - the programmer, the course author, and the student.* For an extramural e-learning system to provide a practical alternative to the existing courseware products, it must match their utility for authoring, managing and delivering multiple courses of study. In addition, it should be able to readily integrate new and improved learning technologies.
6. *Avoid making presumptions about implementation technologies.* Information technologies which work for computer-based training programmes, flexible learning,

and supporting internal university courses may not best meet the special requirements for extramural e-learning. Other possibilities besides the ubiquitous web browser should also be considered as the basis of the student interface.

7. *Enable adaptation to the individual learning task and/or student.* This includes how material is presented, help in the use of one or other system features, querying of subject matter, and assistance in locating supporting material. However, it may not always be desirable or feasible to individualise learning material. Providing an online human tutor may be preferable. And sometimes the learner should adapt to facilitate collaborative work and discussion between students.
8. *Evaluate for functionality, usability and accessibility.* All the system's functions and features should not only work on high-speed urban communication networks, but should be accessible, and tested, in worst case scenarios like that of the farm-based rural student. Multidimensional features should be prioritised ahead of multimedia ones. For the distance student, multimedia enhancements may be nothing more than a form of bloatware, which inhibits accessibility.

3.7 Concluding the literature review – a hypothesis

Chapters 2 and 3 summarise an extensive review that has been conducted of the literature on distance education and computer-based learning. The framework of this review was the research question: "Is there a universal way to apply information technology to distance education, so that it offers the student an advantage over traditional correspondence lessons?"

The breadth of the domain reviewed, and the diverse research agendas, requirements and technologies encompassed by it, underscored the necessity to narrow down the domain of the research problem to a distinct subsection of the broader e-learning field – university-level extramural students within the public education system. The potential benefits (and the downsides) for extramural study of computer-based learning systems were summarised and some criteria for evaluating their suitability were enunciated.

The mainstream computer-based learning technology - web-based course authoring and management systems – was evaluated against these criteria. The areas where courseware fulfilled, or failed to fulfil, this potential were noted, and major unsolved problems identified. The various solutions that have been advanced to address these problems have been reviewed, including the adaptive and artificial intelligence-based technologies which influence much of the research in this field.

The key requirements for a successful computer-based learning environment for extramural students have been synthesised. Such an environment needs to meet additional functionality, usability and accessibility requirements in order to accommodate scenarios such as inexperienced computer users, students working alone from remote locations, students with older machines and/or poor communication infrastructure, and courses where computer skills are incidental to the content. It needs to be simple and usable enough for a relative computer novice to learn without direct supervision; it should provide some ability to adapt, or be adapted, to an individual student's learning needs; and it needs to work as well in the context of slow and unreliable communications infrastructure as on the information superhighway.

The central theme of this chapter has been the need to more clearly identify the specific needs of the university distance student, and to design computer-based distance learning systems as learning environments that meet those needs. This means shifting the weight of the system to the student end by designing it as a *learning* system rather than as a *teaching* or *course administration* system. This approach places a greater emphasis on interface design than is to be found in most computer-based learning literature. It means starting from the requirements of the user rather than from the constraints of current technology.

Fulfilling the requirements for extramural e-learning would enable a computer-based system to supplant the correspondence system as the basis for all extramural study. However, to find a workable way to achieve this some outstanding issues need to be resolved satisfactorily, above all the accessibility and usability limitations of courseware.

From a review of more recent e-learning research it is concluded that very little has been *specifically* directed to identifying and solving the particular challenges of computer-based extramural university study. None of the working and prototype learning systems reviewed, even where they offered significant improvements over commercial courseware, met sufficient of the key requirements to be considered workable in this domain. However, several themes have been identified in this research, which suggest possible ways to fulfil these requirements. These themes point outside the predominant framework for e-learning research and development by emphasising student-centred strategies of specialisation over generalisation, adaptability over adaptivity, and localisation over centralisation.

Drawing upon the review, it is hypothesised that, by combining these three strategies, and following the guidelines for designing an extramural e-learning environment, a

workable way can be found to fulfil the key requirements for computer-based extramural study. In the next chapter a prototype system is conceptualised through which this hypothesis can be tested.

Chapter 4

Conceptualisation of an extramural e-learning system

The previous chapter concluded with the hypothesis that a practicable way can be found to provide the key requirements for an extramural e-learning environment, by combining three strategies – specialisation over generalisation, adaptability over adaptivity, and localised over centralised functionality – and by observing a set of design guidelines synthesised from the literature review. The first step towards evaluating this hypothesis is to conceptualise, in broad outline, a learning system based on these principles. In this chapter a conceptualisation is presented of an e-learning system as a specialised computer for learning.

First an overview of the conceptual design is presented. Then each of the key aspects of the design is discussed in more detail.

Only once the system has been conceptualised is the technological platform for a prototype considered. No assumptions are made about implementation technologies. Therefore, discussion of more specific issues concerning the technical framework for implementing a learning computer is postponed to Chapter 5.

4.1 A computer for learning

An extramural e-learning environment is conceptualised as a specialised computer for learning, in which the three underlying strategies are embedded. The starting point for this idea is an observation by Bork that: "In the developed countries the general-purpose computer will still be widespread for learning, but for many situations a computer built just for learning, a learning appliance, will furnish a new paradigm, much cheaper and simpler than today's personal computer" (Bork, 2001a, p. 64).

The idea of the learning computer is to provide a minimalist, specialised environment that integrates all the functionality an extramural student needs to successfully complete a course, and only that functionality. And this functionality will work where the infrastructure is less than optimal for multi-media networking. It is a simplified system which a relative computer novice can quickly master, thereby allowing it to rapidly

become invisible to the learning process. It is an adaptable environment that a student can customise to suit their learning preferences.

In place of the specialised hardware of Bork's learning appliance, the learning computer is conceived of as a software system implemented over a general-purpose personal computer, in combination with the necessary support applications at the university end. This means that it is available to anyone who has a computer of almost any vintage in his or her home.

4.1.1 Everyday activities imbuing the conceptual model

In the initial stages of conceptualising the learning computer, a brainstorming process was undertaken. The goal of this brainstorming was to draw analogies between the learning computer and some everyday objects and activities to help conceptualise key features of the learning computer's design. These analogies include:

Doing the Laundry. Some automatic washing machines provide a good example of simplifying a task environment by using a special-purpose, embedded computer. A person need only provide some facts about the state of their dirty laundry through a simple interface and then press the start button, leaving the computer to accomplish the rest. Few users are even aware of the presence of the computer. It has become invisible to the process. Similarly, a student is able to interact with the learning computer through a few, simple operations defining what the student wants to learn and in what manner. Once those operations have been mastered, the computer becomes invisible to the learning task.

Navigating with a GPS. Sailors, soldiers, and car drivers equipped with a Global Positioning System (GPS) can roam at will and remain focussed on what interests them without worrying about getting lost. The GPS always knows their current position and can guide them back to where they began or to wherever else they want to be. The learning computer also tracks the student through the course and – no matter what path they take or where they arrive – can always bring them back to any desired position. Learning dimensions are synchronised so that a student can change their method or means of study without losing their position in the course.

Listening to the Radio. Radio and television are both popular means of communicating information about the world. The more multi-media character of television leaves less to the imagination of the audience than does radio. However, radio can also be a highly effective communicator which, because of its simpler technology, can be accessed and used in a greater range of situations than television. The learning computer is more

radio than television. It is not a multimedia simulation of a university and does not need to be. Its more minimalist approach still provides an effective learning environment that is accessible from almost anywhere.

Working From A Briefcase. Company sales representatives carry all their tools of trade in their briefcase, and can sit down anywhere to work at their tasks. They are not in permanent contact with their office, but can get in touch whenever they need to in order to exchange and update data or obtain a second opinion on a key transaction. The briefcase is their office. The learning computer is the extramural student's briefcase.

Learning from Books. The study guide and the textbook have provided the flexible core of successful distance learning for many years. While they have been carefully structured to guide the student in the most logical sequence of study, they do not impose it. The student may approach the material in any order they choose, provided they submit their assignments on time for grading. Additional learning resources, like revision exercises and glossaries, are organised around the book's chapters. Tables of contents and indexes provide points of entry for exploring and accessing those learning resources. The learning computer reproduces this flexible study environment, without attempting to reproduce or replace the textbook electronically.

Going to University. A university is a community of learning in which the student learns through participating in a wide range of formal and informal interactions with other students and with teachers. He or she can attend a lecture or a smaller tutorial session, go to the library, or just sit at a table and read a book. He or she can get additional help by knocking on a tutor's door, by arranging special one-on-one tutorials, or by sitting down with other students to work through a problem. The learning computer enables the extramural student to participate in this university community from the asynchronous, geographically-isolated environment of the distance learner.

4.1.2 Making learning possible

The learning computer is not a means for implementing or evaluating a particular learning paradigm. Nor is it a means for demonstrating that computer-based education can outperform the classroom. It embraces and incorporates the best available educational technologies that support the multiple teaching strategies and learning styles associated with university study, in order to outperform traditional correspondence-based lessons.

If there is a bias in the learning computer it is towards the user-centred, self-paced, exploratory learning style associated with learning from books, and towards capitalising

on the collaborative learning potential opened up for extramural students by the Internet.

The primary focus is upon the distance student as *user* of a computer system. This is based on the presumption that solving the usability and accessibility challenges is the prerequisite to making computer-based distance learning possible.

4.1.3 A reusable educational resource

Courseware emphasises course authoring and other teaching and administrative support. The learning computer emphasises support to the learner. From this perspective it is primarily a learning environment, rather than a teaching, authoring and administrative application. Nevertheless, to offer a practical solution to the challenges of extramural e-learning, the learning computer provides authoring support and promotes reusability across courses.

For the student, the learning computer works for any course for which they are enrolled. However, to maintain simplicity and focus, the student can only view one course at a time.

For the teacher, materials prepared for internal papers may be re-used in external ones. They may be updated while the course is in progress and re-used in subsequent semesters. Course authoring for the learning computer is conceived of as a dynamic process in which learning materials can be added and updated, in much the same way as a tutor may maintain any course website. Thus the tools for authoring a course are also tools for updating it.

A certain level of complexity is unavoidable in computer-based-learning systems. The objective of the learning computer is to place most of the burden of that complexity upon the builders of the system - the programmers and the course authors – and the least upon the student. A greater level of complexity for course authors and administrators is permissible because they can be assumed to be more computer literate and have access to computing and other support services at their university.

4.2 *Specialised interface*

4.2.1 E-learning metaphors

Metaphor is used to help the distance student develop a mental image of the learning computer and improve its usability.

Reflecting a large body of research on the subject, Collins (1995, pp. 192-209, pp. 223-24) emphasises the role of metaphor in interface design, writing that "old media serve as metaphors for new ones". The designer seeks to extend the metaphor, and surprise the user, by "adding magic," that is adding new functionality made possible by the computer environment.

An alternative view is presented by Benyon and Murray who stress the limitations of using metaphor to improve the usability of interfaces. They cite research findings that the "use of metaphor and analogy as a means of making system functionality accessible to user populations... can be a hit-and-miss affair, well-suited for some but not for the population as a whole, dependent as it is upon a closely shared appreciation of the basis of the metaphor" (Benyon et al., 1993, p. 198).

Collins stresses the importance of not surprising the user too much. He points to the computer spreadsheet application as an example of the tension between literalism and magic: "A *literal* interpretation of the metaphor would demand that the user open a computer calculator to do sums, then type the results back into the spreadsheet. But the spreadsheet is *magical* - the cells in a row or column can sum themselves, and automatically update the cell containing the sum. Literalism maximises understanding, but does not add any power; magic maximises power, but may interfere with understanding" (Collins, 1995, p. 192). Recognising this tension is of particular importance in computer-based educational applications.

Courseware typically uses the textbook metaphor implemented as HTML pages, to which magic is added via hypertext navigation and multimedia enhancements such as animation and video. Bork's highly-interactive tutorial method (Bork, 2001a) draws upon the metaphor of the private tutor, which historically was the preferred method of instructing the children of the rich. Hoppe et al. (2000) uses the classroom and its artefacts like the blackboard and the student slate.

The learning computer uses the university itself as its principal metaphor, and seeks to simulate its learning dimensions and study tools, with a minimum of surprises so as to maximise understanding. However, the learning computer does add magic to the university metaphor, in the form of its GPS-like tracking facility whereby the student can explore the course without getting lost and the system can synchronise the use of different learning dimensions and tools.

Another important metaphor is the traditional study guide and its table of contents, around which the learning computer is structured to give it a modular character. This is discussed next.

4.2.2 A modular approach

The environment provided by the learning computer is conceived of as a hierarchy of educational modules, as defined in the traditional study guide. A course is subdivided into sections which in turn are subdivided into topics. Each topic is structured around a small number of learning objectives and key concepts.

A course, or the individual topics in a course, may be taught in different modes approximating the different dimensions of an on-campus course (2.5.3). However, each topic offers the traditional textbook mode of correspondence learning as a minimum. Other modes are added by the course author, when and where they are available and will assist in learning that particular topic.

A study mode is defined by a set of elements, each of which corresponds to a basic constituent of university study. Learning by lecture, for example, may involve a presentation by the lecturer, a set of notes handed out by the lecturer, and notes taken by the individual student. Learning by tutorial, on the other hand, may involve a one-to-one interaction with a tutor, as well as notes made by the individual student (Figure 4.1).

At Massey University, an extramural study guide may contain a course overview and administration guide, a table of contents, assignment specifications, and for each study module - learning goals, reading guides and revision exercises. Each of these elements should be accessible when studying by textbook.

An element may be available in more than one learning mode. Some elements, e.g. a lecture presentation, are only available in particular modes, while others, e.g. notes taken by the student, are accessible from any mode.

Each learning element performs a distinct learning function independently of any other element. Learning resources are attached to each element, according to the current position (topic) in the course. This ensures synchronisation between elements and across modes. Students can then change mode without losing their place in the course, in order to take another approach to solving the learning task at hand. The initial learning elements and study modes of the learning computer are listed in Appendix B.

4.2.3 Rendering the interface invisible

To best support intellectual and creative activity, "designers can pursue the goal of having the computer vanish as users become completely absorbed in their task

domain" (Shneiderman, 1998, p. 18). For learning applications at least, this means simplifying the computer environment: "Nothing should be on the screen that is not directly relevant for learning the material at hand.... Learning does not need the complexities of today's interfaces" (Bork, 2001a, p. 64). An effective way of simplifying the environment for the user is to focus the interface upon the task domain (i.e. the "What" of the user) rather than the means of achieving those tasks (i.e. the "How" of the software engineer) (Gentner et al., 1990).

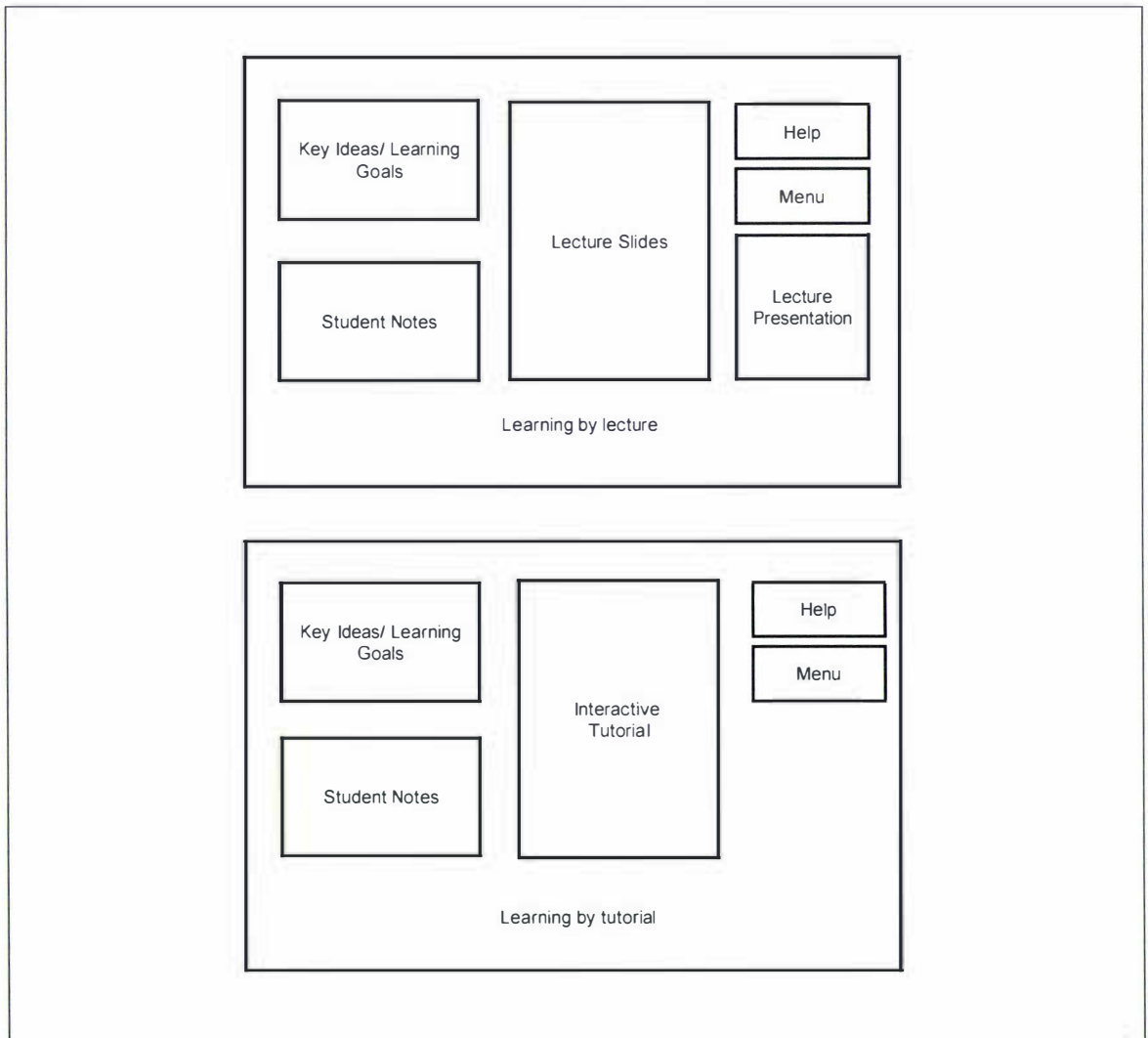


Figure 4.1: Each study mode contains a unique set of learning elements

The learning computer interface focuses upon the content of the learning material rather than the form in which it is stored and displayed on the computer. For example, the student elects to study a particular topic in lecture mode, or selects their notes on that topic. They need know nothing about what a particular file holding their learning material is called, where it is stored, or what application is needed to run it.

4.3 *Individualised interface*

4.3.1 Adaptation strategy

The learning computer is not a "one size fits all" system but has the capacity to adapt (or be adapted) to the learner and the learning material. As discussed in Chapter 3 (3.4.2, 3.4.3), a computer-based distance learning environment may, as appropriate, provide for three forms of adaptation:

- the system adapts to suit the student;
- the student customises the system to suit his/her needs; or
- the student adapts in order to collaborate through the system with other humans - tutors or students.

The emphasis of the learning computer is on providing a customisable and collaborative environment. However, where an adaptable or collaborative solution is unsatisfactory, or there is a proven need, a more sophisticated adaptive approach can also be integrated.

The learning computer's adaptable environment is designed to make it as easy as possible for the student to modify the system to suit his/her study needs (Table 4.1).

The basic forms of adaptation of the learning content are:

- choosing the mode in which a topic is studied, as when an internal student attends a lecture or reads in the library;
- choosing the topic that is studied and the order topics are studied in, as might be done using a correspondence study guide;

At the university level, supplementary individualised help is provided in the form of private consultations with a tutor, peer collaboration and personalised reading programmes. The learning computer takes the same approach to providing individualised study. The student may ask the system, the tutor or other students for guidance on a particular issue.

Different learning levels are also accommodated through one-on-one tutorials (incorporating adaptive tutoring tools where available), and through supplementary learning resources and activities, rather than by adapting all learning materials to the individual user. Furthermore, the learning computer assumes that sometimes the learner will adapt to facilitate collaborative work and discussion among students.

Dimension	Support
Interface “look and feel” - how user interacts with the system.	The minimalist interface provides little opportunity for customising screens apart from re-sizing and re-positioning screen objects.
Sequencing – what part of the course is presented next?	Student can select any topic in the course at any time.
Learning style – how the material is presented to the student.	Different learning styles are selected by the user choosing different study modes.
Learning level – the degree of difficulty of the material presented.	Some individual learning tools and study modes may accommodate different learning levels rather than the system as a whole. This accommodation is typically adaptable but may be adaptive.
Learning help – what help is presented to the user and when.	Queries are initiated and defined by the student user from anywhere in the course, individualised through user-driven interactive dialogue and exploration.

Table 4.1: Individualisation dimensions supported by the learning computer.

4.3.2 Integrated Support, Communication and Collaboration

The basic means by which the learning computer addresses the lonely learner problem confronting extramural students is by integrating communication, collaboration, and study help into a single framework. Distance students can communicate with each other, collaborate together on joint projects and help each other solve problems, seek help and guidance from the course tutor when necessary, and seek help and guidance from the system itself as an avenue of first – or last – resort.

This integrated approach represents a simpler method of achieving the individualised help goals of adaptive help systems. At the same time it adopts the approach of Murray et al. (2000) of developing a design which provides for more advanced or intelligent software features, but implements a less sophisticated version that can then be evaluated for evidence that students would benefit from the more advanced features.

The fundamental concept behind each aspect of this integrated system is outlined in the following paragraphs.

Communications

Students receive messages from other individual students or their tutor, and send messages to individuals, to their work group, or to all participants in the course. These messages are asynchronous, as with standard email, and can be transmitted whenever a student is logged onto the learning network. However, they can be read and written whenever the student is using their own learning computer. All messages are automatically marked with the position in the course (section and topic) at which they were composed.

Collaboration

Internal students organise themselves into workgroups for collaborative projects, and designate group responsibilities such as leader. The learning computer supports the organisation of extramural students on a similar basis. Co-operation is effected through tools that enable sending and receiving messages between individuals, sending and receiving messages within groups, and sharing documents within groups.

Group members communicate with the course tutor as well as with each other. These simple forms of collaboration enable assessment of group work through monitoring group discussion, peer assessment and deliverables produced jointly by the group.

Extramural Support

The learning computer provides two kinds of help – help with how to use the system and help with learning the course material. Help with using the system is provided in a context-sensitive, just-in-time, just enough, basis. Within the learning computer this is called “Help” and is taken up further in Chapter 5. In this section the concern is with providing additional learning help to the extramural student. This is called “Extramural Support”.

A student can obtain extra support in understanding any concept covered by a course. For the internal student, the most effective way to get this support is to ask another student or a tutor. In the learning computer this means sending an electronic message. However, in an extramural context there is no guarantee that someone is on-line and available to answer a query promptly. Therefore, the student can also query the Extramural Support system itself as a first attempt to get a meaningful response to their question, or when no other avenue is available in a timely manner.

The learning computer monitors the student’s interactions with Extramural Support and reports the results anonymously to the course tutor. For example, it notifies the course

tutor whenever a student finds a system response unhelpful. The student may also notify the tutor directly. These notifications, together with other queries and group discussions, including tutor responses, provide the basis for the dynamic updating of Extramural Support by the tutor to improve the system's response to particular queries.

Information gained from monitoring student interactions with the support system may also be used for guiding students towards concepts which they need to revise.

4.4 *Integrated network user interface*

The learning computer provides all the functions and features the extramural student needs for learning, through a single integrated interface. This means transparently integrating study materials sourced from a variety of locations, and in different media formats and learning styles. It includes communicating with other course participants and updating the computer when new or revised course materials become available. The learning computer is therefore more than an interface to learning materials stored on the student's machine. It is also an interface to a network of computers.

Course materials are accessible and usable whether or not the distance student's computer is currently connected to the university. Provision is made to incorporate learning material delivered by alternative means, e.g. the post (CDs) or satellite TV (video). Therefore, the learning computer provides a richer and more specialised user interface than that provided by a standard web browser, including the ability to display and execute software that runs in non-web-browser environments.

At the network level, these characteristics are made possible through the network acting as a delivery system for documents, programs and correspondence, rather than as an interactive learning medium. The interactivity resides on the student machine.

In this way, reliance upon the network to support learning functions is greatly reduced, as the entire weight of the system is shifted away from the education provider and to the individual learner. Most functionality now resides on the student machine (Figure 4.2). It also means that learning materials delivered by alternative means – such as portable storage media through the post, or satellite broadcast – can be seamlessly integrated into the learning environment.

Furthermore, the localised functionality of the learning computer fits well with the move among educationists away from top-down styles of teaching to more exploratory and collaborative forms of learning at the tertiary level. It is a vehicle through which the student can more easily explore their subject, including throughout the available network.

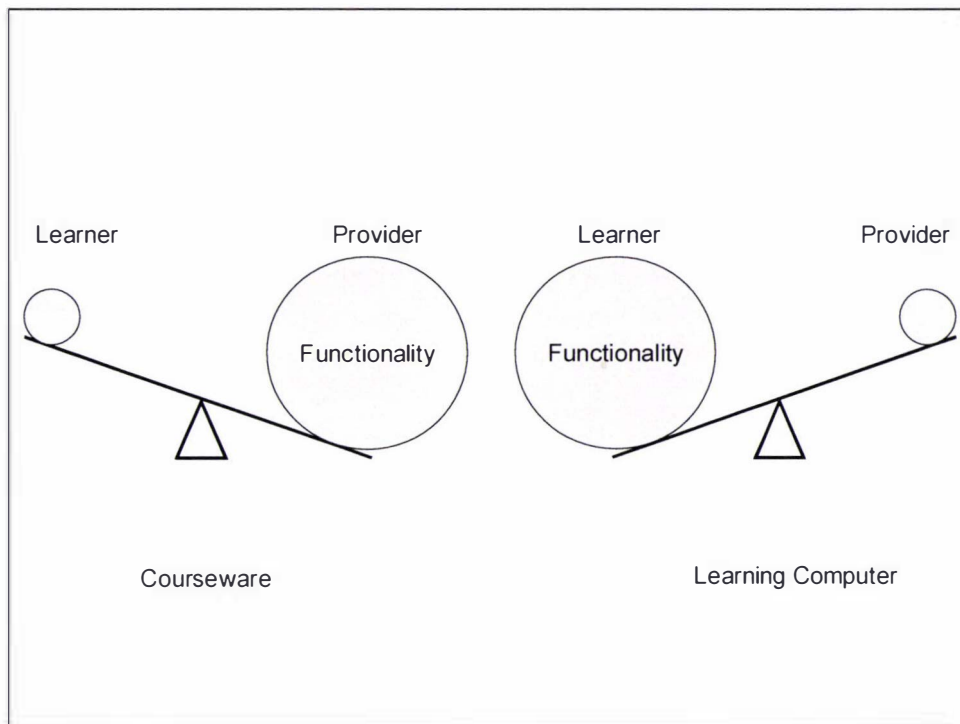


Figure 4.2: Learning computer shifts weight of system to learner.

4.4.1 Network characteristics

The learning computer takes advantage of the communication and co-operative work potential of computers linked into networks (and internetworks). But this potential is realised in forms that work over slow and unreliable network connections, while supporting the special requirements and preferences for an extramural learning environment. Using the time-space matrix for computer supported co-operative work in Shneiderman (1998, p. 481, based on Ellis et al., 1991) the system is both asynchronous and distributed:

- *Asynchronous.* The network supports anytime, anywhere study, including in different time zones and with different media and levels of sophistication in telecommunications. Key functionality cannot require synchronous communications between any components of the network, or assume high-speed network connections, and makes provision for an unreliable network. This requires an asynchronous communication model as with email.
- *Distributed.* Most system functionality resides on the student machine. Extramural study is largely self-paced with minimal requirements for centralised monitoring of students' activities. The network supports the frequent exchange

of short messages and periodic larger updates of the student's course materials from a central repository.

At the same time, to support *co-operative work* – i.e. for students (and tutors) to collaborate and communicate with each other over the network – the data held by each participant is periodically synchronised and updated through a central distribution point.

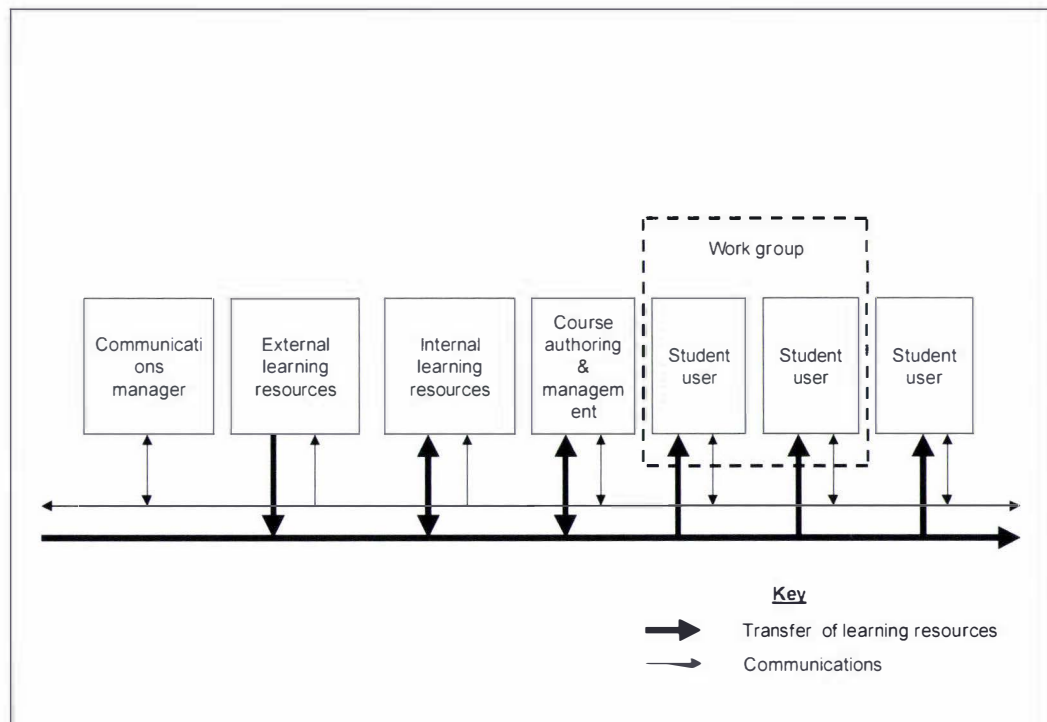


Figure 4.3: Networked extramural e-learning – conceptual view.

4.4.2 A networked extramural e-learning system

The conceptual view of the overall network required to support the learning computer is depicted in Figure 4.3. The high-level requirements each for the network components are summarised in Table 4.2.

The system supports two-way communications among all users, including the exchange of documents within workgroups, and the one-way downloading of learning resources from the university to individual students. It provides an internal repository for storing course resources and communications, as well as access to external repositories through which the student can access additional learning resources.

There is a communications manager which controls users' access to internal course resources (repository), including workgroup files and messages, and two distinct user components – the learning computer for the student and the course authoring and management system for the tutor.

Component	Rights	Responsibilities
Student	View own course View own grades Receive messages View group discussion View assignments View group biographies View supporting material Access Extramural Support Receive course updates	Logon onto system Make notes Send messages Add to discussion Submit assignments Ask questions to system
Course Author	View course View own messages View all assignments View all grades View all students View supporting material View Extramural Support View student records	Log onto system Edit/Update course material Send messages Add to discussion Grade assignments Update grades Locate/update links Update, respond to queries Add/delete students
Communications & Repository Manager	Access all internal resources	Control network access Manage messaging system Provide updated resources to users
Internal Resources	None	Store all course learning materials - documents, videos, tutorials; work group files, messages
External Resources	None	Provide supplementary materials - library, www

Table 4.2: Requirements for network components.

Provision is made for students to be organised into workgroups, which are monitored by the tutor. All monitoring of individual student performance beyond course deliverables and workgroup participation is private to the student component and used only to support an individualised help system.

4.5 Conclusion

In this chapter, a high level conceptual view of an extramural e-learning system has been presented, built around three design strategies -- specialisation, customisation and decentralisation. At its heart is the concept of a learning computer providing the extramural student with a simplified, easy to use, networked learning environment. To

support this environment a range of network services and support is required. In Chapter 5, the technological platform for this overall extramural e-learning system is specified.

Chapter 5

Towards a specification for an extramural e-learning system

In Chapter 4, an extramural e-learning system was conceptualised as a learning computer. In this chapter, a prototype e-learning system called IMMEDIATE (Integrating MultiMEdia in a DIstAnce learning and TEaching environment) is introduced. IMMEDIATE embodies a specification for a learning computer and its networked support services.

Incremental prototyping has been used to develop IMMEDIATE. This prototyping process is discussed in Chapter 6. In the present chapter, the major technological platform and design decisions that provide the framework for this prototyping are discussed. They are:

- the style of network user interface for the learning computer;
- the architecture of the IMMEDIATE communications network;
- the development platform upon which IMMEDIATE will be implemented;
- the basis for the learning computer's modular construction;
- the framework for providing an adaptable learning environment; and
- the overall architecture of the IMMEDIATE prototype needed to prove the concept.

None of these issues could be decided without reference to any of the others. The order in which they are discussed in this chapter reflects the priority in which they had to be addressed flowing from this interdependence.

The user-centred approach of the learning computer meant that the logical starting point for the design was the character of the user interface. Once this question was settled then the network architecture necessary to support this interface could be determined. To minimise any technological bias, the hardware and software development platform was not considered until these issues had been resolved.

5.1 Network User Interface

Fundamental to the concept of the learning computer is the integration of all the functionality an extramural student needs into a simplified graphical interface. This interface seamlessly combines resources located on the user's computer with those accessed over a network, or delivered via different communication media. Most functionality resides on the learning computer, with the network acting as a delivery medium. The system should be usable with or without a network connection.

For a student to link their home computer to a university server they must connect via the telephone system – either through a direct dialup connection or using the Internet via a dialup connection to their ISP. From the remote student's perspective, there is no performance benefit and considerably more expense (toll charges) to the direct dialup alternative. From the university's perspective, there is little to choose between the two methods, especially where the university network uses the Internet communication protocols (TCP/IP). Therefore, for the purposes of the IMMEDIATE prototype, network requirements can be addressed in the framework of the Internet.

In this section, alternative network user interface (NUI) styles are considered in light of these requirements, and an approach for providing an integrated interface is outlined.

5.1.1 Alternative network user interfaces

To determine the best method for implementing an Integrated Interface, a comprehensive review of alternative internet-oriented NUI styles has been carried out (Appendix C). While most such NUIs are referred to under the generic term, "web browser", their functionality and relationship to the underlying operating system have undergone an evolution, which have taken most of them a long way from the "write once, run anywhere" concept which browsers were originally designed to support. The main variations that can be identified are classified here as the standard web browser, the multi-platform enhanced web browser, the single-platform enhanced web browser, the specialised client interface, and the telecommuting interface. Each of these will be discussed in turn.

General-purpose browsers

The *standard web browser* accepts as input, text documents containing embedded instructions (tags) which describe the format in which the document is to be displayed. These text documents are stored on a web server and served over the network to the browser client. In this way, the same document can be correctly displayed on any

computer, regardless of its underlying hardware and operating system platform, provided a web browser application is available for that platform (Tanenbaum, 1996, p. 695).

These browser applications are known as thin-clients to distinguish them from client/server applications in which the client machine runs software that is stored locally (Dewire, 1998, p. 118).

To provide some interactivity, static web documents may be replaced by dynamic or active documents, which can respond to user inputs (Comer, 1999, p. 434). A dynamic document is created at the server end whenever a browser request is received, enabling the contents to vary from one request to the next. An active document is a program which is downloaded to the user's browser where it continuously recompiles and re-displays the document in response to user input. To run active web pages a browser must be "enhanced" by adding the capacity to compile pages from programs before displaying them to the user. The three basic types of web document are compared in Figure 5.1.

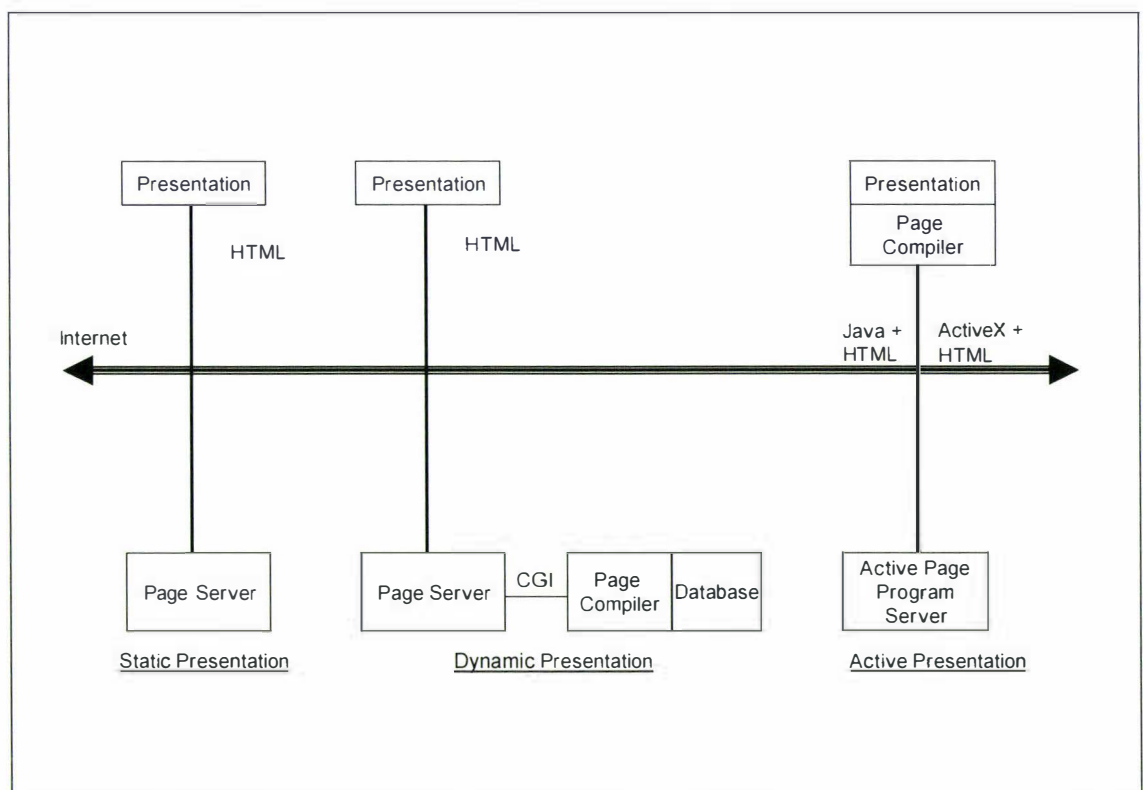


Figure 5.1: Three basic types of web documents.

Dewire notes: "In striving to be portable, HTML sacrificed some of the interactive capability that is proprietary to each operating system" (Dewire, 1998, p. 31). Most

browsers have been *enhanced* in order to utilise more of the functionality offered by the computer than is available through HTML¹ (Figure 5.2).

A wide range of technologies has been applied to extending the functionality of web browsers. One useful way these can be categorised is whether they work across multiple hardware/software platforms or whether they are specific to a single-platform.

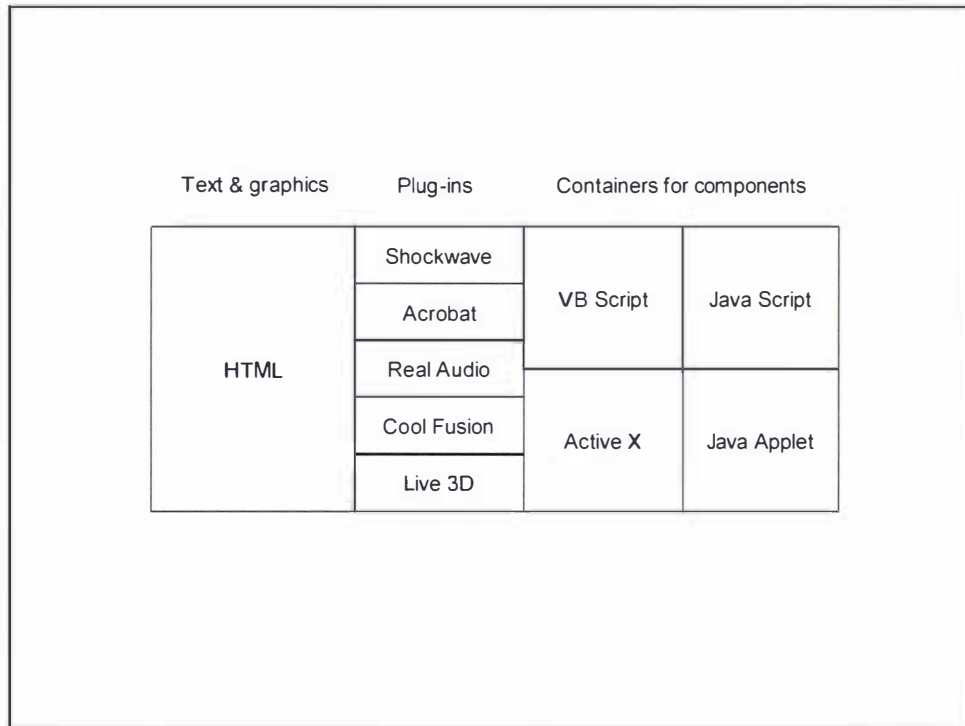


Figure 5.2: Enhanced web browser components (from Dewire, 1998, p124).

The *multi-platform, enhanced browser* provides additional functionality by using its own Java Virtual Machine to run Java programs (applets) downloaded with web pages. These programs run within the confines of the web browser and only interact indirectly with the operating system. They should work on any platform for which the web browser itself has been written. The applets are not persistent and must be downloaded each time they are required (Dewire, 1998, p. 210-13).

However, to use computer functions not available inside the browser environment – e.g. to print files or to write them to disk – Java programs will often be written to make direct calls on the operating system, tying them directly to the underlying

¹ Many browsers also have some capability to handle documents stored in some XML formats (Harald et al., 2002, p. 102). While this may increase the range of documents a browser can display, it has no effect on the architectural issues under discussion here.

hardware/software platform (Goldman et al., 1999, pp. 338-39). They are then essentially a single-platform enhanced browser.

The *single-platform enhanced browser* installs special programs called helper applications or plug-ins, on the user's machine, which interact directly with the operating system to provide extra capabilities. The browser passes a file to a helper application which runs directly on the Desktop in a separate window, whereas the plug-in displays the file within the browser environment. Such programs must be written for each computer platform.

For the Windows platform, plug-ins and helpers have been generalised as reusable software components called ActiveX controls. These are downloaded as required and stored locally for use by any application installed on the computer. For Java, JavaBeans play an analogous role within a browser (Dewire, 1998, p. 210-13).

The vast majority of browsers in use today are single-platform enhanced browsers, mostly in the form of the Windows-specific Internet Explorer.

Myth of the platform-independent thin-client,

In the course of reviewing various Internet-oriented NUI styles it has become clear that two attributes of the web browser favouring anytime, anywhere functionality – its thin client architecture and multi-platform capability – have become more myth than reality.

Goldman et al. (1999, pp. 287-88) noted that the scope of web browser software has expanded horizontally to include accessing resources on the client machine and local network, as well as Internet-attached resources. And this scope has also expanded vertically to include all Internet-attached resources and services, and not only the World Wide Web.

“Web browsers' transition to managing local attached hardware resources means that they must now be more closely integrated with client operating systems. Whereas in the past, web browsers were just another application program executing over a particular operating system via APIs and operating systems calls they are now more like a single integrated piece of software with a particular client operating system embedded within the web browser software...” (ibid., pp. 288).

The Microsoft/Netscape browser war of the 1990s transformed browsers into huge and complex software systems, as Netscape attempted to embed operating systems within their web browsers and Microsoft attempted to embed web browsers within their operating systems (ibid. p. 288). With Windows 95, before Microsoft merged its web

browser with its operating system, Internet Explorer required more disk space to install than the operating system (Dewire, 1998, p. 123).

The implications for web application developers is that rather than writing one program that will run anywhere, they find themselves increasingly having to provide different versions for each browser/operating system platform. For instance, the main impetus to Jade developing a special-purpose internet thin client for its database applications "was that the different suppliers' browsers were not always synchronised to each other with their functionality and we ended up with a multiplicity of special code to generate for each version/make of browser."²

The same problems face courseware developers who use the general-purpose web browser for presenting course materials to the learner. Web-based courseware's multimedia content requires standard plug-in and helper modules like a media player and PDF document reader. Any specialised functionality, such as TILE's Audiograph (Jesshope et al., 1998), requires application-specific software modules which must be written for each platform.

The implications for the student user of a web-based learning system are that they face the complexity of installing additional software on their computer, as well as opening them up to greater security risks (Goldman et al., 1999, pp. 337-38).

Specialised interfaces

Specialised client interfaces provide enhanced functionality more simply by building an interface exclusively for the specific application. Jade, for example, uses a three-tier architecture in which the interface accesses the application over the Internet, which in turn accesses the database over a local network (Jade, 2000). JADE provides an example where the functionality and usability of an Internet-based client-server database application has been improved by dispensing with the general-purpose browser in favour of a purpose-built thin client. A three-tier structure also allows for the application to be moved to the client, so that only data has to cross the Internet, while presenting the same interface to the user.

With the *telecommuting interface* the remote user logs onto their company LAN via a direct dial-up connection (Goldman et al., 1999, pp. 462-66), or via the Internet using

² Private communication from Keith Cowan, 10 September, 2001

tunnelling protocols as discussed later in this chapter (5.2.3). The NUI provides the same view and functionality to the remote user, as if they were directly connected to the LAN. Application logic is located on the remote machine or it is distributed between the client and the server. Remote resources are presented to the user as if they were stored locally. In the mobile computing variation, the user must be able to work off-line, requiring only periodic updating and synchronising of files across the network.

5.1.2 Integrated Interface

None of the NUI styles reviewed above exactly match the requirements for a learning computer interface. Rather, its integrated interface (Figure 5.3) combines aspects of several of these styles.

Of the general-purpose browsers, the integrated interface shares most technically with the single-platform enhanced browser delivering active web documents. With this style, the functionality resides on the client with the Internet acting as a delivery mechanism. And it combines the display of web pages with other interface objects that directly access additional operating system functionality as needed. However, the single-platform browser remains connection-dependent and presents its documents within the clutter and complexity of the general-purpose environment.

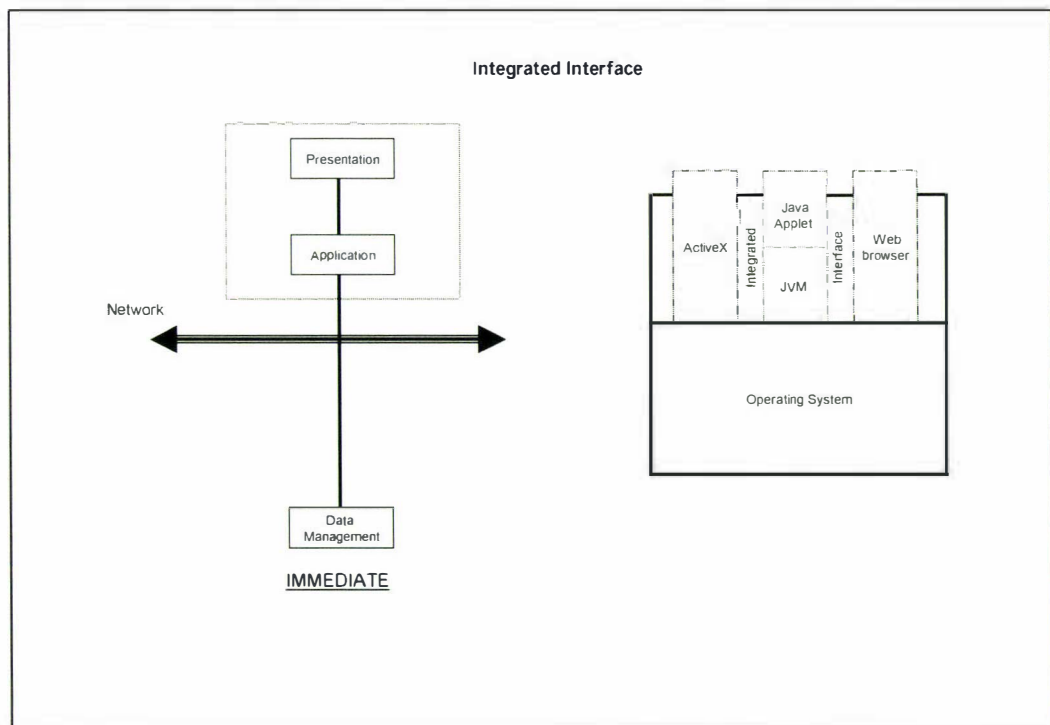


Figure 5.3: Integrated Interface.

The integrated interface implements the specialised client interface style in that the needed functions and features are simplified for the user by being presented within a specific interface for learning. Unlike the specialised thin client, however, the learning computer is not connection-dependent and makes richer use of local computer functionality.

The integrated interface combines the simplified interface of the specialised client with the integration of network and local features of the telecommuter interface. And, like the mobile computing client, the learning computer has all the required functionality available locally. Only periodic synchronisation with the university back-end is required.

It is now necessary to define a network architecture for IMMEDIATE that can support this combination of aspects of several different NUI styles reviewed above.

5.2 Network Architecture

If the learning computer constitutes the front-end of an extramural e-learning system then the back-end is the supporting services provided by the university. These services may be located on a single computer or a network of computers connected by the university LAN. Each learning computer connects to these services via a variety of possible paths encompassed by the wider Internet.

The learning computer only requires one-way broadband service for the downloading of learning materials from the course repository in the form of text, graphics, programs, audio and video. Collaboration and communication between students can be supported by narrow band two-way telecommunications as with Internet email. Figure 5.4 highlights the rich variety of communication media that can be supported with this approach. They include copper wire and wireless-based Internet, high-speed fibre optic backbones, satellite broadband download, as well as the transmission of data by satellite television broadcasts or via the postal system using portable storage media such as CDs or DVDs.

Within this spectrum of digital communication media, it is practical and feasible for a university to support unidirectional broadband and bi-directional narrowband communications world-wide.

Television broadcasts, textbooks and other non-computer-based materials can also be part of the overall learning environment supported by the learning computer, but IMMEDIATE leaves it to the user to synchronise them with related computer-based material.

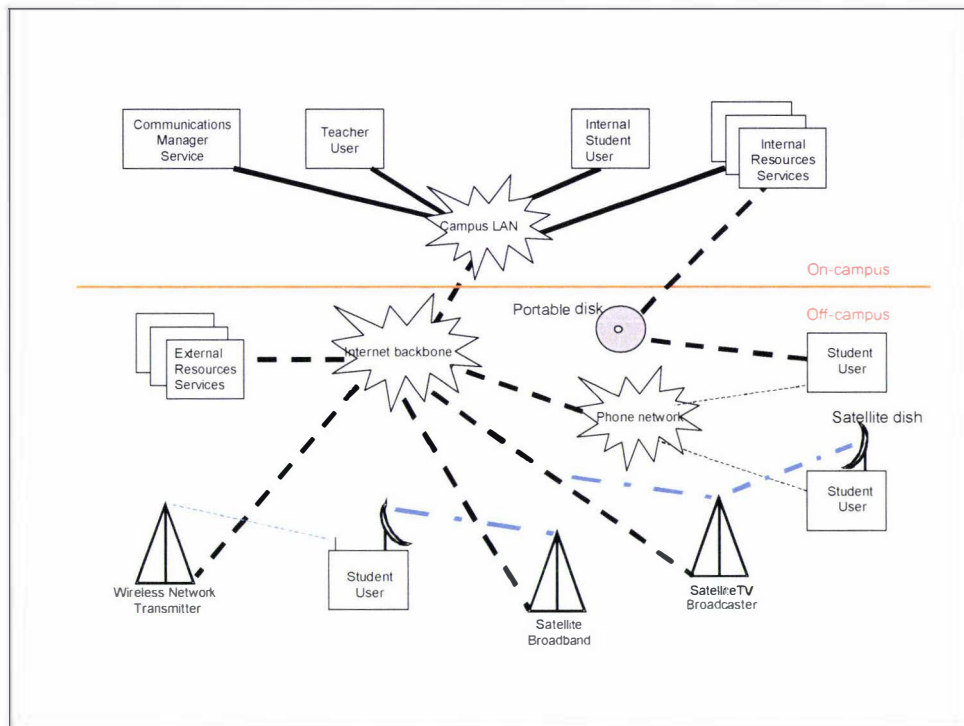


Figure 5.4: IMMEDIATE accommodates diverse communication media.

5.2.1 Architectural style

Tagg et al. (1997, p. 238) use an analysis tool called the Gartner Spectrum to distinguish between major client/server architectural styles (Figure 5.5). Using this spectrum it is possible to highlight the contrasting network characteristics of IMMEDIATE and web-based courseware.

Courseware is a centralised, hierarchical system, which implements the remote resource style. Only presentation software (i.e. a web browser) is located on the student machine. Application logic (functionality) must be downloaded with the data from the university server.

In contrast, IMMEDIATE is best represented by the distributed logic (peer-to-peer) style. All essential application logic and data are distributed to each user's machine. Data is periodically updated through a remote database, which acts only as a central repository rather than as the centralised data manager of the remote data management style.

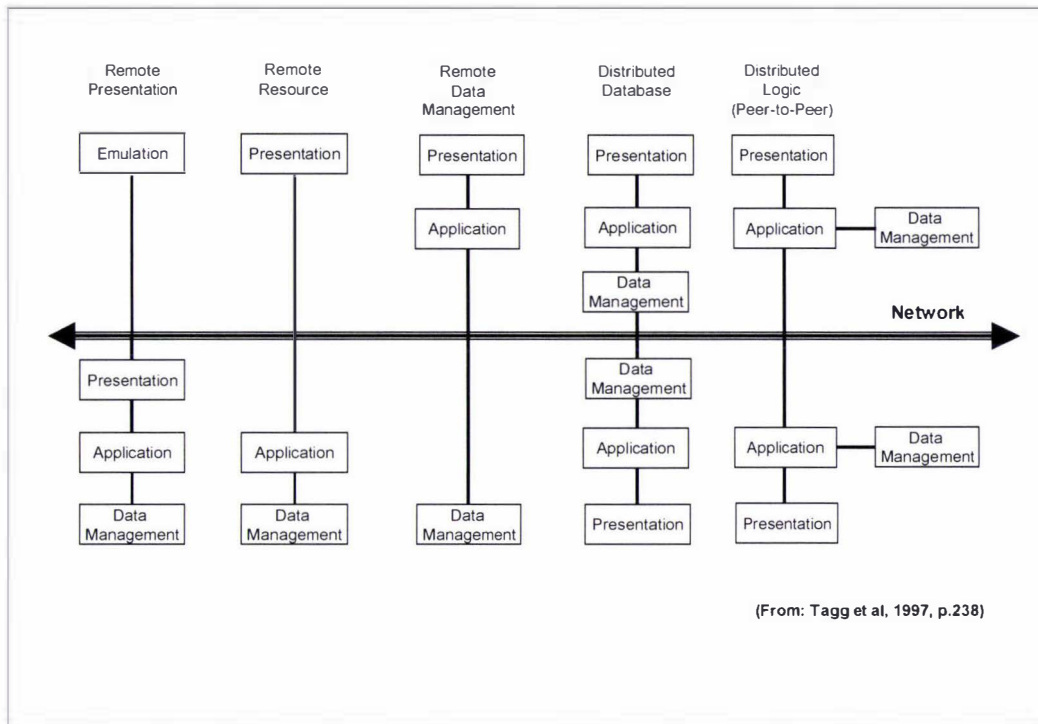


Figure 5.5: Gartner Spectrum of Network Styles.

5.2.2 Communication Style.

There are two main styles of communication where one application is requesting a service from another across a network – synchronous or asynchronous – which Tagg et al. (1997, pp. 280-81) aptly describe as hot or cool. With hot (synchronous) communications, the client and server applications are tightly integrated through intermediate software called middleware. With a Remote Procedure Call, for example, the calling application accesses the server application through the middleware's API, treating the request in the same way as a request for a service from the local operating system. The client waits for a response from the server before executing further code. If the request is not immediately executed by the remote application because that application is not running, is executing another request, or the connection is broken, then the calling program will fail. This style of communication is best suited to a fast, reliable communication network such as an Ethernet LAN (Comer, 1999, pp. 476-79; Goldman et al., 1999, pp. 252-253).

With the alternative cool (asynchronous) message-oriented style, communications are more like email. The client application sends a request to the server as a message. Once the server receives the message and has been able to process the request, it sends its response back to the client as another message.

IMMEDIATE will use a variety of the message-oriented style called message queuing

which is well-suited to communications across a less reliable network: "*Message queuing* replaces the direct connection between applications with a message queue. Each application attaches to a message queue that holds all incoming and outgoing messages until they can be processed. Because the message queue exists independently of the application, this removes the message-passing requirement that both applications be running before communication is attempted" (Goldman et al., 1999, p. 243).

5.2.3 Connection style

The communication protocols defined by the TCP/IP Reference Model (Tanenbaum, 1996, pp. 35-38), which underpin the Internet, have increasingly replaced proprietary protocols like Novell and Microsoft's NetBeui on LANs as well. For simplicity, all IMMEDIATE's network components should be built upon TCP/IP.

Under TCP/IP, all communications are broken down at the starting point into small parcels of data called packets, which may be transported by a variety of means and routes to the destination computer, where they are reassembled in the appropriate format. These communications can be as varied as a short asynchronous text message, a Remote Procedure Call, a large data file transfer, or a connection to a remote server for an entire work session. The correct handling of different types of communications is determined by additional protocols implemented on top of TCP/IP at each end of the line. Several relevant higher-level protocols in the TCP/IP family are summarised in Table 5.1.

SMTP	HTTP	S-HTTP	FTP	PPTP
Simple Mail Transfer Protocol for delivering email across an internet	Hypertext Transfer Protocol for communications between web servers and browsers	Secure-Hypertext Transfer Protocol for secure transmissions between web servers and browsers.	File Transfer Protocol for logging on to a server and downloading any permitted file	Point-to-Point Tunnelling Protocol for virtual private networking over the Internet

Table 5.1: Some higher-level protocols implemented over TCP/IP.

Web-based courseware provides for email messaging between course participants and the direct transfer of data files. However, it is mainly built around web pages (HTML) which use the HTTP protocol. HTTP is a stateless protocol in which each access of a file on the server (i.e. a web page or component of a web page such as an image or an

applet) is treated as a separate connection (Comer, 1999, p. 425). It is designed so that anyone can download any web page from any web site in any order. Thus, systems like courseware, which control who can access a website and may also constrain what they can see, must implement additional measures on top of HTTP. This may be as little as capturing user information for logon purposes or it may involve using database technology to generate web pages dynamically according to information obtained from the user, to restrict what that user can see and provide some interactivity.

If necessary, tunnelling technology can be used to implement a secure VPN across the Internet, using encryption (Goldman et al., 1999, pp. 603-05). Tunnelling provides the user with a continuous connection to a remote computer over an entire session. VPNs are used by remote employees to securely log onto their company network via the Internet to periodically update their files and exchange data or, where a broadband service is available, to maintain a permanent connection with the company LAN for telecommuting.

IMMEDIATE's support for a learning computer requires only that the network provide a mechanism for transferring files and synchronising data between computers. It is more analogous to the model of the remote worker updating their work files over a VPN than to the modified web browser/web server model of courseware. However, IMMEDIATE does not require the complexities of tunnelling. Therefore, IMMEDIATE will use custom protocols to ensure correct synchronisation built on top of the File Transfer Protocol.

Access to the repository of learning resources can be secured through password-protected logons. If additional protection is required for any data transfers, then encryption/decryption utilities can be implemented at either end of the file transfer process, using a freely available encryption algorithm such as the single key RC4 algorithm.³

5.3 Development platform

Given the ubiquitous nature of the World-Wide Web, the Personal Computer (PC) and the Windows operating system, there are only three basic options for a development platform on which to build the student end of IMMEDIATE for installation on a home computer. These are:

³ <http://www.vocal.com/RC4.pdf>

- a multi-platform implementation based around an enhanced web browser,
- a multi-operating system solution specific to the PC, or
- a Windows-specific implementation.

Each of these options will now be considered in turn.

5.3.1 Multi-platform

The multi-platform approach is the one usually adopted for developing computer-based learning systems today. It centres upon using Java applets to add additional features and functions to an HTML- (or more recently, XML-) based browser environment. This approach, however, has a number of disadvantages for implementing the learning computer strategy.

In order to work on all operating systems and hardware platforms Java applets must be limited to those features and functions available on all of them (Honeyball, 2002b). The learning computer requires a richer interface than can be easily built using these lowest common denominator capabilities. It would also be an advantage to have greater control of the application's look and feel than is possible using Java's algorithm-based screen organisation and display (Naughton et al., 1999, p745).

To provide all the required learning functionality, extensions such as applets, plug-ins and ActiveX controls, have to be written for each platform on which the learning computer is to run, negating many of the multi-platform benefits of using Java in first place.

Lastly, and most importantly, the simplified and integrated requirements of the learning computer interface are at odds with the visual clutter and complexity which accompanies a general-purpose web browser (Figure 3.1). Therefore, to effectively use the multi-platform approach, a special-purpose web interface would have to be built for every major computer platform.

5.3.2 PC-based

There are essentially two types of operating systems that are available for the PC – Microsoft Windows and Linux. A number of business users, and a small layer of experienced home PC users have shifted to the Linux operating system, an open source product based on Unix, due to its cheapness, adaptability and reliability. It has become especially popular as a server system for the Internet (Hall et al., 2000, p12). However, the great majority of home computer users, including almost all extramural

students (Extramural News, 2002), use PCs supplied with one or another version of Windows. To re-configure a Windows PC to run a Linux-based learning computer is not a trivial task and could not be expected of most computer users working in isolation. This is discussed further on in the chapter (5.4).

More interestingly, because Windows and Linux are both PC-based and therefore share the same hardware instruction set, applications written for one operating system are often relatively easily ported to the other (Wells, 2000). For example, applications built with the Delphi Integrated Development Environment (IDE) can be designed so as to be portable to the Linux platform via the Kylix development environment (Swart, 2001). Thus, in theory at least, a Windows-based implementation of a learning computer can be readily customised to work in all PC-based environments.

5.3.3 Windows-based

The Windows operating system appears to the systems programmer as a set of specialised C libraries, the WIN32 API, which can be directly included in a standard C or C++ program (Hart, 1997, Ch 2; Petzold, 1996, Ch. 2). Thus, the components of a learning computer might be built directly from the basic Windows libraries and made available to a learning system developer as ActiveX controls. "ActiveX controls are precompiled software objects that can be embedded into other applications. ActiveX controls can be written in a variety of programming languages including C, C + +, Visual BASIC, and Java" (Goldman et al., 1999).

A simpler and more efficient method for achieving the same end is to use a visual programming development environment, such as the Java-based JBuilder or the Object Pascal-based Delphi, which offers more direct support for an object-oriented, component-based, software engineering approach.

Java has become a popular programming language because of its strong support for object-orientation and its portability across computer platforms. But as an interpreted language that only uses a subset of the Windows API functionality it is a less effective option for building a Windows-specific application than Delphi, which is fully compiled and more tightly integrated with the underlying operating system.

According to Cantu (1999), Delphi "was and still is the best combination of object-oriented programming and visual programming for Windows" (pp. xxix). Delphi maximises what a competent programmer can get from Windows, without their having to directly deal with the complexities of the Win32 API. Because Windows libraries can be dynamically linked to any calling program at runtime, the full functionality of the

operating system can be accessed through the Delphi development environment. It does not matter that Delphi itself is not C-based.

The components of a learning computer can be built in Delphi and then made available to a system developer as a package (i.e. a "tab") within Delphi's "drag and drop" visual component library ("palette"), or they can be "re-wrapped" by Delphi as ActiveX controls, thereby becoming available to any Windows-based development environment supporting ActiveX (Cantu, 1999, pp757-59).

A more recent development has been the introduction of the .NET Framework by Microsoft, which is intended to eventually replace the Win32 API as the basis of all Windows applications. .NET provides a standard framework for developing standalone or web-based applications, which may be written in any language, removing the need for each language to have their own framework. The programs are then compiled to run on a virtual machine that draws upon the full functionality of the Windows operating system. The latest versions of Delphi are .NET-based and provide a facility for migrating Win32 applications to the .NET Framework (Microsoft, 2005; Borland, 2004).

Based on all these considerations it was decided that IMMEDIATE should be prototyped for the Windows platform using the Delphi IDE.

5.4 A Specialised User Shell

The goal of the learning computer is to simplify the computer environment and render the interface invisible to the user. In this section, the technical outlines for meeting these requirements within a Windows framework are discussed. The student view of the learning computer is a special-purpose, direct-manipulation, graphical interface, which provides all the functionality needed for a user to complete a task in the learning domain.

To achieve this it is necessary to remove any visual garbage not directly relevant to the task at hand. General-purpose features and functions associated with the operating system or a web browser, that are not needed to accomplish a task in this domain, should be treated as a distraction and not be made available.

In a Windows context, this means:

- preventing the user from accessing or being distracted by general-purpose features such as the Desktop and the directory system;
- removing the standard menus and toolbars associated with individual windows, including web browser windows; and

Approach	Possible implementation	Comments
Specialised machine	General purpose PC that is only used for a single purpose.	The "visual garbage" would still be there. Not realistic to expect it to remain single purpose in the home environment.
Specialised user profile	User profile with specialised system settings i.e. customised desktop and directory view	Goes beyond options available for Windows profiles. More realistic on Linux which has unique directory system for each user and can be customised to meet special requirements. Does not hide superfluous features and functions.
Specialised application	A Windows application with standard menus etc removed.	Simplest implementation. Can be readily installed on any machine using same operating system. User still within the Windows environment.
Specialised operating system	Dual boot Windows/Linux with a customised Linux GUI as learning computer.	Requires skilled installation
Specialised user shell	Superimposes own desktop over the operating system desktop. User can access operating system features and functions only through the specialised shell	Can be installed as an application, but shuts out the general Windows environment.

Table 5.2: Possible methods for rendering the interface invisible.

- limiting generic functions like electronic mail and web browsing to the current learning domain.

Five possible methods have been considered for implementing these requirements on a PC running some version of the Windows operating system (Table 5.2):

- *A single-use computer.* Implementing a single-use Windows-based PC approach requires that the extramural student maintain a special computer at home reserved

for learning. All Windows general-purpose functions and features would remain accessible.

- *A special-purpose user profile.* Windows operating systems allow user profiles to be set up which restrict what is displayed on the Desktop and what data files can be seen by a particular user. All programs installed on the machine can be accessed, however, as can all their general-purpose features and functions. It is not realistic to expect all extramural students to be able to set up and maintain their own user profile for learning.
- *A special application.* Installation wizards have made it possible for relatively inexperienced computer users to install Windows applications. A special application can be presented through a custom interface without all the standard Windows features like menus and taskbars. However, the learner is still working within the overall operating system environment whose complexities and distractions are just a mouse-click or keystroke away.
- *A specialised operating system with a dual-boot PC.* Operating systems are implemented on at least two levels – an inner kernel containing the functionality, and an outer user command interface, or shell, through which the user interacts with the kernel functionality (Flynn et al., 1997, pp. 4-7, Brookshear, 2003, pp. 113-115). Modern operating systems offer a direct manipulation graphical user shell.

Linux, with its open source code and a separately-available kernel, has been designed so that programmers can add their own user shell (Dyer, 2004). In theory, the learning computer could be implemented as a specialised user shell for the Linux operating system. However, to run on a Windows-based PC would mean creating a dual-boot system by partitioning the computer's hard drive and installing the specialised operating system in the space created (Hall et al., 2000, p. 54). This is way beyond what can be asked of a typical home computer user.

- *A special-purpose graphical user shell for Windows.* Another possibility is to add a special-purpose user shell to Windows. Implementing this shell is more problematic than in Linux, because Microsoft's proprietary system does not provide an open source or kernel-only version that would allow its GUI to be easily replaced by a custom version. It, therefore, would have to be superimposed over the Windows GUI in such a way that Windows becomes invisible to the user.

From reviewing these alternatives it was concluded that the best approach for implementing the learning computer would be as a special-purpose user shell. This could be done most cleanly as a Linux-based implementation. But to meet the

requirements that the learning computer be simple to install and run from the Windows environment, the best solution was for IMMEDIATE to be implemented as a specialised graphical user shell – a Learning Shell – over the Windows operating system.

5.5 Modular construction

The learning computer should be designed and built on a modular basis, with separate interface, application logic and data management layers. To further facilitate maintenance and reusability, distinct logical functions within each of these layers should be implemented in separate software modules.

A requirement of the learning computer is that it teaches a topic in a number of alternative study modes. These modes are defined by a unique subset of elements, each of which encapsulates a specific learning artefact. This modularity would suggest that the component approach to assembling interfaces, popularised by rapid application development (RAD) environments like Delphi and Visual Basic (Osier et al., 1997, p. 4), is a good fit for building an Integrated Interface. Study modes can then be implemented through a mechanism for selecting from a pre-installed set of learning components (Figure 5.6).

Dewire (1998, pp. 198-99) describes a component as a reusable software module, designed to be used within another application called a container. It may consist of one class, be a composite of several classes, or be an entire application. In contrast to an object, whose code must be accessible for the object to be used, a component is designed so that it can be used without modifications to its source code.

With a RAD approach each learning element can be implemented as a separate software component encompassing the necessary functionality. The following functionality may be provided by the component:

- customising a general-purpose tool provided by the underlying operating system (e.g. a media player);
- linking to a separate application installed on the user's machine or accessed via the network (e.g. an adaptive tutoring application); or
- encapsulating a more sophisticated, purpose-built learning tool utilising new educational technologies (e.g. XML-based learning objects).

A learning component uses the same general form to display different content to the student at different points in a course. Conversely, different learning components map the same general learning content to different forms at the same point in a course. The

learning computer maps the appropriate data file to a component.

Not all elements of the learning computer interface embody learning tasks. Some perform system management tasks such as capturing logon or other information about the student, or to enable the student to navigate the course and access different study modes. These should also be implemented as reusable software components.

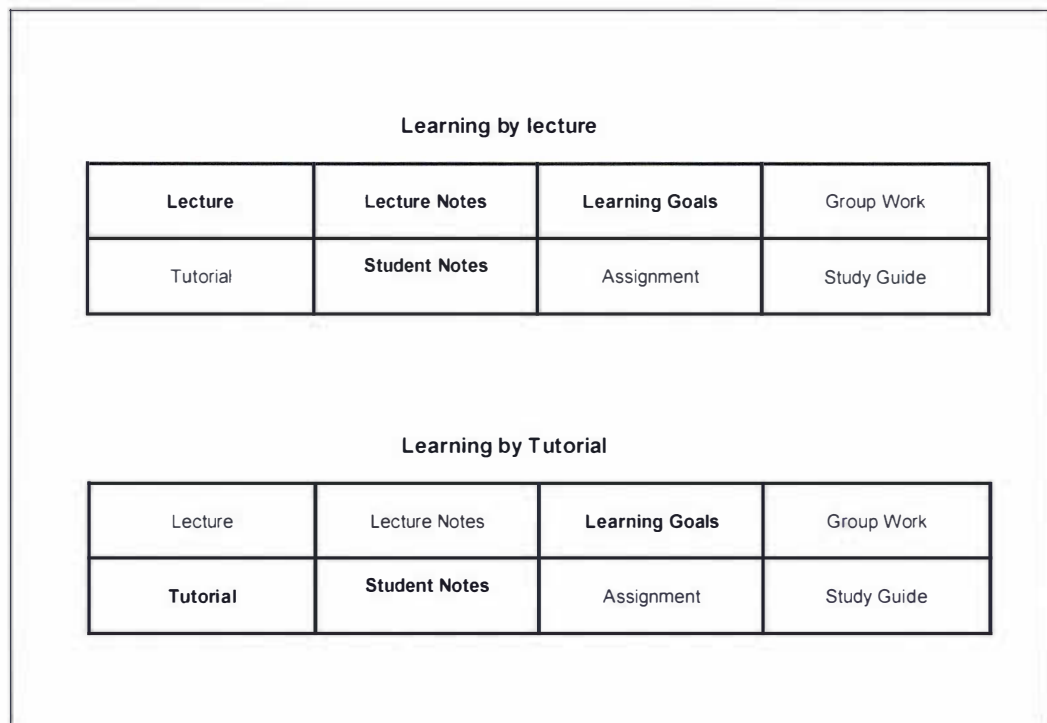


Figure 5.6: Modes implemented by selecting from pre-installed components.

Just-in-Time Help

To provide help in using the various elements of the system, a Help component should be linked to every interface element. This component should provide specific, context-sensitive help on using the particular interface element. It should not merely provide a path to an overall help system, which the user must try to navigate to find what they are looking for.

In the Delphi RAD environment, very specific help on a component is provided to the user (i.e. the programmer) by their selecting the component with the mouse and clicking the "F1" key (Inprise, 1999b, pp. 2.11-12). In an analogous way, IMMEDIATE should provide just-in-time, just enough information for using a component whenever the user (i.e. the student) selects that component and clicks the "F1" key.

5.5.1 System Components

The Learning Shell is required to manage Windows functions like file input and output, document printing, the opening and closing of screen windows, and networking, in such a way as to render the operating system invisible to the learner. This requires an intermediate level between the learning computer interface and the operating system. This system level is required for translating and synchronising actions between the domain-oriented objects and interactions of the Learning Shell and the generic file-oriented functions of the operating system. To facilitate modification and re-use the system level should also have a modular design.

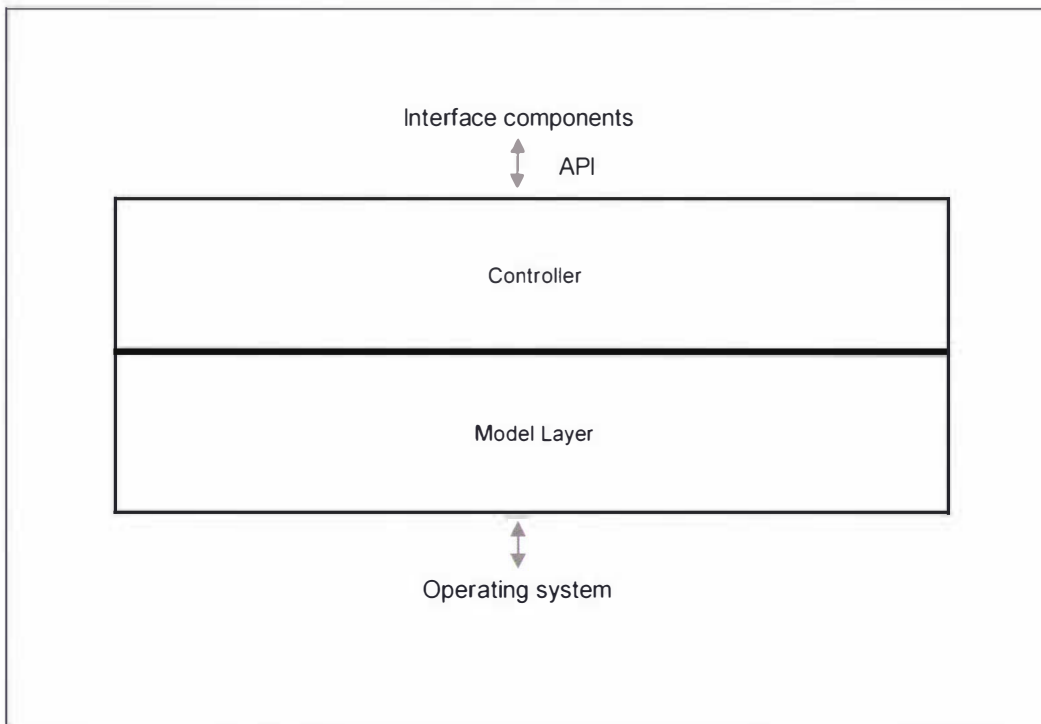


Figure 5.7: Learning Shell System components.

The essential parts of the system level are shown in Figure 5.7. They consist of a Controller object and a model layer. The Controller object provides an API through which interface components interact with each other and with the system level. It plays an analogous role to a browser's controller, which manages the other components and calls on them to perform operations specified by the user (Comer, 1999, p. 427). The learning computer Controller, however, has additional capacities to manage the interface for the user, through reference to the model layer.

The model layer is a set of data structures and related operations, each of which models a different aspect of the learning computer. These data structures together constitute a *reference model*, enabling interface objects to interact and synchronise

with each other correctly and with the Windows file system.

The reference model incorporates the System Tree, a hierarchical structure modelling the course table of contents. The System Tree provides the basis for navigation to different sections and topics within the course and records the student's progress through the course, using a traffic light metaphor (Figure 5.8). Changing position in the System Tree does not change the student's study mode.

The reference model stores information about each study mode and learning component offered by a course. This provides the basis for swapping between study modes within the course. Changing study mode does not change the student's position in the System Tree. The reference model also models the directory system containing course learning resources. It maps a learning component to its correct resource file.

To facilitate individualisation, the model level must model the student as well as the system. It therefore includes a Student Model, which stores information about the user needed by the learning computer that would not otherwise be available. The Student Model is discussed further in the next section.

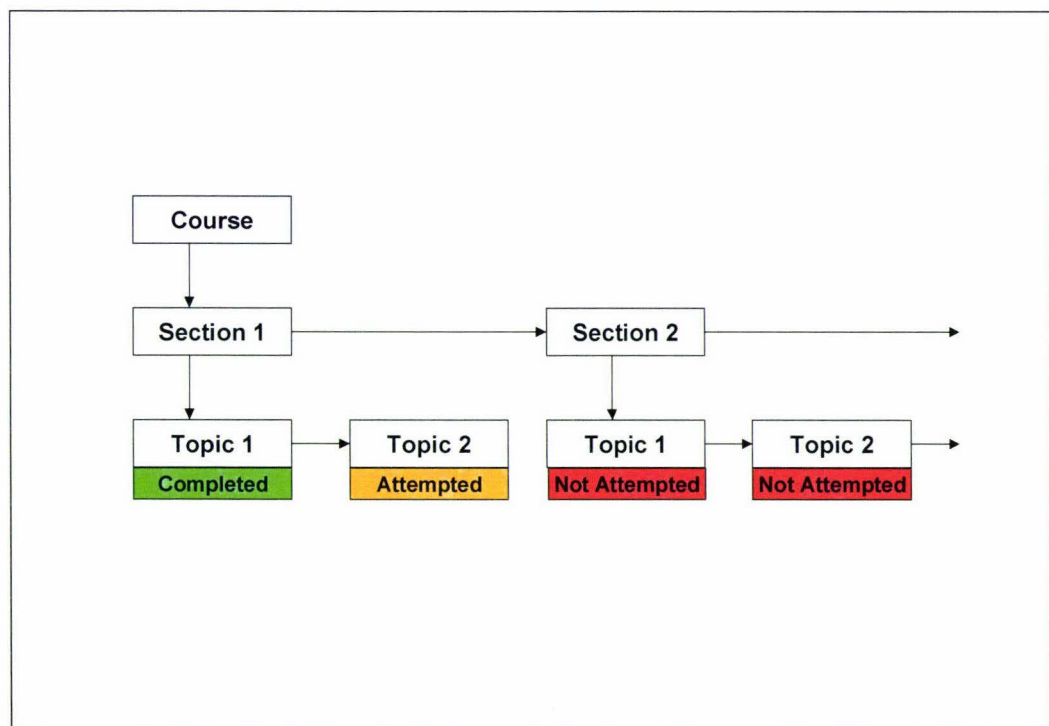


Figure 5.8: System Tree models table of contents and student's progress.

5.6 Adaptable environment

A Learning Shell must provide an adaptable environment which the student can individualise to suit his or her own learning preferences and needs. Murray et al. (2000)

highlight advantages of the adaptable approach to individualising a learning system such as retaining the locus of control with the user and being simpler to implement.

This section specifies the key elements for achieving this goal within the IMMEDIATE prototype.

5.6.1 Student Model

Individualisation of the Learning Shell is made possible through reference to a locally-stored Student Model.

Adaptive teaching systems model the student's knowledge so that the system can dynamically adapt its teaching strategies to suit the individual student (Allessi et al., 1991, p. 463). IMMEDIATE models the student's knowledge so that the system can help the student adjust their learning strategies and priorities.

The Student Model records the student's current state vis-à-vis the course, which includes:

- their name and password;
- their current position in the course (section and topic);
- their current study mode;
- the sections and topics they have attempted or completed;
- their progress in meeting key learning goals and concepts; and
- any other information necessary for configuring the user's environment.

The information in the Model is saved to permanent storage whenever the student's state is changed.

The Student Model enables the system to track the student through the course in order to preserve their position when changing study mode or quitting the system. On re-entering the Learning Shell, the user can be given the options of returning to where they left off previously, or of viewing how far they have progressed through the course and selecting a new topic or mode of study.

5.6.2 Integrated Learning Support

Providing learning support to individual students is facilitated by the integration of this support with the communication and collaboration facilities.

The basic techniques for collaboration and communication amongst students and with

a tutor across a public communications network are well-established and are now a basic component of courseware packages. The simplest method is to provide email-based correspondence and discussion groups. Courseware systems may utilise their database capabilities to organise these discussions in specific threads or link them with specific parts of a course (Goldberg, 1997).

Extramural Support should enable direct student queries on a particular subject. And it must work when the student is off-line and therefore cannot be connected to a central database.

To meet these requirements the integrated system should be built upon a relational database, which stores all correspondence, discussion and learning support information. SQL queries offer a straightforward mechanism for providing individual student and tutor views of this database, and for searching for answers to student queries.

This database should be organised along the lines of IMMEDIATE's distributed logic architecture. The best fit would be along the lines of models that have been developed to support mobile computing such as the "Briefcase model" (Inprise, 1999a, pp. 13.17-13.18) or "asynchronous replication" (Connolly et al., 1999, p. 712). These represent a more peer-to-peer style of distributed database organisation in which each user has a local copy of the database and some mechanism is provided for caching of updates to allow asynchronous, off-line remote functioning.

Separate email-based communications are not necessary in IMMEDIATE because all students can be linked to a central copy of the database over the Internet. Local copies of the database on student machines can then be synchronised and updated automatically whenever the student logs onto the network.

5.6.3 Individualised Support

As well as integrating learning support with the system's communications and collaboration, IMMEDIATE must *individualise* it so that a student can receive meaningful, relevant help from the system itself, without having to turn to the course tutor or fellow students with every query.

IMMEDIATE should provide an adaptable mechanism through which the user initiates a dialogue with the system by defining a query on a particular subject and then receives a specific, context-sensitive response. Laddering, i.e. the repeated probing of an issue via Why-type questions (Reynolds et al., 1988) should be supported. In other words, if the student is not satisfied with the system response then it should be

possible for the student to repeat or re-define the query to multiple depths, in addition to being able to pass the query on to the tutor.

Extramural Support should be able to learn from its experience and improve its responses to the same or similar queries over time.

The starting point for meeting these requirements is recognising that because the learning computer tracks the student as they work, all communications -- including queries and their responses -- are linked to a particular position in the course. This provides the framework for organising these communications into a coarse-grained knowledge base.

Organising Learning Support as Knowledge System

There are three elements to be considered in utilising the system database for a knowledge system:

- Providing a more user-friendly interface query interface than standard SQL or Query By Example can provide;
- Developing a directed search strategy which aims to return a more meaningful result set than is possible through a blind or brute force search of the entire database.
- Modifying the structure of the database (e.g. by adding fields) to facilitate implementing either of the above strategies.

User-Friendly Query Interfaces

Harmon et al. (1985, p. 262) note that: "In many cases, small knowledge systems derive their utility from their user-friendly nature rather than from their ability to capture knowledge that would be difficult to represent in a conventional program."

Querying databases is considered a promising arena for applying Artificial Intelligence techniques for natural language parsing because databases usually represent a coherent and restricted domain (Luger et al., 1998, Ch. 8). Another simpler approach allows a subset of English to be used in natural-language-like queries which may be parsed by their syntax or their semantics (Heinrich et al., 2000; Heinrich et al., 2001).

For the purposes of this initial prototype a highly-simplified modification of the semantic querying approach in Heinrich et al. (2001) will be used. Queries are limited to four pre-defined types "What?", "Why?", "Where?", "Who?". The semantics of these queries are summarised in Table 5.3. The user selects one of these query-types and then

associates it with a key word or phrase in the course material. The efficacy of this approach may then be evaluated, and the need for a more sophisticated approach assessed.

Dimension	Meaning	Comment
What?	What is the meaning of this word or phrase?	Involves finding the best match for the phrase (or its synonym) in a course glossary.
Why?	Explain this concept further.	Involves searching a database for the most appropriate explanation or combination of explanations of a particular concept (phrase).
Where?	Where can I find further reading on this concept?	Involves searching a database for further references (links) to the current phrase, subtopic or topic.
Who?	Who has written further on this concept?	Involves searching a database (or internetwork of databases) for further references (links) to authors on the current phrase, subtopic or topic.

Table 5.3: The four dimensions of the Integrated Help system.

Directed Search

A perusal of Table 5.3 shows that the concept of searching a database is at the centre of Extramural Support. The simplest method is a text search of the entire database looking for key words, and returning every matching record. This blind or brute force search strategy will find any and all references to the key words in the database but will have no way of selecting and ranking those results likely to be most helpful to the user. A more directed search strategy is required.

Gonzalez et al. (1993, Ch. 1) describe search strategies as the "foundation of artificial intelligence" (p. 3). The principal components of a Knowledge-Based System are a database, a user interface, and an intelligent program, or inferencing mechanism, mediating between the user and the database (Figure 5.9).

Within the Extramural Support system, the inferencing mechanism consists of the rules for searching the database to provide the most meaningful possible result. A directed search strategy includes what to look for, where to look for it, and how to rank/present results? Extramural Support implements these in the following manner:

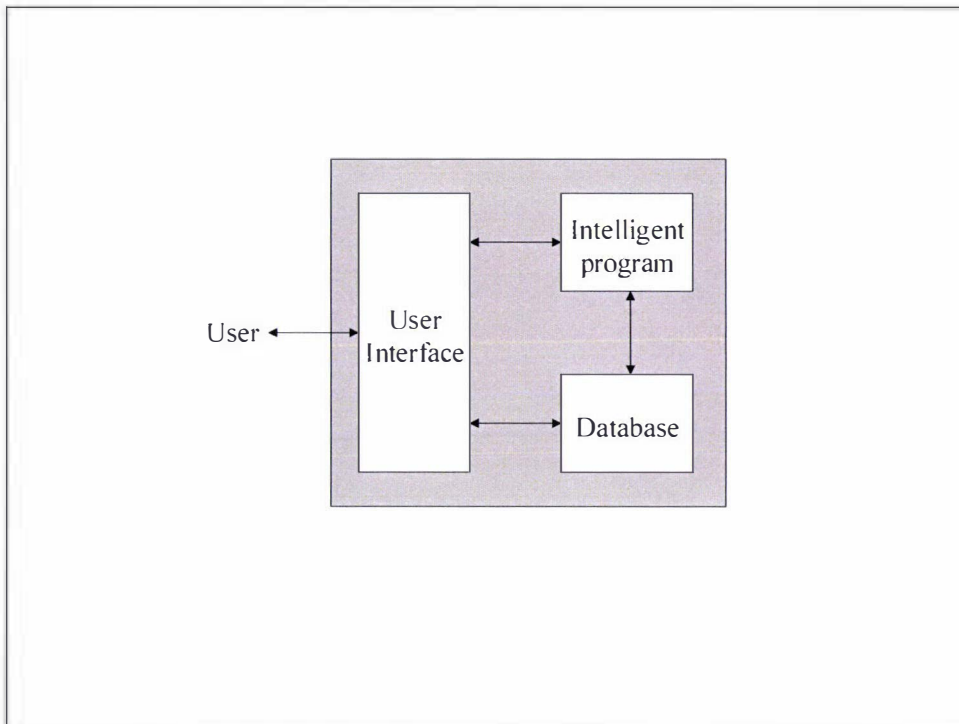


Figure 5.9: Knowledge-based system components (Gonzalez et al., 1993).

- *What to look for.* The user selects what type of query to make and on what subject (i.e. what key word or phrase). The system maps the key word or phrase to related terms.
- *Where to look for.* The system uses the system tree to search the database, beginning with the current topic, then the closest adjacent topics (i.e. leaf nodes in the System Tree), until sufficient results have been found.
- *How to present/rank results.* The query results are ranked so that the most relevant result is presented first and the least relevant, last. At its simplest level, the results are automatically ranked by their proximity in the System Tree to the current node. The implementation should allow for more sophisticated ranking algorithms to be added during the prototyping process.

Modifications to the database

Because the System Tree is the meta-description of the learning computer – used by the student model, extramural support and the directory system – text and keyword searches of such a structured database are directed in a way not otherwise possible. The teacher edits his or her responses to student queries and other useful discussion items, including by adding keyword and other links between entries to facilitate various query types and deeper searching. In this way the integrated help system is built dynamically and collaboratively by the tutor and students.

5.7 IMMEDIATE specification

The learning computer is conceptualised as the front-end of a networked e-learning system for extramural study. The main components of this network, and their high level requirements, are discussed in Chapter 4 (4.4.2). Ease of maintenance and reusability require a clear separation of functionality between these network elements

IMMEDIATE is a prototype system developed to test and evaluate the concept of the learning computer. It therefore needs to implement only those elements that are necessary to achieve this.

While the student interface is central to the learning computer concept, this project is much more than an interface evaluation exercise. It is also about proving the feasibility of a mechanism for the authoring and delivery of learning material in support of a collaborative learning environment, that is accessible and usable by extramural students studying alone, utilising strategies of specialisation, localisation and adaptability. This means that to implement IMMEDIATE for the purposes of evaluating the hypothesis, three separate prototypes need to be built and then integrated into a working system. They are a Learning Shell, a Course Authoring and Management Application and a Communications Manager.

The Authoring application is linked to the Communications Manager and the Course Repository via a Local Area Network (LAN).

Earlier in this chapter (5.4), the rich variety of communication media that can be supported within this approach was highlighted, including standard telephone, wireless, digital satellite and postal technologies. The essential requirements are the provision of a bi-directional narrowband Internet connection for communications and a uni-directional broadband media for downloading learning resources from the course repository to the learning computer. Variations between telephone, wireless and satellite broadband delivery are resolved at the level of the computer hardware and operating system, and appear to the learning computer as an Internet connection. Therefore the two main categories of communication media that need to be evaluated through prototyping are the Internet and portable storage media.

The Learning Shell is linked to the Communications Manager via the Internet using a secure FTP connection and through portable storage media such as CD-ROM. This provides one bi-directional communications medium (i.e. the transfer of message files over the Internet using FTP) and two unidirectional download media (i.e. downloading resource files using FTP via any Internet-enabled channel or by mail using portable data storage). Resources external to the Repository are accessed via the Internet.

The essential components of IMMEDIATE and its configuration are depicted in Figure 5.10. This is a simplified network architecture specification, compared to that in Figure 5.4, but it is sufficient to evaluate the learning computer concept.

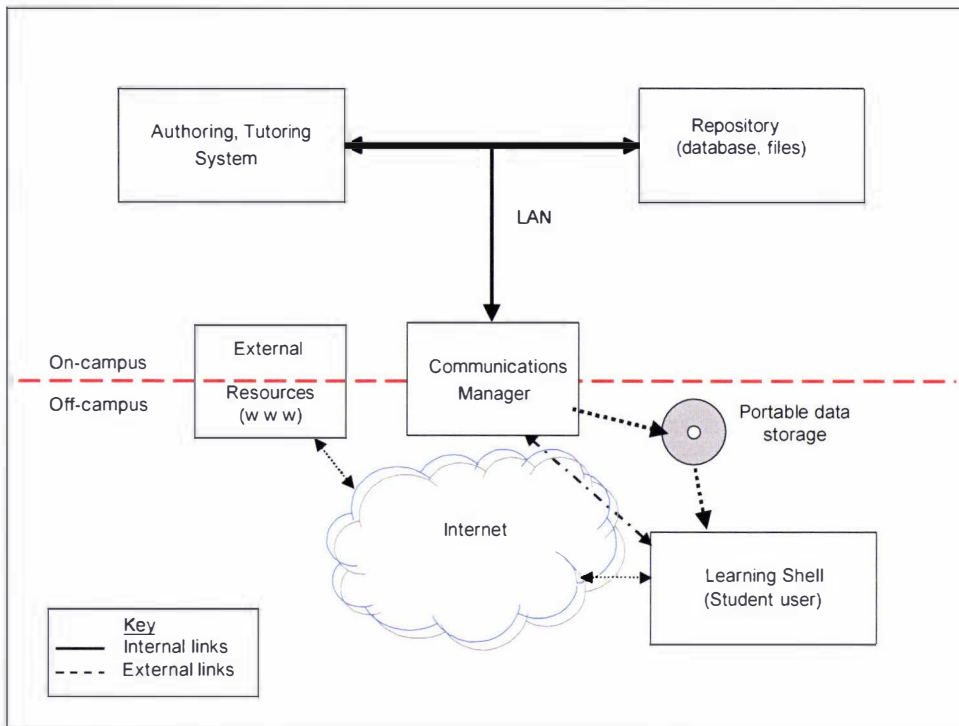


Figure 5.10: IMMEDIATE network components and configuration.

5.8 Summary - Prototype Focus

This chapter has presented a partial specification for a specialised computer for learning, and its supporting network services. This provides the framework for implementing a prototype system called IMMEDIATE. Key aspects to IMMEDIATE's design include:

- A client/server network architecture in which the student client is implemented as a specialised Learning Shell over the Windows operating system. The university server provides data updates to the student clients over the network, or by alternative media, and enables two-way communications between students and between students and teaching staff. Learning functionality is distributed to the student clients.
- The Learning Shell is assembled from reusable RAD "drag and drop" components with settable properties. Different study modes (learning dimensions) are defined by particular combinations of components, to which appropriate learning resources are bound. Learning components are defined from the user's perspective by the

“what” rather than the “how” e.g. “lectures” not “media player”, “study guide” not “web browser”.

- The integrity of the Shell is maintained by a system layer, containing a Controller object and a reference model, which maps Shell operations to the operating system. The reference model is centred upon a System Tree, defined by the structure of the study guide, as the basis for a directory system, a student model and learning support. The System Tree defines a hierarchical course structure by course title, section, and topic. Each learning component has a directory system mapped to it. Changing the study mode does not change the student's position in the System Tree. Changing the position in the System Tree does not change the study mode.
- The integration of communication, collaboration and extramural learning support into a single distributed database system organised on a mobile computing model. This is mapped to the System Tree to provide the structure for a coarse-grained knowledge base which can be queried along several dimensions.

The next step in the methodology is to use prototyping to further explore and refine the key elements of IMMEDIATE'S design. These elements can then be integrated into a working prototype, including additional applications to test the networking and authoring support. This prototyping work is the subject of the next chapter.

Chapter 6

Prototyping the extramural e-learning system

This chapter discusses the prototyping phase of the research. The primary objective of this phase was to build a working version of IMMEDIATE to test the technical feasibility of the learning computer concept, introduced in Chapter 4, and to make possible a subsequent evaluation with users.

The focus of the prototyping effort has been the Learning Shell, reflecting the primary emphasis IMMEDIATE gives to the student end of an e-learning system. However, in order to evaluate the overall concept adequately, it has also been necessary to develop working prototypes of the course authoring and communications management applications and to integrate them into a functional network.

IMMEDIATE has been implemented using the open-ended *evolutionary*, or *incremental*, prototyping method (Pressman, 1997, pp. 285-288). Beginning with the framework outlined in Chapter 5, the design has been explored and refined through the prototyping process itself. This has involved developing and evaluating the key elements of the design individually. Then the refined elements have been progressively incorporated into an overall working system. This in turn was evaluated and refined under laboratory conditions in preparation for field-testing with users. The results have been codified in a series of documents and in the prototype itself.

To assist in the incremental prototyping, scenarios, use cases, and sequence diagrams have been used to capture key interactions between the user and the Learning Shell interface, and between various system elements which support interface functions (Appendix D). This documentation subsequently provided the basis for the scenarios used to evaluate the prototype with users.

Borland Delphi Professional Version 5.0, running on top of Windows 98, was chosen as the prototyping environment. In order to facilitate rapid prototyping, wherever possible application modules have been assembled from reusable Delphi or third party software components using the Visual Component Library (VCL).

Delphi also offers a fully SQL-compliant database functionality which interfaces with most popular database systems. For prototyping purposes, Paradox tables were used. In conjunction with the Borland Database Engine, they provided the necessary

functionality and could be freely and easily installed on multiple computers.

The key aspects tested in the prototyping process were:

- Identifying the functional requirements for a student user.
- Using a components-based approach to assemble a learning system by transforming individual learning elements into software components.
- Simplifying the interface, by using a system level reference model to embed user calls on the operating system in the learning tasks themselves.
- Refining the overall interface design for simplicity and usability.
- Integrating learning support and communications functionality.
- Resolving network and communications management issues.
- Demonstrating a method for the authoring and re-use of learning material.
- Merging the Learning Shell, the course authoring and the communications management prototypes into a working system.

Discussing each of these aspects in turn provides the framework for the remainder of this chapter.

6.1 Functional requirements for student user

The essential starting point for prototyping IMMEDIATE was to identify more clearly the functional requirements (Bennet et al., 1999, p. 99) which the student end of the system, i.e. the Learning Shell, must fulfil. These requirements emerged through the process of reviewing the literature on distance learning, from personal experience as an extramural student and tutor, from perusing several extramural course study guides, and from a series of informal interviews with people with practical experience in this area. Those interviewed included:

- A number of teachers with experience in preparing and coordinating distance learning courses at the university and pre-university level;
- Several former and current distance learning students;
- An educationalist with research and practical experience in applying technology to learning; and
- A professional instructor in the authoring of web-based university courses.

These requirements were further refined during the process of conceptualising a learning computer. The starting point in the prototyping process was to represent and refine these functional requirements using scenarios (Carroll, 2000). The Start-up scenario is shown in Figure 6.1. All the scenarios are included in Appendix D1.

The functional requirements which the Learning Shell must satisfy can be broadly categorised as those specific to a particular learning element, and those related to more general interactions between the student and the system. Primary attention was given to capturing the latter on the basis that they define the fundamental mechanisms which the Shell must support. They were further documented in the form of use cases and sequence diagrams (Appendix D2 and D3).

6.2 Component-based approach

In the proposed approach the Learning Shell is assembled from reusable "drag and drop" software components with settable properties. Different study modes are defined by particular combinations of components, to which appropriate learning resources are bound.

Prototyping of learning components was broken down into seven steps:

- Identify study modes
- Identify learning elements that make up each mode
- Simulate these elements in Delphi code
- Translate these code modules into reusable Delphi components
- Assemble into learning interface
- Refine for modularity, reusability and ease of maintenance
- Evaluate the results.

6.2.1 Study modes

In Chapter 3 at least six dimensions of learning at the university level were identified (3.2.3). These were learning by textbook, by lecture, by exploration, by collaboration, by doing and by tutorial. Learning by exploration, for example, involved students acquiring knowledge by following their own lines of thought through the course subject matter, as they might browse a library book or other relevant material. These dimensions were used as the basis of the Learning Shell's study modes.

Start-up Scenario

Mary starts up her PC, logs on to Windows using her learner profile, and double-clicks the Learning Computer icon to enter the learning system. The Windows desktop is replaced by the Learning Computer desktop. Mary is now presented with a login screen which welcomes her to Communication 101, displays her username and prompts her to enter her password. She enters her password and clicks the OK button (or hits enter).

Mary has mistyped her password. A message notifies her of this and she is given the option of selecting CANCEL and quitting the Learning Computer, or OK, after which she can re-enter her password. Mary selects OK, enters her password correctly, and a screen appears offering Mary a number of options. She clicks on the Help icon and a page is displayed that explains each of these options thus:

- Return to previous topic The course will open in the study mode and topic that you were in when you last used the system
- Select new topic The Course Explorer will open from where you may choose to explore any available topic and study mode
- Update course material. You should select this option if you wish to update your course resources. You will be presented with the options of updating from a CD-ROM that has been mailed to you, or directly from the University via the Internet
- Restore from backup You should select this option if your course materials have become corrupted. You will be presented with the options of updating your course material from a CD-ROM that has been mailed to you, or directly from the University via the Internet
- Update system settings Select this option if you wish to change your personalised system settings, such as the drives you use for updating, backing up or accessing course materials.
- Exit Exit the course.

Mary selects "Update Course Material". She is asked to nominate whether she will update from a disk or from the Internet. Mary chooses the update form disk option and she is asked to place the disk in the D drive. She does so clicks OK and she returns to the user options screen. (If she had chosen the update from the Internet option, she would only have to click OK and wait for the Learning Computer to connect to the course repository and download any updates to her machine. If no updates are available the system will notify Mary of this.)

If Mary had chosen the "Restore from backup" option a similar process would be followed.

Now Mary chooses the "Update System Settings" option. A screen opens presenting Mary with her current personalised system settings and options for changing them. She opts to change her backup drive to the floppy drive (A), clicks OK and is returned to the User Options screen again. This time she hits ENTER and the course opens at Section 1, Topic 2 in Text Book Mode, which is where Mary was working when she last exited the course.

<End of start-up scenario>

Figure 6.1: Start-up scenario

Learning by doing was further broken down into an Assignment and Practice mode, better reflecting the actual structure of an extramural course. While the Learning Shell as a whole has been designed to support self-paced, exploratory learning, a separate Exploration mode was created to support student browsing of the university library and other supplementary reading sources.

6.2.2 Learning elements

The learning artefacts defining each study mode have been identified through analysis of the different dimensions of university teaching and learning, and of a typical university extramural course, as part of the functional requirements analysis. As discussed earlier (6.1), this analysis drew upon personal experience with extramural and internal university study, perusal of course study guides and web sites, and discussions with university lecturers and online course designers.

The set of scenarios, depicting interactions between a student and the learning computer in each of the study modes, helped further identify learning elements and other necessary screen artefacts (Appendix D).

Sixteen separate learning elements have been specified. They fit into two general categories: mode-specific artefacts that are only used when studying in a particular mode (e.g. a lecture object), and generic objects which students may use in any mode (e.g. a notepad). Three additional interface elements were specified that provide general support to the computerised learning environment (e.g. a User Options object needed for logging on and off the course). All interface components are shown in Figure 6.2.

6.2.3 Delphi code modules

Simulating the learning elements in software required investigating a wide range of implementation methods. Most of the learning artefacts could be grouped by functionality into five basic categories representing generic functions like text editing or viewing web pages (Table 6.1). Each of these categories was implemented as a super class, from which classes representing the actual artefacts were inherited and customised (Appendix E). Many of these super classes were able to be built by modifying existing Delphi and third party software components or sample code.

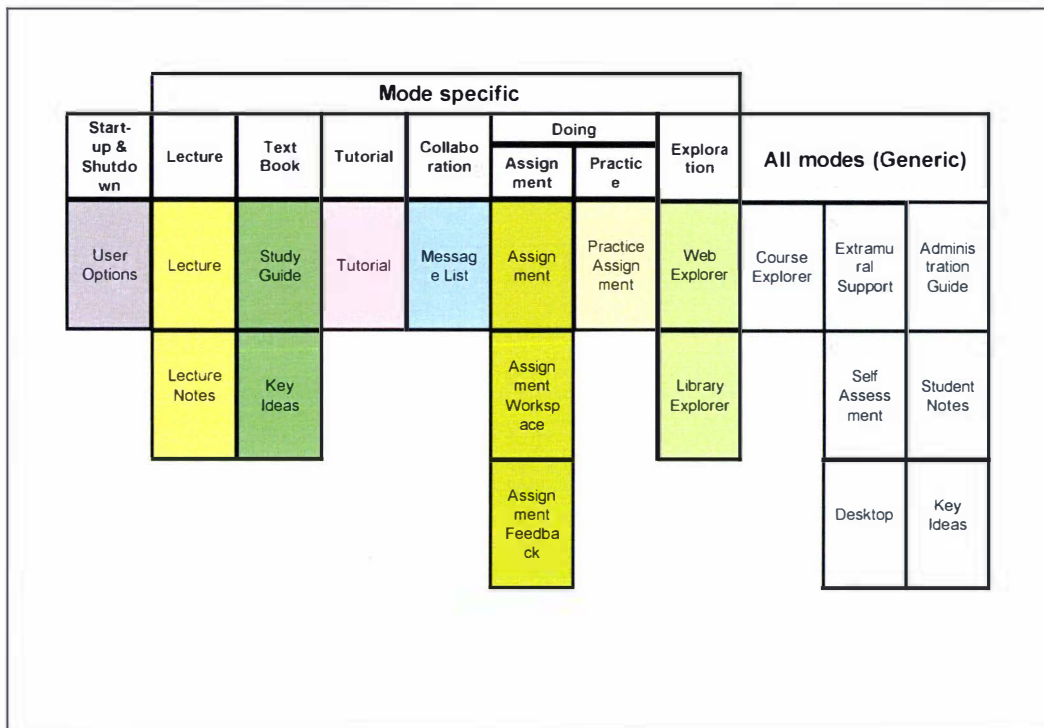


Figure 6.2: Interface components of the Learning Shell

Category	Basic functionality	Implementation
Multimedia viewer	Display multi-media file	Delphi component wrapping Windows media player
Document viewer	Display an HTML-based document	Delphi web browser component encapsulating Internet Explorer engine
Editor	Word processing	Customise sample RTF editor
Browser	Browse selected websites	Delphi web browser component
Launcher	Launch or link to independent application	Third party component that launches executable files
Custom	Various application-specific functions	Build from scratch using basic Delphi components where possible

Table 6.1: Basic learning element categories

A sixth category consisted of the remainder of the learning elements. These had functionality very specific to the Learning Shell, such as the messaging function. These had to be custom-built, using Delphi controls as building blocks wherever possible.

Working versions of all interface artefacts were successfully prototyped. Each interface

element is a self-contained software module that performs a specific task independently of any other.

While the implementation was in some cases complex, involving multiple windows and data files, it has been encapsulated so that each learning artefact appears to the Learning Shell as a single form, e.g. the Message List component (Figure 6.3). When a learning element is required to be displayed, the Shell supplies this form with the path of the learning element's start-up data file and then shows it.

```

var
  MessageList: TMessageList;
implementation
uses vlaController, vlaSystemDictionary, vlaSystemUtilities, vlaMsgData, vlaMessaging,
      vlaAddMessage, vlaReplyMessage, vlaForwardMessage, vlaAllNewMessages;
procedure TMessageList.FormCreate(Sender: TObject);
begin
    inherited;
    MessageSystem := TMessageSystem.Create(self);
    MessageSystem.setMsgList(self);
    MessageSystem.Color := self.Color;
    beforeUpdate := 0.0;
    loadGroupMembers;           // biographies & photos
end;
procedure TMessageList.addNewMessage(addressee: string);
begin
    ...
    NewMessage := TNewMessage.Create(self);
    ...
    NewMessage.ShowModal;
    if NewMessage.ModalResult = 1 then begin           //new message added to database so
        dsMessage.DataSet.Close;                       // show with new message added
        dsMessage.DataSet.Open;
        self.Paint;
    end;
end;

```

Figure 6.3: Message List component encapsulates multiple forms.

6.2.4 Reusable learning components

Delphi components are software modules which encapsulate a single logical function or group of functions. This functionality may be relatively simple as in a list box, or very complex as in a complete web browser. There are two forms of components recognised by Delphi and similar visual programming environments – those that can be "dragged and dropped" from a component palette onto a base form, and those that can be copied or inherited from a form template stored in the Object Repository. Once a component has been added to a project it is then customised by setting its properties. A Delphi component may also be extended by a programmer to provide additional functionality (Inprise, 1999a, Ch. 2 & Ch. 31).

Learning elements are software modules which perform a single logical learning system service or related group of services, and provide a standardised interface to the rest of the Learning Shell when this service is required. To maximise modularity and reusability, these elements do not directly interact with each other. Instead, each interfaces to the Shell Controller object, which manages the interface and passes requests for services between interface components and the system layer.

Because the Shell is Windows-specific, each element can directly call on the Windows API (libraries), e.g. to draw itself on the screen, open data files, etc. These calls are only managed through the Controller where necessary to maintain the integrity of the Learning Shell by avoiding directly exposing the operating system to the user.

Due to the careful modular design of the learning elements, embodying them in a single parent form (Appendix F10), transforming them into reusable Delphi components is quite straightforward. Delphi provides a template for writing the source code for wrapping the learning element module, including defining any settable properties and their default values (Figure 6.4). Each learning component implements a `createLearningComponent` method which creates an instance of the component's parent form, which, in turn, creates any subsidiary forms on an as-needed basis (Figure 6.4). Delphi uses this template to compile and install the modules as visual components on the VCL palette (Figure 6.5).

Two levels of learning components have been prototyped. A base level component encapsulates a super class such as an RTF editor and must be dragged onto a form and then customised. An implementation-ready component wraps a fully functional learning element, such as Assignment Feedback, and needs only to be dragged onto a form.

6.2.5 Learning Shell assembly

For a fully operational Learning Shell to be assembled from learning components, an application template is required that embodies all the logic needed to manipulate these components correctly. This has been achieved using Delphi's form template facility.

The Learning Shell appears to a learning system application programmer as a component in the Delphi Object Repository. This component creates the Learning Shell background screen or Desktop. The Desktop form encapsulates all the system functionality of the learning computer. By inheriting this form from the Repository, and dragging the desired learning components onto it from the VCL palette, a full Learning Shell can be built (Figure 6.5).

```

unit vlmFeedback;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  vIaComponent, vIaFeedback;
type
  TvImFeedback = class(TvIaComponent)
  public
    procedure createLearningComponent; override;
  end;

procedure Register;
var
  Feedback: TFeedback;
implementation

procedure Register;
begin
  RegisterComponents('VLM', [TvImFeedback]);
end;

{ TvImFeedback }

procedure TvImFeedback.createLearningComponent;
begin
  inherited;
  Feedback := TFeedback.Create(application);
end;
end.

```

Figure 6.4 : Code to wrap Feedback learning element as Delphi component.

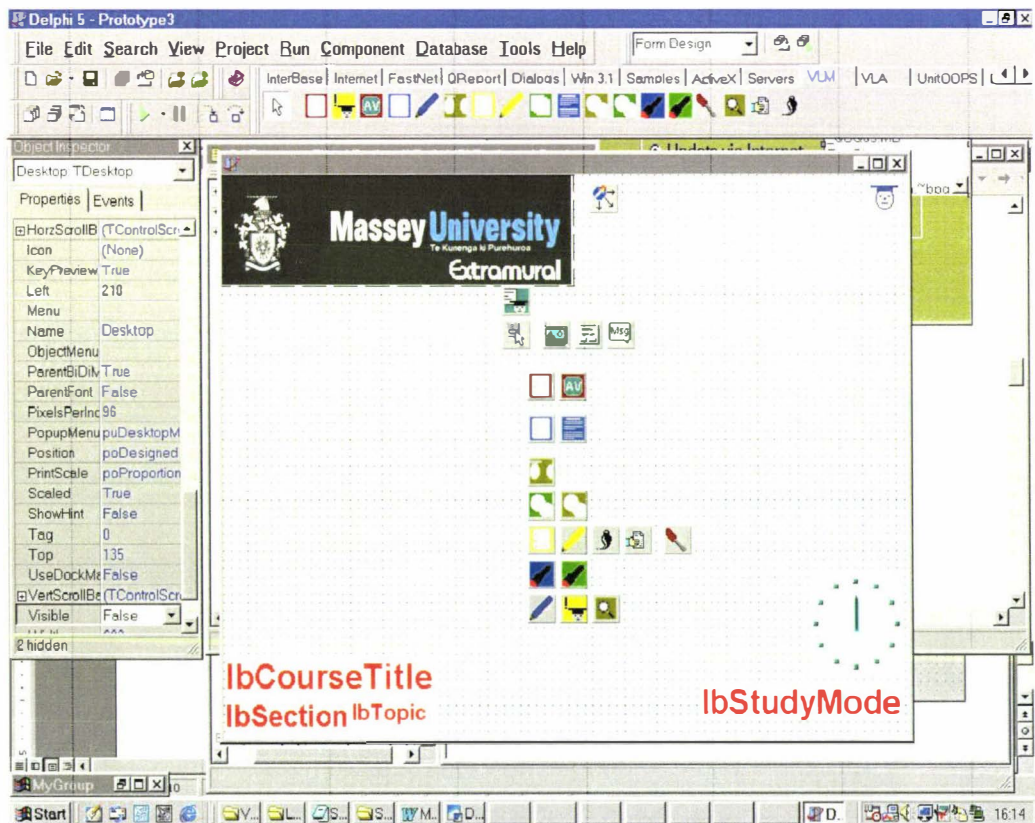


Figure 6.5: Shell assembled by dragging components onto Desktop form.

This encapsulation has been implemented in the following manner.

By default in Delphi, an application object creates all the main objects as instances of global variables, usually forms, which are then available from anywhere in the application. In the Learning Shell, the application only instantiates a Desktop object, which in turn creates all the Shell objects through its constructor method, including the principal forms of all learning components (Figure 6.6). The Controller and all interface components are created as instances of global variables accessible from anywhere in the application. System objects, such as the reference models, are created by the Controller object as instances of local variables, and can only be accessed through the Controller's public methods.

All these implementation details are hidden from the application programmer.

```

procedure TDesktop. FormCreate(Sender: TObject);
begin
    controller := TController.Create(self);
    CourseExplorer := TCourseExplorer.Create(self);
    StartUpDesktop := TStartupDesktop. Create(self);
    UserOptions:= TUserOptions.Create(self);
    ResourceMenu := TResourceMenu.create(self);
    SystemLog:= TSystemLog.Create(self);
    ExtramuralSupport := TExtramuralSupport.Create(application);

    createComponents; // that have been dropped onto form
    ...
end;

procedure TDesktop. createComponents;
var
    i : integer;
    comp: TvlaComponent;
begin
    for i := 0 to self.ComponentCount - 1 do
        if (self.Components[i] is TvlaComponent) then begin
            comp:= self.Components[i] as TvlaComponent;
            comp. createLearningComponent;
        end;
end;

```

Figure 6.6: Desktop encapsulates all Shell components.

At the time when the Learning Shell (Desktop) component is compiled and added to the Object Repository, it is not known what learning components will be added to a particular application. In fact, some entirely new components may be written and added by a learning system programmer. Therefore, the names of learning component's principal forms cannot be included in the source code of the Shell component, which is not available to the programmer.

To enable the Controller to manipulate the forms created by learning components, the parent forms of all learning components must follow a naming convention. The name must:

- be unique and not be used by any other component in the application;
- use upper case characters rather than spaces or underlines to separate words; and
- be meaningful, so that it can be parsed as the component form's caption.

The programmer must add this name to a file 'components.txt' which the System Model (Section 6.2.3) uses to construct a list of learning components. This list is used by the Controller to reference the actual screen forms through a *processForm* method (Appendix F1).

6.2.6 Refinements to promote consistency and modularity

A number of refinements have been made to promote consistency throughout IMMEDIATE and to enhance modularity, reusability and maintainability:

- A System Dictionary unit has been created to define constants such as system directory paths and other values that are used in different modules in the Learning Shell and its supporting applications. In this way, changes only need to be made once to be reflected throughout the system (Appendix F3).
- A System Utilities unit (Appendix F4) has been provided containing frequently-used functions and procedures that can be accessed by classes and units anywhere in the overall IMMEDIATE project.
- Another "write once, use anywhere" approach was found to be useful for some frequently-used routines, such as file and folder manipulation, that may involve complex direct programming of operating system API calls. This was to implement the routines as the properties and methods of non-visual Delphi components. As components, they may encapsulate existing Delphi controls, whose methods implement some or all of the required API calls. As non-visual components, they may be dragged onto forms where they are accessible to the application, while remaining hidden from the user.
- The Controller API has been made available to component programmers. This enables components to utilise a wider range of system functionality and to enable some information to be passed indirectly between components in a uniform manner (Appendix F1).
- A comprehensive Delphi object hierarchy has been defined for the Learning Shell, to support consistency and reusability throughout the application. The object

hierarchy defines the inheritance paths for all the approximately 80 classes making up the Shell (Appendix E2).

6.2.7 Feasibility of component-based approach

The process of prototyping and refining the classes and components making up the Learning Shell is documented in Appendix E.

The prototyping has confirmed that it is technically feasible to assemble an extramural e-learning system using a component-based approach. The experience of actually implementing the learning computer prototype has shown that this approach allows for a very large degree of flexibility in defining and redefining learning dimensions and components.

Working components supporting all the defined modes were successfully implemented. These provided all the learning functionality offered by typical web-based courseware systems. Moreover, additional functionality not generally offered by courseware, such as one-on-one adaptive tutorials, was also successfully prototyped and integrated into the Shell. And, the modular design of the learning components demonstrated a mechanism by which existing components can be easily updated, or replaced by ones with more sophisticated functionality.

At the programmer level, reusability is achieved by adopting the object-oriented template and component approach of "drag and drop" Rapid Application Development environments.

6.3 System level support

The third major prototyping task was to evaluate a method for embedding user calls on the operating system in the learning tasks themselves, thereby rendering the operating system invisible to the student.

The proposed method, as discussed in Chapter 5, was to create an intermediate layer between the Learning Shell interface and the operating system which models the course structure (5.5.1). This system layer tracks the student through the course and maps user interactions with Shell components to the correct files and objects within Windows. A *System Tree* defined by the course Table of Contents is central to this reference model.

The prototyping steps followed were:

- Implement the System Tree data structure.

- Identify system level operations required to support the Learning Shell.
- Specify a modular structure to best support these operations.
- Implement system modules as classes encapsulating the necessary data types.
- Refine for modularity, reusability and ease of maintenance
- Evaluate the results.

6.3.1 System Tree

A table of contents is a shallow tree structure of sections and topics. This is easily represented by a list of lists. This implementation accommodates additional levels such as attaching a list of learning goals or key concepts to each topic (Figure 6.7). The System Tree is made visible to the user through the Course Explorer interface component (Figure 6.15)

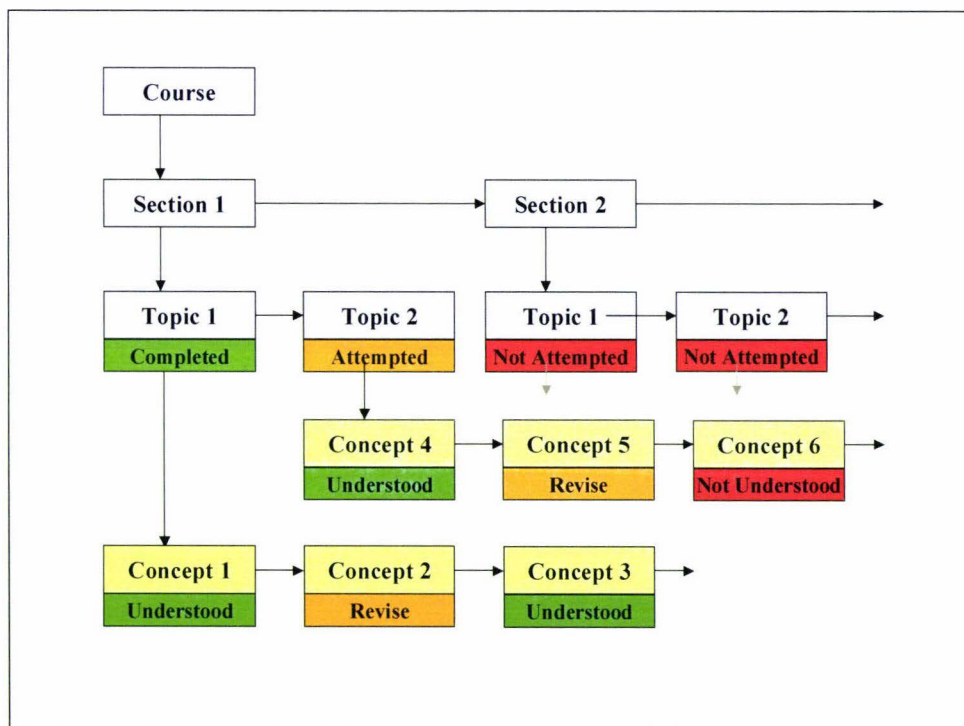


Figure 6.7: System Tree with additional concept level.

6.3.2 System level operations

At the heart of the learning computer concept is navigation – tracking and storing the user's current position and taking them to any new position they select. An initial list of operations (Table 6.2) needed to support the Learning Shell's navigational logic was

extracted from use cases and interaction diagrams. This included storing and providing information about the current state of the system and of the student. The sequence diagram for the Change Mode use case is shown in Figure 6.8. Further sequence diagrams are included in Appendix D3.

System Level Operations
Initialise System Model
Save System Model
Initialise student model
Save student model
Get current student position (mode, section, topic)
Set current student position (mode, section, topic)
Get current resource (section, topic, component)
Set current resource (section, topic, component)
Get topic status – not started/attempted/completed
Set topic status – not started/attempted/completed
Save student work (section, topic, component)
Change Study Mode (new_mode)
Change Topic (new_Section, new_Topic)
Get Component List (selected_mode)
Open (this_component)
Close (this_component)
Set Revision Mode

Table 6.2: System level operations defined from use cases.

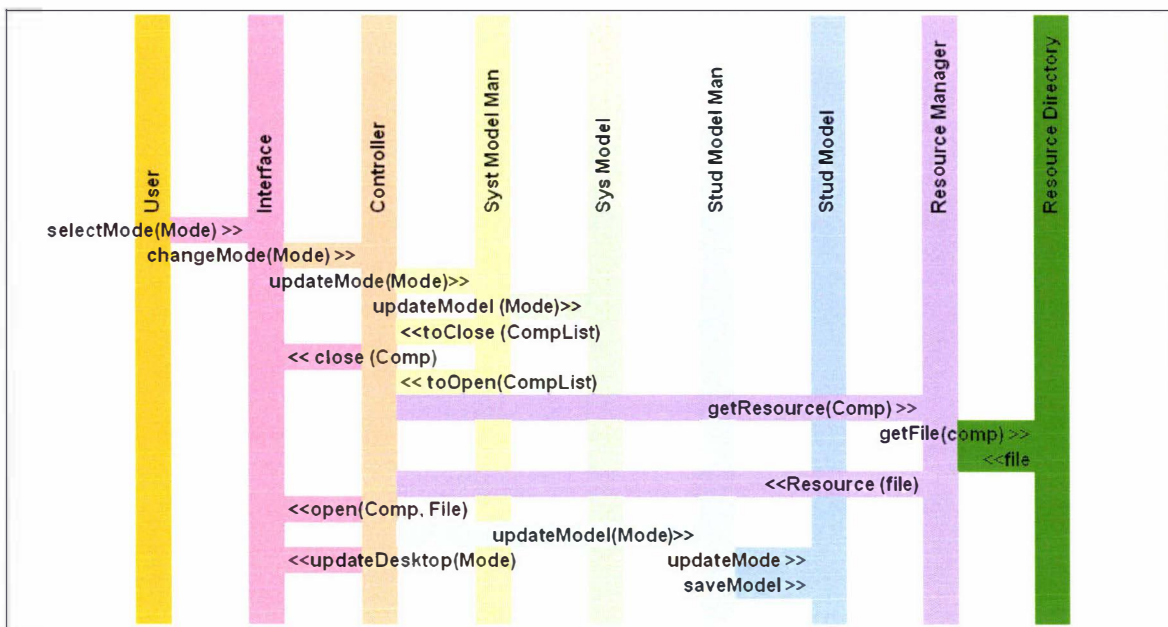


Figure 6.8: Sequence diagram for Change Mode Use Case.

6.3.3 Reference model specification

In addition to the *System Tree* object it was found to be advantageous to subdivide the reference model into modules with specific responsibilities. It was also more efficient to store certain information about the student, such as what topics they had completed, with the system data structures themselves, rather than in the separate Student Model.

The *System Model* object stores information about each study mode and learning component offered by a course. For example, it knows whether the data displayed by a component can be edited and saved by the user or is read-only. The System Model provides the information for swapping between study modes within the course.

The *Resources Model* object encapsulates the directory system that contains course learning materials. It maps a learning component to its correct resource file, if one exists. For any topic in the course, it provides a list of all components for which resources are available. The Resource Model must map to the actual directory structure to provide for updates, insertions, deletions and graceful recovery from errors (e.g. file not found).

The *Student Model* object stores the student's current position in the course (i.e. section, topic and study mode) and any other data necessary to individualise the course to a particular user that would not otherwise be recorded.

6.3.4 Implementation of data structures

The class implementations for the reference models are generally straightforward. Each model class is defined as a data type with a set of public operations, and contains the data structure and additional functions and procedures needed to implement these operations.

Most of the data types involve the manipulation of lists of strings, which are implemented using arrays. Custom string list classes were used, rather than those provided by Delphi, to enable more direct support to the required operations. To initialise and permanently update the reference models, the data structures are stored as text files (Appendix F13) in the Models sub-directory of the System folder (Figure 6.9). The complete folder system for the Learning Shell is shown in Figure 6.10.

The Resources Model encapsulates the Resources folder structure. Files stored in the root Resources folder follow a naming pattern indicating the learning component they belong to and their position in the course. This file contains the names of all the resources associated with a particular component at a given point in the course. These

resources are stored as files in that component's subdirectory in the Resources folder (Figure 6.10).

For example, "s01t02le.txt" is the file associated with the lecture component in Section 1.02 of the course (Figure 6.11). This file contains the name of a video file and of an image file of the lecturer making the presentation, which are located in the Lecture subfolder of the Resources directory. The lecture component obtains this information from the Controller when its show method is called (Figure 6.12).

The Resources Model uses algorithms based on this naming pattern to return the file path for a component, and other related operations.

The Controller object uses the information supplied by the reference models to map learning components to Windows forms, pass the correct data files to those forms, and then show them to the user (Appendix F1).

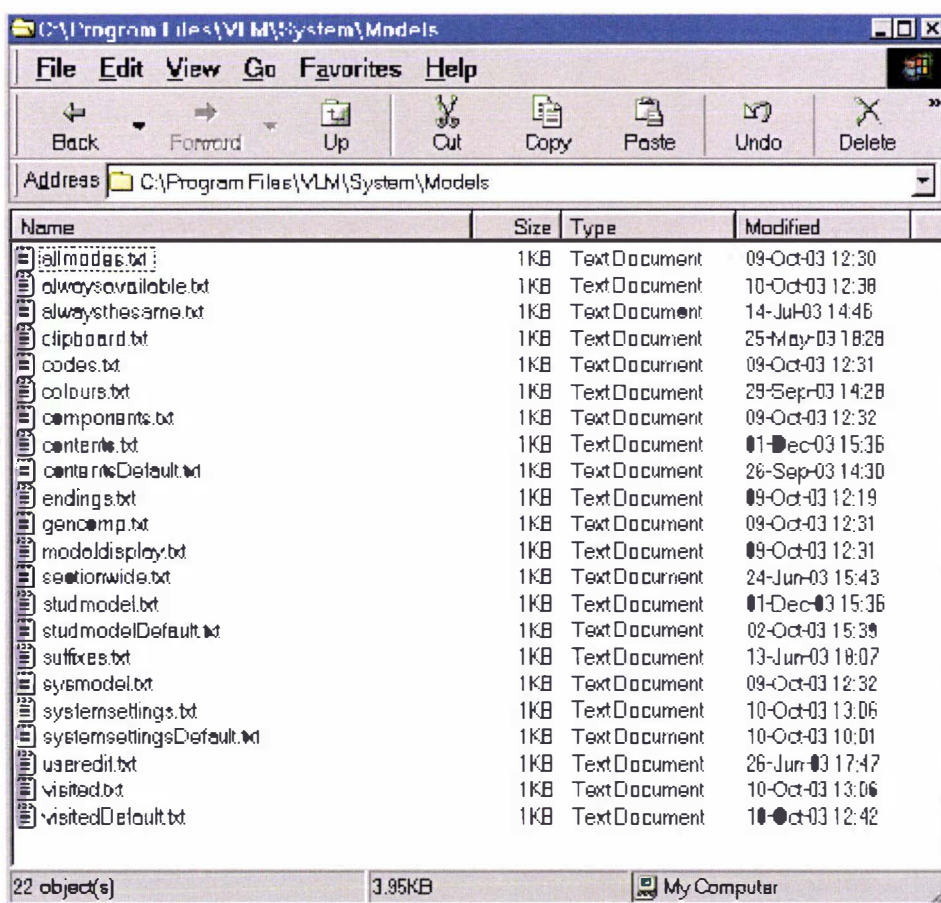


Figure 6.9: Reference models stored as text files in Models directory.

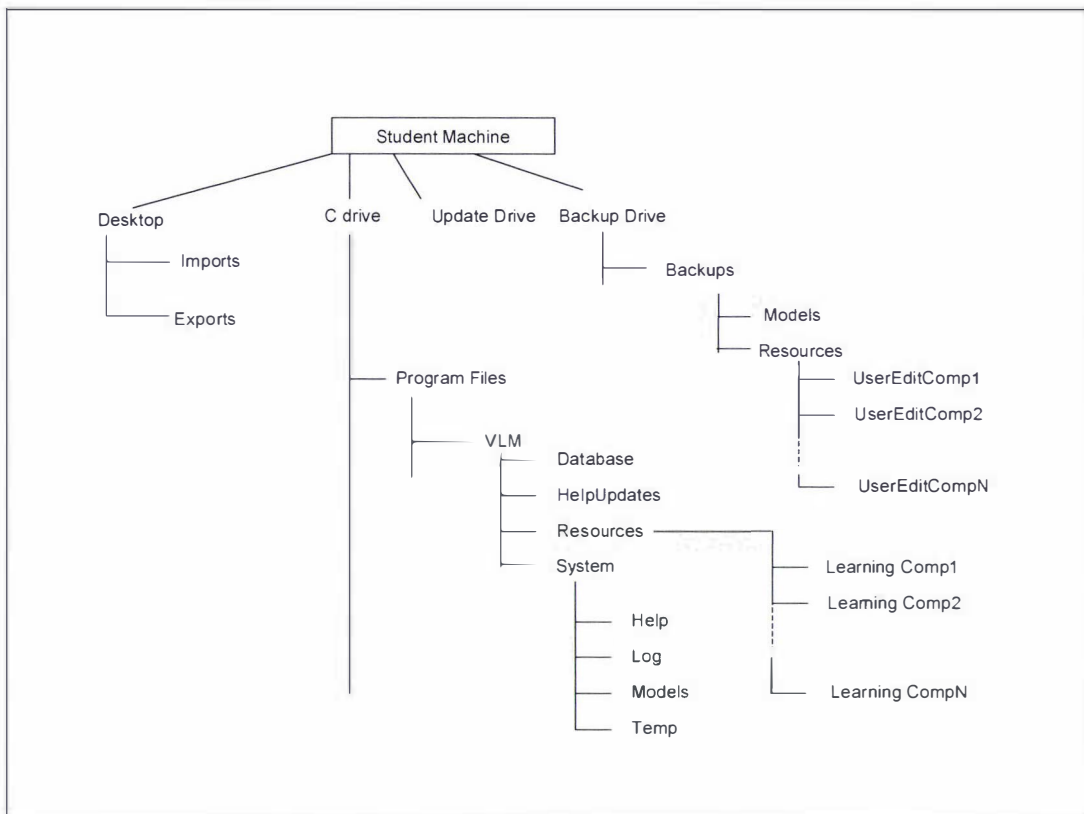


Figure 6.10: Learning Shell directory structure.

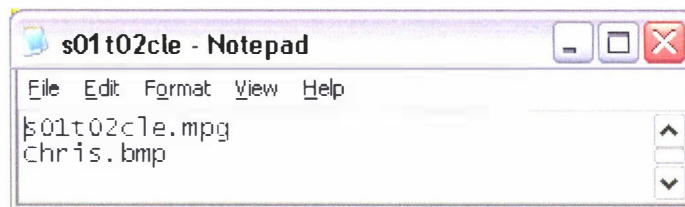


Figure 6.11: Content of “s01t02cle.txt” in Resources directory.

6.3.5 Refinements to system modules

Through the prototyping process the modular design of the system layer was refined to improve reusability and make modification easier. Figure 6.13 shows the Learning Shell’s refined architecture.

A model manager object was introduced to remove any dependencies between the Controller object and each model, for easier modification of the prototype. This was important because implementations that use Windows controls, e.g. a hidden form, are not interchangeable with custom-built classes that make no calls on the operating system libraries. The interfaces for the model managers are included in Appendix F.


```

procedure TLecture.FormShow(Sender: TObject);
begin
    ...

    // set lecture and image file paths
    getFileInfo;

    //get image
    imLecture.Picture.LoadFromFile(imageFile);

    //display message to user
    StatusBar1.SimpleText:= caption+' Click picture to load.';

end; //of formshow

procedure TLecture.getFileInfo;
var
    infile: textfile;
    line: string;
begin
    filepath:= controller.getFilePath( self.name );
    try
        AssignFile( infile, filepath );
        Reset ( infile );
        readln( infile, line );
        lectureFile := LECTURE_DIR+ line;
        readln( infile, line );
        if not eof( infile ) then
            imageFile:= IMAGE_DIR+line
        else begin
            imageFile:=IMAGE_DIR+DEFAULT_IMAGE;
        end;
        closeFile( infile );
    except
        showVLAMessage( 'Could not load lecture.' );
        exit;
    end;
end;

```

Figure 6.12: Component *show* method gets data filepath from Controller.

Mechanisms were successfully added for saving and backing up student's work, and performing management tasks like logging on and off, with minimal input from the user.

The reference models were modified to also support IMMEDIATE's authoring and course management responsibilities. The model manager layer provides the interface to the authoring application, passing different parameters to the model layer depending on whether it is being used for learning or authoring. (See, for instance, the System Tree Manager class interface in Appendix F8).

6.3.6 Embedded operating system

The creation of an intermediate system layer between the Learning Shell interface and the operating system, successfully embedded operating system operations in the task domain. In particular, by direct manipulation of the Course Explorer, it was possible to

navigate to any point within the course, and to change from one study mode to another, with the system automatically handling the necessary opening and closing of files.

However, during the course of prototyping the Shell it became clear that to manually define a course, and then correctly add learning materials, was a complex task requiring a thorough understanding of the Learning Shell's implementation. For the learning computer approach to be a viable alternative to conventional courseware, a method had to be demonstrated for a non-programmer to author and update a course without being exposed to the internal complexities of the system. This issue is addressed further in the section on the authoring prototype (6.6).

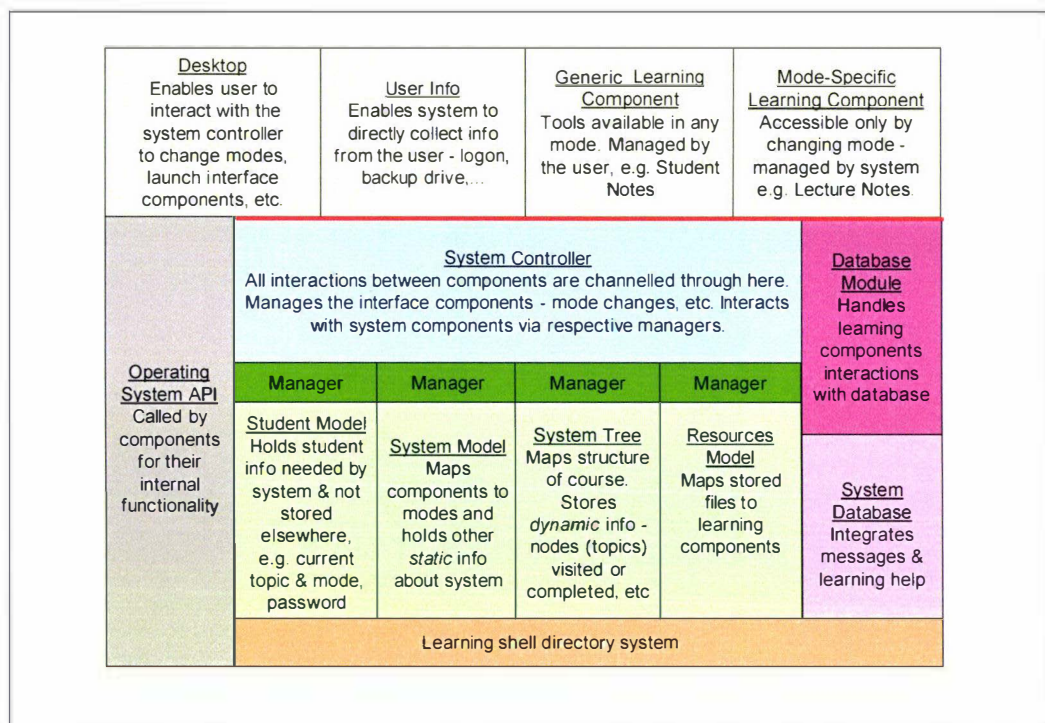


Figure 6.13: Learning Shell architecture.

6.4 Interface Design

Successful e-learning requires designing the computer environment for form as well as content (Smulders, 2003). In this section the overall look and feel of the Learning Shell is described. Important considerations shaping the design and refinement of the interface are discussed using Nielsen's usability heuristics (Figure 6.14) as a framework.

In the initial stages of developing the Learning Shell, the primary focus was upon implementing the individual interface components and the underlying mechanisms

needed to support the special learning computer functionality. As new learning components were developed and tested, they were added to the prototype one-by-one. Similarly, new system functions and their corresponding interface artefacts were added step-by-step.

While consideration was given to the general design of the interface during these stages, it was inevitable that some inconsistencies and redundancies would be introduced that inhibited usability. Therefore, once these objectives had been achieved, the prototyping emphasis shifted to the design and refinement of the overall environment provided by the Shell. In this phase, consideration had to be given to the general principles of good interface design *and* the special requirements for a learning computer.

1. Visibility of system status	6. Error prevention
2. Match between system and real world	7. Recognition rather than recall
3. User control and freedom	8. Flexibility and efficiency of use
4. Consistency and standards	9. Aesthetic and minimalist design
5. Help users recognise, diagnose recover from errors	10. Help and documentation

Figure 6.14: Nielsen's usability heuristics (Nielsen, 1994).

6.4.1 Learning Shell look and feel

The concept of the learning computer entails a domain-specific, minimalist interface in which many functions normally managed by the user are managed by the system. The emphasis is on usability and simplicity. In line with this minimalist design, the Shell does not provide many of the features and shortcuts that experienced Windows users have come to expect.

The Learning Shell provides a direct manipulation graphical user interface (Shneiderman, 1998, p. 71), centred upon the use of a mouse pointing device. User interactions with the Learning Shell are documented in the scenarios outlined in Appendix D1. Additional screen shots of the Shell can be found in Chapter 7 (Figures 7.2 – 7.13) and Appendix G.

The Learning Shell replaces the Windows GUI with its own special-purpose one. In this way it provides a minimalist interface with just the functionality that the student needs for his/her current learning task. However, Microsoft does not directly support the

development of alternative GUIs for its operating systems, by providing programmers with source-code and kernel-only versions, as Linux does. Therefore, the Learning Shell could only be implemented by “tricking” Windows. It has been implemented as a Windows application with some special properties that shut out, rather than replace, the Windows shell in the following manner:

- Upon opening, the Learning Shell overlays the Windows Desktop with its own Desktop screen, and disables all the special Windows system keys on the keyboard. In this way, all keyboard and mouse interactions are confined within the Learning Shell environment. The Desktop is implemented as a maximised, borderless window. Disabling the system keys is activated by a method in the System Utilities unit (Appendix F4) and is achieved by a call to the Win32 API. The keys are re-enabled by a similar method when the user exits the Shell.
- To remain visible and accessible to the user, all learning components within a study mode must remain above the Desktop, even when they are not the immediate focus of the learner’s attention. This is achieved using Windows forms’ FormStyle property. The Desktop’s form style is set to normal, and the style of the base form inherited by all learning components is set to always stay on top, i.e. above the Desktop (Appendix F10).
- Conventions for the design of components in the form of inheritance hierarchies, templates and guidelines have to be upheld by component programmers to avoid exposing the operating system, e.g. through file saving dialogues.

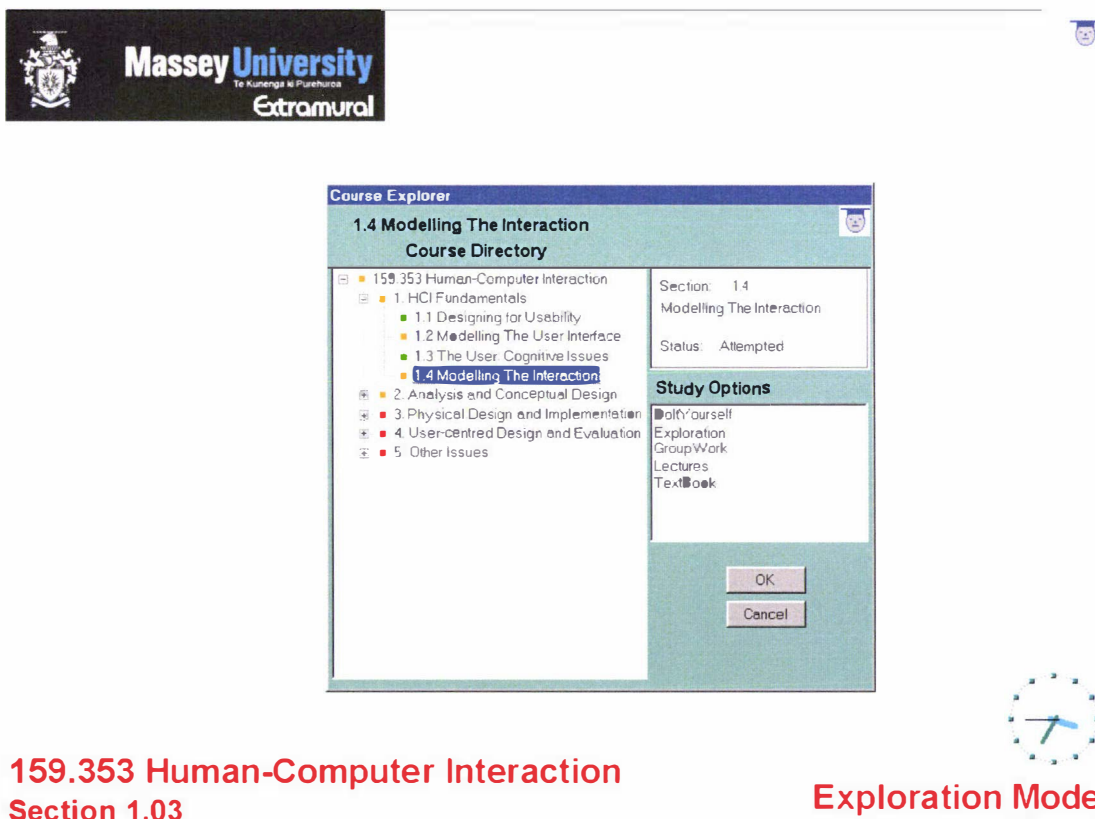
Desktop

The Learning Shell environment is entered by clicking an icon on the Windows Desktop. The Windows Desktop is then replaced by the Shell Desktop displaying a logon form. Once the user has logged on, the logon form is replaced by a User Options form. From the User Options form the user can complete housekeeping tasks such as updating the course content, return to where they were in the course when they last logged off, or navigate to a new position via the Course Explorer.

The Desktop (Figure 6.15) provides a plain background for the Shell. The only items displayed on it are the university logo and course title, a help icon, and the current section, topic, study mode and time. All other features must be accessed through a Desktop Menu which pops up when the Desktop is right-clicked with the mouse. A key aspect of the Learning Shell’s minimalist interface is that all its functionality is accessible by memorising just three paths:

- Pressing the F1 key brings up a Help screen for the selected interface component (Figure 6.16). Clicking on its Help icon has the same effect. Performing this action when the Shell Desktop is selected brings up help on the system as a whole.
- Right-clicking with the mouse on the Desktop pops up the Desktop Menu, from where the user can navigate forward and backward through the course, open the Course Explorer, access additional learning tools like Student Notes, get Help, or exit the system.
- From the Course Explorer (Figure 6.15), the student can navigate to any topic, and change to any study mode within that topic.

All system-related features except the Help icon are hidden when the user is engaged in a learning task.



159.353 Human-Computer Interaction
Section 1.03

Exploration Mode

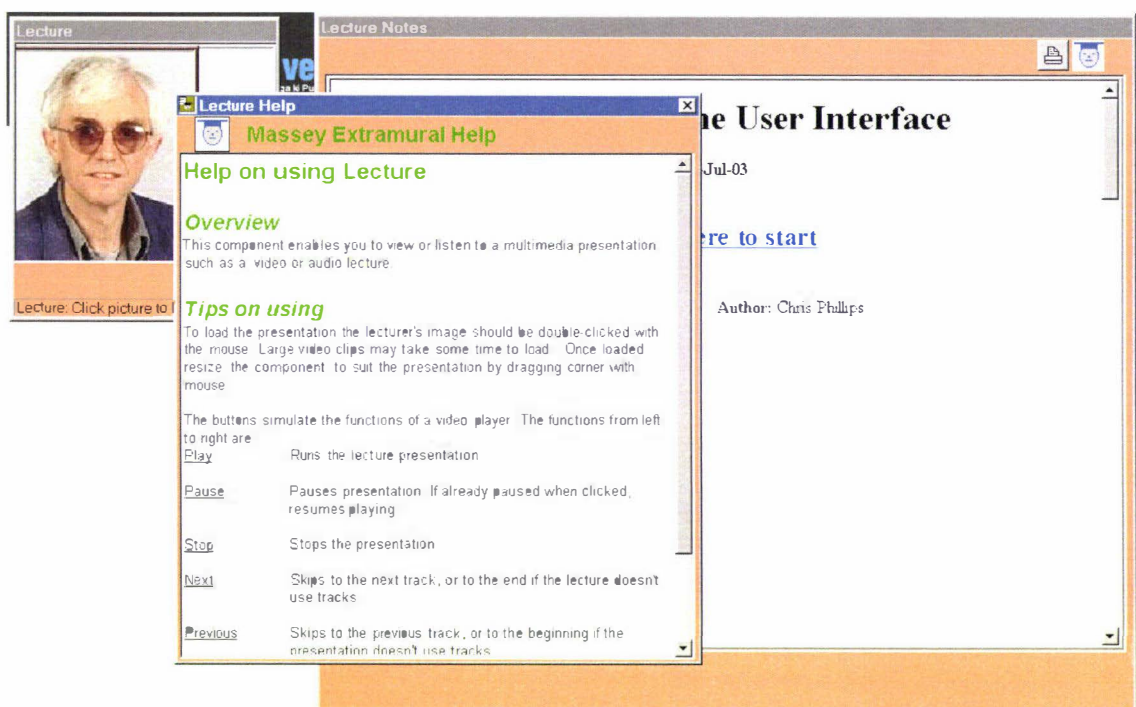
Figure 6.15: Desktop with Course Explorer open.

Course Explorer

The most important interface feature is the Course Explorer (Appendix F2), through which the learner navigates their course. It has been implemented using Delphi's TTreeView component, which uses the graphical control developed by Microsoft to display the Windows directory system.

The Course Explorer provides an example of how the Learning Shell implements many of the qualities emphasised by Nielsen such as visibility of system status, user control and freedom, recognition rather than recall, and flexibility and efficiency of use.

When the student opens the Explorer, the course table of contents is visible in the form of a collapsible tree. The student's current position, i.e. topic, in the course is displayed. The current section node is expanded to show all its child (topic) nodes (Figure 6.15). The Explorer also displays separately a list of all the study modes available for that topic. This design draws upon the findings of Larson et al. (1998) that broader and shallower menu structures are quicker and easier to use than narrower and deeper ones, which have more levels to navigate.



159.353 Human-Computer Interaction

Topic 1.02

Lectures Mode

Figure 6.16: Help screen for Lecture component.

Whenever a topic is selected in the tree, the list of available study modes is updated by the controller from information supplied by the Resource Manager. Clicking an OK button or double-clicking a topic or study mode, passes the selected items to the controller for processing as a Change Topic, Change Mode, or Change Mode and Topic, event. Colour-coding, based on a traffic light metaphor, is used to show the student's progress through the course, in line with Najjar's recommendations on the effective use of colour (Najjar, 1990). It shows which sections and topics have been

previously attempted (amber), which have been successfully completed (green), and which have not yet been attempted (red). The status codes are stored in the System Tree (Figure 6.7) and are updated whenever the student carries out some self-assessment tasks, as is discussed in Section 6.4.

6.4.2 Interface Refinement

The goal in refining the Learning Shell interface design was to have the computer vanish as users become absorbed in their task domain (Shneiderman, 1998, p 18). This requires a user-centred-design that minimises presumptive, controlling, or intrusive features (Murray et al., 2000).

The framework for refining the interface was a self-evaluation using Nielsen's usability heuristics. As a result of the evaluation, a number of modifications were made to individual interface objects, to how they appear in relation to one another, and to the underlying system functionality of the Shell.

In the remainder of this section, some important features of the refined interface are discussed by reference to each of Nielsen's heuristics.

Visibility of system status

While the Learning Shell interface is minimalist, this is not at the expense of transparency. The student's current position in the course is displayed on the Shell Desktop. This display is updated, via the Controller, whenever the screen is repainted.

Visual cues are used where operations may take seconds or even minutes to complete (e.g. loading a video file). An egg-timer cursor is wrapped around all such routines, and a sequence of messages conveying the status of the selected operation are displayed in the component window's status bar.

Match between system and real world

On the understanding that "old media serve as metaphors for new ones" (Collins, 1995, p. 223), the Learning Shell implements, in electronic form, the basic artefacts of the correspondence-based extramural course and of the core modes of university study. In this way, the Shell seeks to create a familiar and intuitive environment for the student. However, elaborate graphical representations of real world objects have been rejected in favour of simple, plain screens.

User control and freedom

A number of devices have been introduced to keep the user informed and in control, and to extricate themselves from unexpected situations. The most important of these is the Course Explorer, which enables the user to rapidly navigate to and from any point in the course.

The Shell strongly supports user-driven self-learning. But to maximise freedom of learning it places greater constraints on what the learner can do as a computer user, compared to the Windows environment. Nevertheless, while managing many tasks for the student, the user is always informed of any significant change and given the chance to cancel the action before it becomes irreversible.

It is sometimes essential that the user completes one task, encapsulated by a particular interface component, before moving onto another e.g. when logging on to the course or completing a self-assessment to register progress through a topic. This requirement is recorded in the System Model. In such cases, the user's freedom of choice is constrained by the Controller presenting the component's parent form in a modal state. This means that the user must do what is necessary to close the form, i.e. to complete the task, before being able to access anything else on screen.

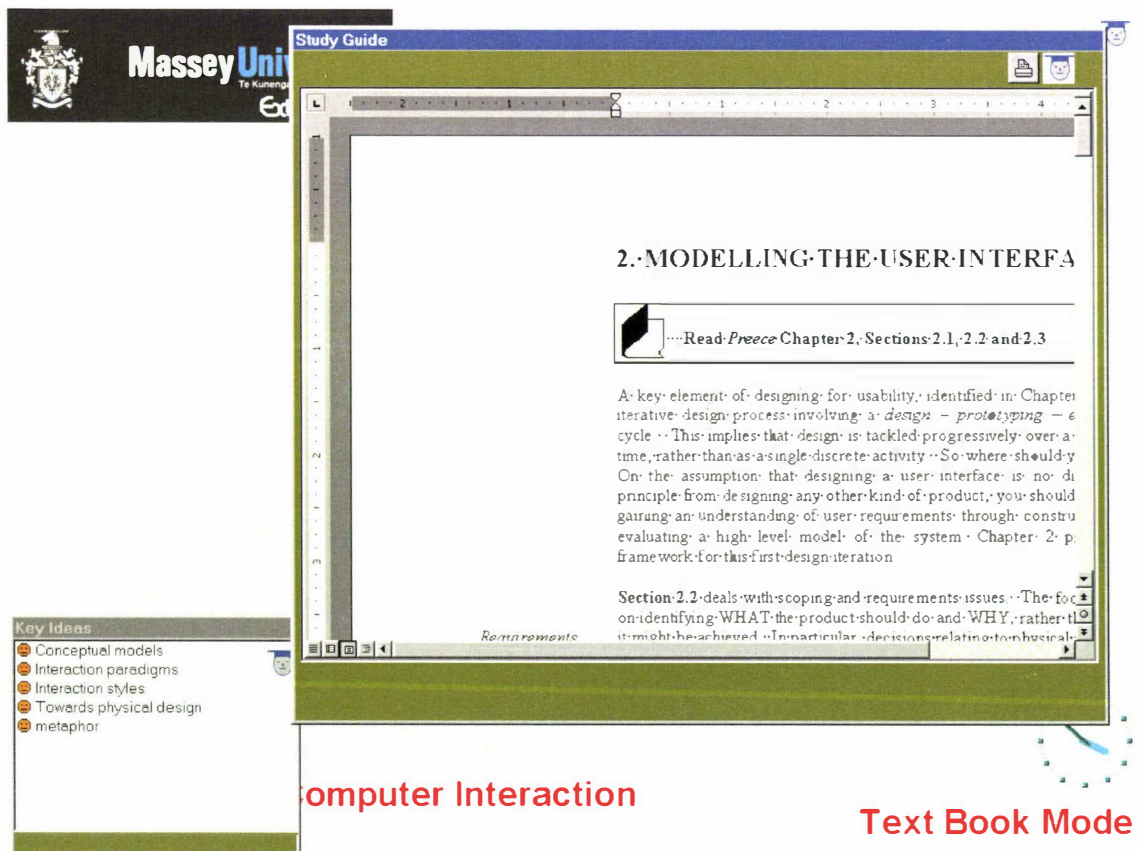


Figure 6.17: Mode components are colour-coded.

Consistency and standards

A consistent, minimalist look and feel to the interface has been enforced through constraints built into the implementation of learning components. This includes the size, positioning and labelling of widgets, the captioning of screen forms and the use of custom dialogue boxes to display system messages. Wherever possible, these constraints have been embedded in super classes within the Learning Shell object hierarchy. A new learning component then inherits these constraints via a form template within the Delphi Object Repository. The source code for the learning component super class is included in Appendix F10.

To help visually distinguish study modes, it was decided to colour code their components (Figure 6.17). Research such as that of Wright et al. (2001) has shown that appropriate use of colour can aid user memory and facilitate the formation of effective mental models. A colour-coding scheme is stored in the System Model and passed to a component at run-time. To allow for a component to be shown in different modes, it obtains its colour-code from the controller whenever its paint method is called. System interface objects have their own colour-code as well.

Help users recognise, diagnose, recover from errors

Because the Shell tracks the learner through the system, and has a simplified, transparent structure without a myriad of hidden features, it is able to utilise a few simple strategies to prevent the user getting lost and to help them to identify and recover from errors. This includes always clearly displaying the student's study mode and current position in the course, visualising the entire course in the Course Explorer, and providing just-in-time, just enough help for the interface component with which the user is currently interacting.

The Help screen for each component emphasises a "handy hints" approach for how to use that component.

Error prevention

To prevent errors, for all critical actions – e.g. the saving of student work such as an assignment before shutting down, or modifying settings such as how the system will be updated from the Repository – dialogue boxes are displayed that provide the user with a clear description of the selected course of action and the option to confirm or cancel that action.

However, the major error prevention strategy has been to manage the user's environment so as to minimise what the learner has to remember or do vis-à-vis basic computer tasks. All essential functionality can be accessed by right-clicking the Desktop to pop-up the Desktop Menu, or by pressing the F1 key, or clicking the Help Icon on each component, to access context-sensitive help.

Some functionality required by learning components (e.g. word processing or web browsing) was available ready-made as fully-featured Windows ActiveX controls with which the Shell could interface. However, maintaining the integrity and simplicity of the Shell was problematic because these controls were separate applications, which exposed the user to more features than were required for their learning, and could provide a path for exposing the user to the underlying Windows environment. Where possible it was found to be better to use a Delphi component which "wraps" an ActiveX control, hiding its functions, or to produce a simpler version of the control by adapting and re-using sample code. These approaches enable the system developer to provide a customised component, making available to the user only those features needed to achieve the desired learning task.

Recognition rather than recall

To aid recognition rather than recall, a design consistency is maintained across all interface components and study modes. Wherever possible the functions of interface controls like buttons are suggested by appropriate icons and labels. All features are readily accessible from the Desktop, without the need to memorise or navigate complex pathways to them.

Flexibility and efficiency of use

The most important way that flexibility and efficiency of use has been supported by the Shell, is in the ability of the student to rapidly and effortlessly switch between study modes, between different topics in the course, and in and out of the system altogether.

To maintain simplicity, there is generally only one way for the user to activate a particular function, mostly through clicking a screen item with the mouse. However, some standard shortcut features have been introduced. All learning components support the Help, Copy, Cut, and Paste accelerator keys. And the student may use the Desktop Menu to move back and forth between adjacent topics.

Because each component of a study mode is opened in a separate window, the interface could appear somewhat disjointed. "Docking" was introduced to address this

problem. This involved adding the capability for individual components of a mode to be dragged into position with the mouse, and docked together into a single composite window (Figure 6.18).

Aesthetic and minimalist design

There are two categories of interactions the user has with the Shell – those associated with learning tasks, and those associated with setting up and maintaining the system. To simplify the user's environment, and minimise distractions during study, maintenance tasks such as changing system settings, updating resources, and backing up the student's work, are embedded in the User Options component. This is only available to the student when entering or exiting the Shell.

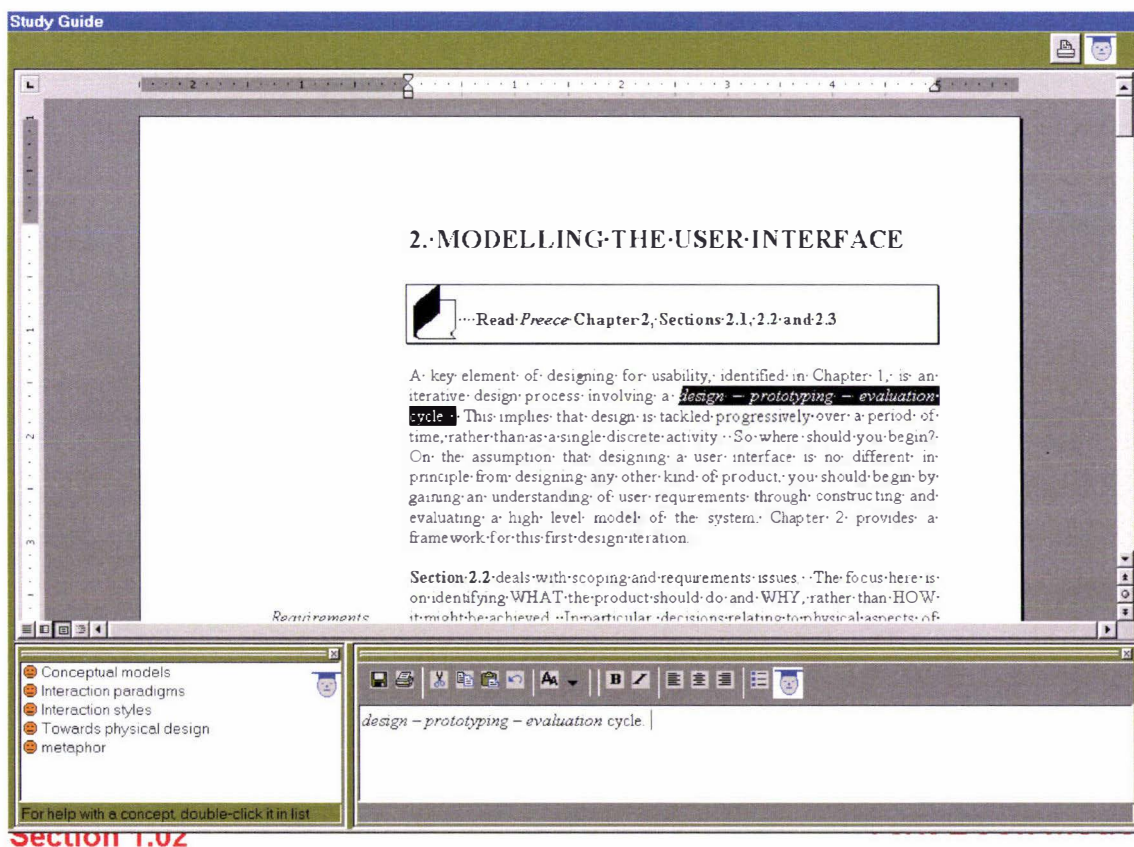


Figure 6.18: Separate learning components may be docked together.

Simplicity is also maintained by the Controller opening and closing individual learning components whenever the user changes topic or study mode. Users only open and close general-purpose learning tools, which they access through the Desktop Menu (Figure 6.20).

Help and documentation

A pop-up hint is attached to each interactive control such as a form button. This is displayed when the mouse cursor is on the widget.

All form templates include a Help component encapsulated in a tutor icon (button) (Figure 6.16). The Help screen is organised to first present quick hints for using the component, followed by more in-depth explanations.

The Help file is written in rich text format using a document template to enforce a consistent format and style. These files are stored in a Help folder under the name of its component. This provides a generic way for the system to match the help file to the calling component. If a correct help file has not been provided, then a default screen will be displayed when the user clicks the Help icon, informing them that no help is available yet.

The provision of additional hardcopy training documentation is addressed in Chapter 7.

6.5 Integrated learning support and communications

So far in this chapter, the aspects of IMMEDIATE's prototyping that have been discussed focus on the standalone features of the Learning Shell. The remainder of this chapter will address the Learning Shell's networked features to support collaborative learning and dynamic updating of learning resources, and the authoring application.

There were two main stages in prototyping IMMEDIATE's networked features:

- Developing the additional functionality for the Learning Shell; and
- Developing the communications and networking services needed to support this functionality.

The first stage is discussed here. The second stage is discussed in the Communication Management section (6.6).

The major prototyping task addressed in this section is to develop and evaluate a method for integrating the learning support, communications and collaborative work aspects of the Learning Shell, by means of a relational database. This was broken down into the following steps:

- Design and build the integrated database system.
- Develop the communication and collaboration components.

- Refine and evaluate for usability and modularity.
- Develop the learning support subsystem.
- Refine and evaluate for usability and modularity.

6.5.1 The system database

To meet the requirements for integrating communications and learning support a relational database is implemented at the system level of the Learning Shell. The integrated database model is shown in Figure 6.19. The database stores all communications to and from the student in the Messaging table. "Section" and "Topic" fields allow these communications to be organised by position in the course. The "GroupNo" field enables communications to be organised as collaborative group discussions. Keyword fields enable the tutor to link communications received by him/her to key concepts as well as topics in the course to facilitate monitoring student learning and updating the support database. Some tables, e.g. Concept, have only one attribute and function as look-up tables to ensure consistency in terminology, and facilitate linking, across the entire course.

The Connection table links individual concepts to broader themes to support more in-depth querying and exploration of learning support by the student. In the prototype, student queries on a concept may return additional information on that or related concepts stored in the What table, further reading (References), or related web urls (Links), depending on the type of query. The QueryType table maps natural language phrases selected by the student to the allowable query types.

The UserStats table stores system-level data tracking student interactions with the support system, which can be used to improve responses to student queries. This and other system information can be sent anonymously across the network using a special user type stored in the UserType table. The tutor receives a different view of the database, also determined by a different user type.

The database is replicated on each user's computer and is periodically updated by and updates a central copy, as described in Section 6.6. It has been implemented as a set of Paradox tables using Delphi's Database Desktop utility. The utility allows features like indexes and foreign keys to be defined, and SQL queries to be developed and tested separately from the application.

Learning components access the database through a Delphi data module, which belongs in the system layer of the Learning Shell architecture (Figure 6.13). The

module is a non-visual form which enables direct database access to be shifted from individual interface components to an intermediary layer for greater modularity and easier maintenance.

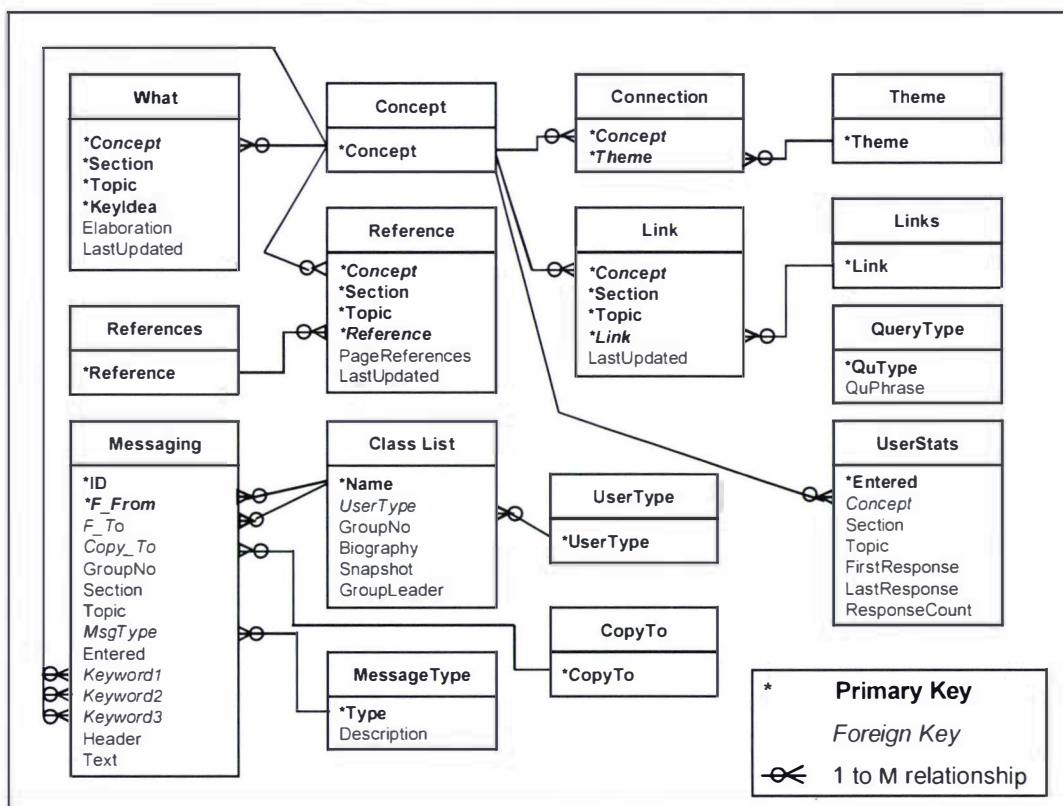


Figure 6.19: The integrated System Database model.

6.5.2 Communications and Group Work

Communications are handled in the Learning Shell through a Message List component (Figure 6.20) with a similar look and feel – and functionality – to an email client such as Microsoft's Outlook Express. The principal differences are:

- All messages are tied to a particular topic in the course, and can only be viewed when the Group Work study mode is selected for that topic.
- Incoming and outgoing messages can be shown together in sequence to facilitate collaborative discussion.

All new messages are added to the System Database. To provide for working offline, new messages may be sent immediately or later. When the student clicks the Update button (or sends a new message immediately) the Shell attempts to connect to the Course Repository via an FTP Client, providing appropriate messages in the status bar of the Message List. The FTP Client is implemented using a Third Party Delphi

component. If the connection succeeds, then all unsent messages are transmitted to the Course Repository, and all new messages for the user are downloaded from the Repository.

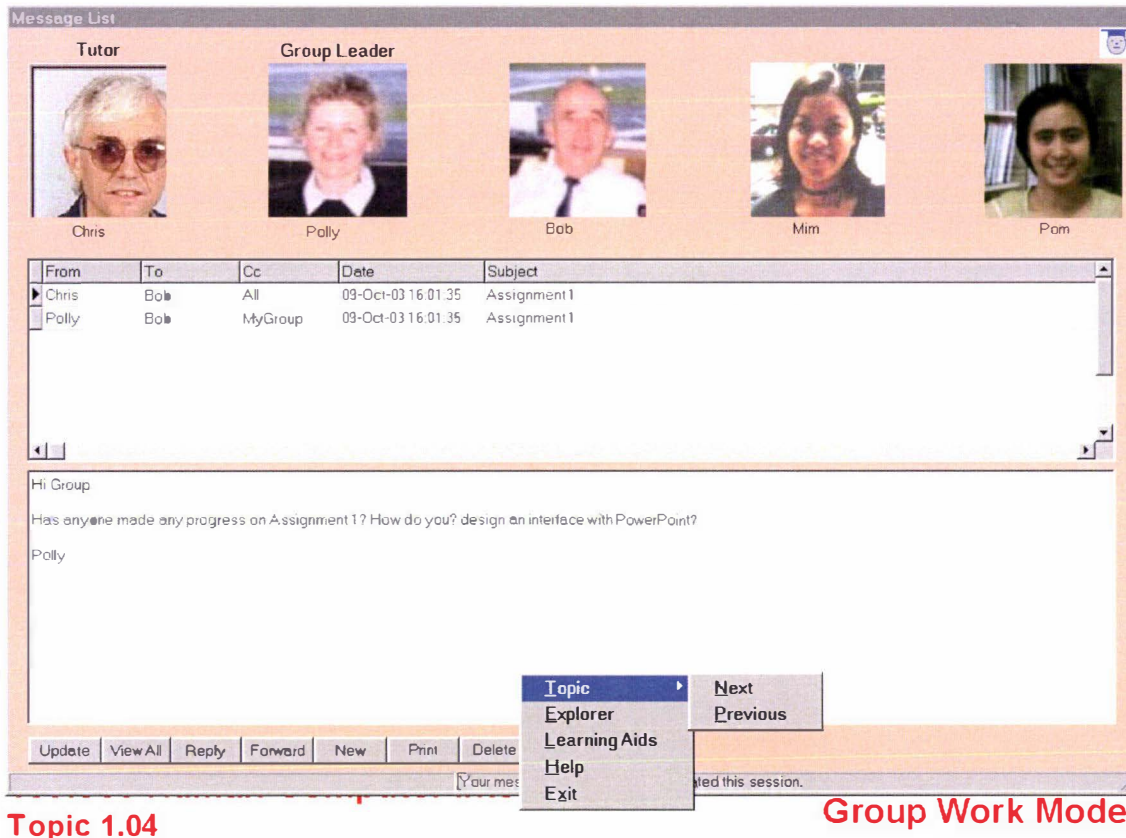


Figure 6.20: Group Work component with Desktop Menu open.

Refinements to Group Work component

A refinement added during the pilot study (Chapter 7) allows a list of all new messages received to be viewed on arrival, from anywhere in the course.

To facilitate collaboration, some additional features have been added. Photos of each member of a student's work group and their tutor are displayed. By clicking on a picture with the mouse, the user can access a short biography of that person or send them a message. Students may send a message to anyone involved in the course by selecting from a look-up list. Copies may be forwarded to the student's work group, to the tutor or to all students in the course.

A method was outlined for the storing and sharing of documents among group members to facilitate collaboration, but was not implemented in the prototype. In essence, this involved storing the file path as a field in the Messaging table and

providing a shared storage space in the Repository. To observe the narrow bandwidth constraint of the electronic mail service, shared files would be limited to text-based data rather than large multimedia executables like Word documents, which may run to several megabytes.

Integration of basic communication and collaborative facilities

The Message List component implements sufficient functionality to demonstrate the viability of the method for integrating basic communication and collaborative facilities into the Shell architecture, and for adding more sophisticated features where these will assist learning. Additional refinements to assist collaboration that could be usefully explored are an alternative format for displaying messages that better visualises the interactive dialogue within work groups, and directly incorporating a communication facility in the Assignment mode.

6.5.3 Extramural Support

Learning support has been implemented in the form of an Extramural Support component (Appendix F11). This component has been developed using the approach of starting simple, but providing for the addition of more sophisticated features should they prove necessary. Therefore, key aspects of the specification have been implemented at a basic level, but in a modular form that would easily accommodate new functionality. These include:

- *The subject of queries.* The point of entry to Extramural Support is selecting one of the key concepts or learning goals in a particular course module (topic). These have been defined by the course author, but can be updated by the tutor. Selecting a concept from the displayed list opens the Extramural Support window initialised to that concept. The student may then query the system on that concept. The database stores information on a greater range of subjects than are listed as key concepts or learning goals. These can be accessed within Extramural Support by searching related concepts as discussed under "Refinements".
- *Natural language-like querying and dialogue.* This is currently limited to the user choosing one of three initial questions on a selected topic ("Explain...", "Who wrote more on ...", "Where do I find more on...") and receiving a response. The student can then continue a dialogue with the Support system via a set of buttons, which are enabled or disabled according to the state of the dialogue (Appendix D4 and D5). Figure 6.21 shows the top level methods for supporting this dialogue. Provision is made for accommodating a wider range of query types. This would

require adding a more sophisticated parsing mechanism to enable translation into more detailed specification of SQL queries.

- A *directed search algorithm* to return the best available result to the user. In the base implementation, the ranking of the results is determined solely on proximity in the database to the current topic. Additional ranking criteria could be easily added. For instance, providing for a ranking to be added to each database entry (e.g. Level 1, 2 or 3) would enable multi-level support to be offered. Results could then be based on a breadth-first or depth-first search of the database, depending on the preference of the learner.

```

procedure TExtramuralSupport.HelpUser;
begin
    prepareHelp;
    parseUserInput;
    executeAsSQL;
    presentBestResultToUser;
    waitForUserResponse;
end;

procedure TExtramuralSupport.respondToUser;
begin
    actionUserResponse;
end;

procedure TExtramuralSupport.TryHelpingUserAgain;
begin
    presentNextBestResult;
    waitForUserResponse;
end;

```

Figure 6.21: Top level methods for interactive dialogue with user.

- *Dynamic updating* which improves the response to users' questions over time. The course tutor may update the items in the support database at any time. The tutor is automatically informed whenever a student rejects the system's response to a query (i.e. clicks the NO button). In addition, the student can redirect the question to the tutor with his or her own comments. The tutor may respond personally to the tutor, to all students, update the database, or all of the above. Provision has also been made to track students' dialogues with the system and store them in a UserStats table. These can be periodically transmitted to the tutor to help in analysing and updating the database entries.

Concept Map

An early refinement was to add the Concept Map (Appendix F12). This component uses the technique of concept mapping (Novak et al., 1984) to assist learning through visualisation. It provides a user-centred avenue for exploring the Support database

more deeply in a guided manner. When the student chooses to view a Concept Map for the current concept, all related concepts in the database, and the relationships between them are graphically displayed (Figure 6.22).

Any related concept may be selected as the basis for a new query. Concepts and their relationships are entered into the database by the tutor. The map itself is implemented as a graph ADT in which the nodes are concepts and the arcs are relationships. Drawing the map is then a matter of processing the graph.

Extramural Support was re-implemented to be accessible from any component through the Controller, rather than as an independent learning component in its own right. The student accesses Extramural Support via a Key Ideas component, which lists the key concepts to be learnt in the current topic (Figure 6.17). A future refinement is to make context-sensitive learning support accessible from every learning component by opening Key Ideas via an icon on all forms, in the same way that the Help feature currently is offered.

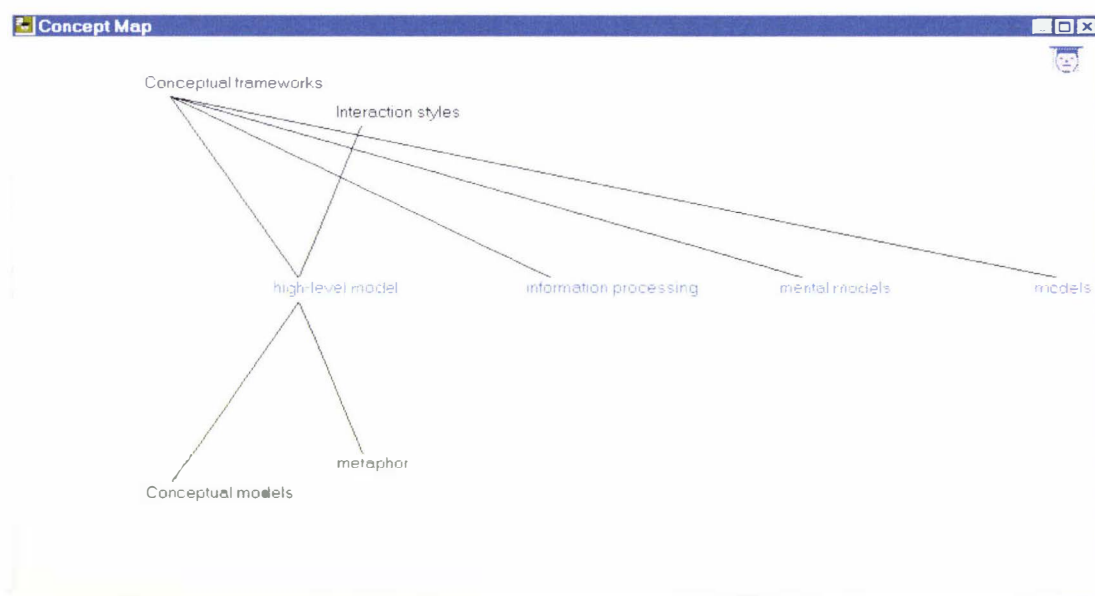


Figure 6.22: Concept Map for “conceptual frameworks”.

Self-Assessment

In the Learning Shell, Self-Assessment is a separate learning component approximating to a revision quiz in a textbook. It assesses the learner’s understanding of the key learning goals in a particular topic. This information is passed to the Controller which stores it in the System Tree. It is then available to be used by the Course Explorer to visually display the student’s progress through the course as described earlier (6.3.1).

A further refinement takes advantage of the ability of the System Tree to store additional levels of information (Figure 6.7). The self-assessment information is used to display to the student their progress in understanding individual concepts and learning goals. An icon is displayed to the left of each concept in the Key Ideas component (Figure 6.17). The traffic light metaphor is again used to colour-code this icon to show the student's level of confidence in that idea – GREEN (fully understood), AMBER (partially understood but requires further revision), or RED (not understood at all).

Self-assessment may be by any method that can be implemented on a computer. The prototype currently provides for assessment via a multiple-choice quiz – if the author has provided one – or by default, a questionnaire through which the student expresses their confidence in each concept. Concepts are presented to the student in random order. Once the student has completed the self-assessment, the results are saved via the Controller. They are then used by the Key Ideas component to update each concept icon, as well as by the Course Explorer to update topic and section icons in the course Table of Contents. (This is illustrated in Figure 7.12.)

Self-assessment can also serve as a revision tool. Initialising the Learning Shell, for the purposes of revision, is simply a question of setting values for all topics and concepts in the model to the default. This sets their icons in the Key Ideas and Course Explorer components to RED.

6.5.4 Coarse-grained learning help

As implemented, Extramural Support is a coarse-grained form of learning help structured by topic and limited to a few basic query types. In the first instance, it organises material that might appear in a textbook's glossary, chapter summaries and bibliography, in a way that it can be accessed by topic. This base is then built upon by the tutor adding in and updating material as required, or as it becomes available.

Integration with the communications system offers a number of advantages by combining both individualised and collaborative forms of learning support, and by providing a mechanism for dynamic updating of the content. Provision has been made, in its modular design and implementation, for the incorporation of more sophisticated querying and search mechanisms on an as-needed basis.

6.5.5 Individualisation

The learning computer was conceptualised as supporting individualisation along four dimensions: sequencing, presentation, learning level, and learning support (4.3.1). This section discusses how these have been addressed in the prototype.

Sequencing

Sequencing is concerned with the order in which parts of the course are presented to the student for study? In the learning computer prototype this feature is adaptable, i.e. the student can select any topic in the course at any time from within the Course Explorer component. Colour-coding associated with each section, topic and key concept, as described under System Tree (6.3.1) and Self-Assessment (above), guides the student in their choice. This same mechanism would allow more prescriptive sequencing (e.g. hiding some sections and topics until prerequisites have been successfully completed) although this would conflict with the learning computer's student-centred exploratory emphasis.

Presentation

Learning style, or how the content of a particular topic is presented to the student, is selected by the user through choosing from the different study modes supported by the course author. These are displayed in the Course Explorer for the currently-selected topic. The modular character of the Learning Shell means that a new mode can easily be defined, and learning components can be added or modified to support any desired, computer-runnable functionality for that mode. The new mode will then be displayed for any topic in which the course author has provided the necessary files. These files may be learning content, they may provide links to learning content stored on a CD or a networked computer, or they may launch an independent application installed on the student's machine.

Learning level

The learning computer concept does not require that all content be adapted to the student's current level of understanding. However, the Student Model makes provision for storing the individual user's learning level, which would be determined through a similar self-assessment process as that used to assess understanding of individual concepts and topics. This would enable some individual learning components or study modes to accommodate different learning levels, using adaptive or adaptable strategies. Both these approaches have been well-evaluated (Brusilovsky et al., 1997;

Murray et al., 2000) and were not implemented in this prototype. Some components in the prototype use their own internal mechanism to provide adaptive or adaptable content, e.g. a Tutorial component that links to a separate ITS application.

Learning support

The most important dimension of individualisation to be prototyped is learning support (Section 6.5). An adaptable mechanism is provided through which students can initiate and define queries from anywhere in the course. This is individualised through user-driven interactive dialogue and exploration. Important information is stored in the Student Model and other system layer modules about the student's understanding of individual concepts and topics and his/her previous interactions with Extramural Support. This provides the basis for implementing more sophisticated search algorithms to better tailor system responses to the individual student, as the need for this is demonstrated.

6.6 Communications management

This section discusses the issues addressed in developing the communications and networking services needed to support the Learning Shell.

The requirements for an e-learning system (Table 4.2) demand that students be able to exchange messages within the course, update their course materials as needed, and receive up-to-date learning support. Tutors, on the other hand, must be able to communicate with students and update learning materials and the support database as required. All these tasks require communication over the network. Some IMMEDIATE features, such as Extramural Support, also require behind-the-scenes communication across the network at the system level.

In the network specification for IMMEDIATE (Figure 5.9), each Learning Shell connects to the Course Repository via a Communications Manager, using custom protocols on top of FTP. In the implemented version the Communications Manager has been broken down further into two components – an FTP Server (Appendix H1) and a Repository Manager (Appendix H2) – which have been installed together with the Repository on a single computer.

The implementation assumes that the Repository would be managed by an experienced systems administrator, who would set up new users, complete some tasks semi-manually, and optimise settings as required. Repository management services operate at two levels – the FTP Server and the Repository Manager.

The FTP server acts as the point of entry into the IMMEDIATE network. It logs the user onto the network, assigns them to their correct folders, and allots them the permissions to complete the necessary file transfers. To support this, two separate user profiles have been created in the server – a tutor and a student. All tutors are assigned a single profile and share a “Teachers” folder system. Each student shares the same profile, but is also allotted their own root directory (Figure 6.23).

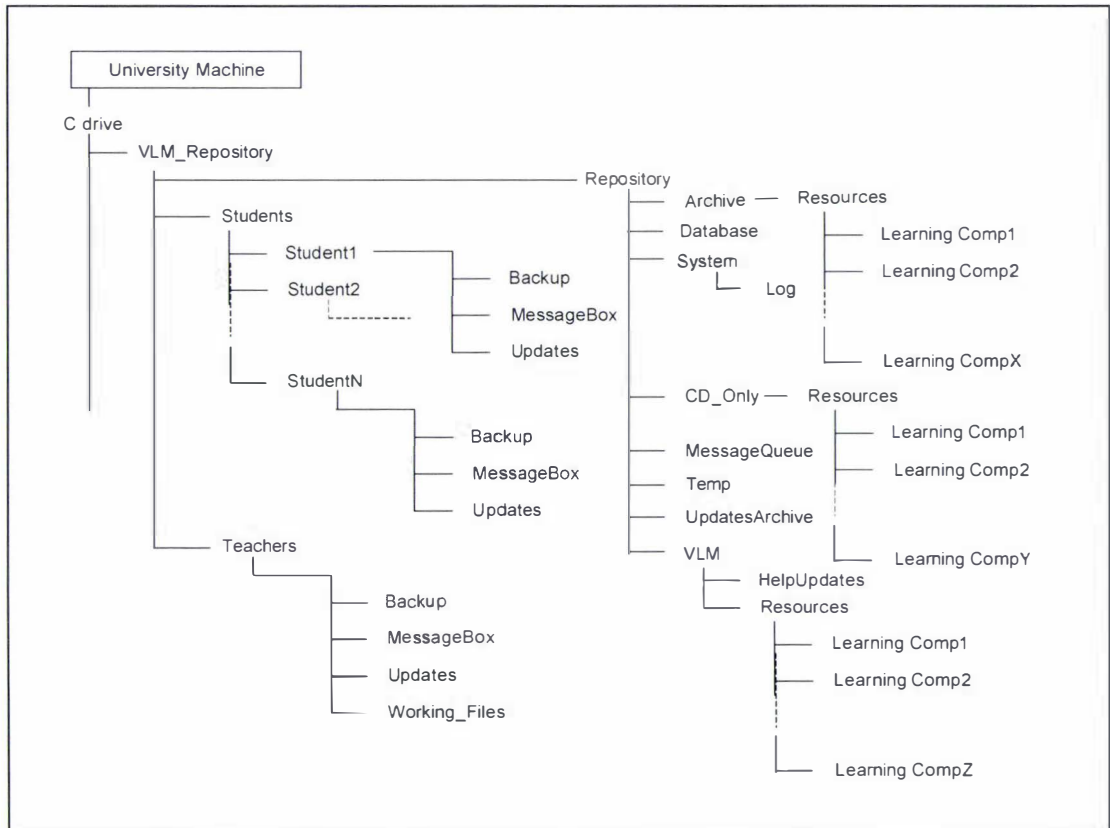


Figure 6.23: Repository folder structure.

The FTP server used is War_FTP, developed by Jarle Aase and freely downloadable from his web site.¹

For the IMMEDIATE network to be accessible, both the FTP Server and Repository Manager must be online at all times. Once launched, they remain running in the background in a minimised state, appearing as icons in the Windows task bar. The system administrator double-clicks on these icons to bring the applications back into view.

¹ <http://www.jgaa.com>.

6.6.1 Repository Manager

The Repository Manager implements those aspects of communications and data transfers unique to IMMEDIATE. It enables data on users' machines to be synchronised and updated across the network, while allowing for simultaneous access by multiple users, and for the possibility that the connection with any or all users may be broken at any time during the process. To meet the requirements of the Learning Shell three kinds of data transfers are managed: asynchronous electronic messaging between course participants, periodic updating of the Extramural Support system on each student's computer, and the uploading of updated learning resources. This has required:

- an appropriate Repository structure;
- a set of communication protocols, implemented over the TCP/IP protocol family which ensure accurate synchronisation between sender and receiver;
- a message-queuing mechanism, which ensures that the earliest message received is the first processed;
- a parsing mechanism for reading/writing messages from/to a database; and
- a separate set of protocols for updating learning resources on each student's machine.

A detailed, daily log is provided to enable the administrator to monitor network traffic and troubleshoot problems, and optimise settings. The main screen of the Repository Manager can be seen in Figure 6.24. Its underlying architecture is shown in Figure 6.25. Selected source code is attached as Appendix H3.

Repository

The Repository is a set of folders with the structure outlined in Figure 6.23. The folders contain an archive of the entire course content including a master copy of the Learning Shell System Database; all communications amongst those involved in the course; updated course materials for the students to upload; and individual folder systems for each student, including a mailbox for new messages and space for the Learning Shell to store backups of the student's work if desired.

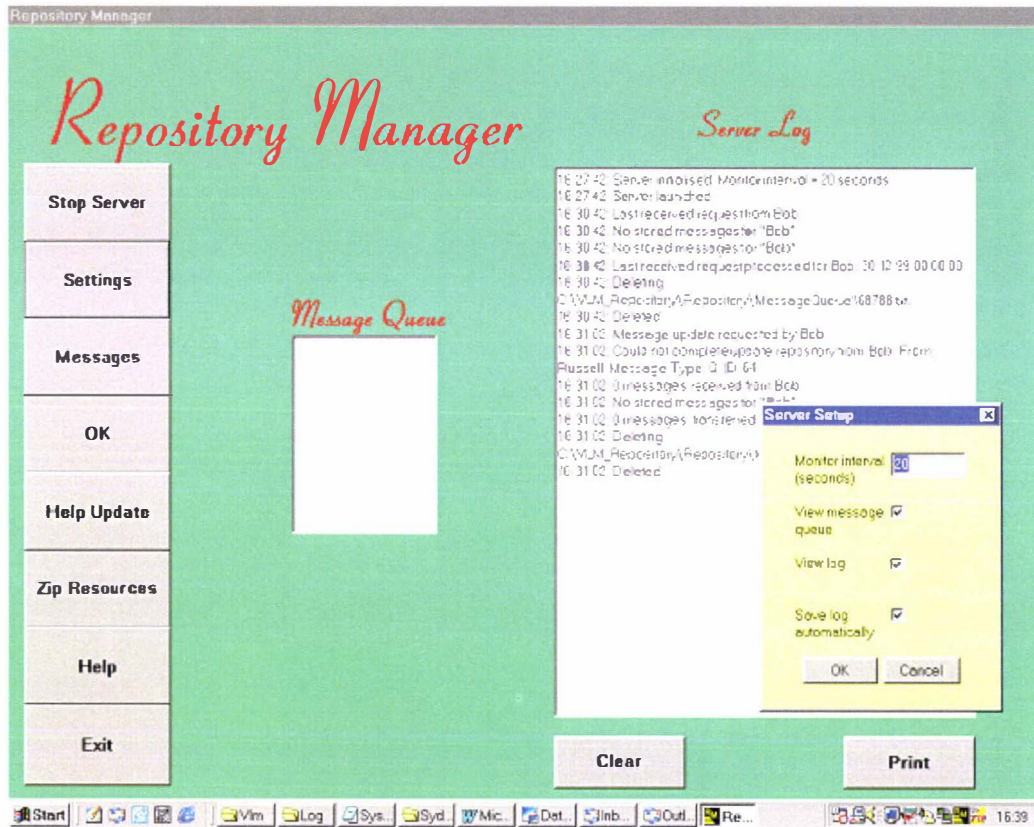


Figure 6.24: The Repository Manager.

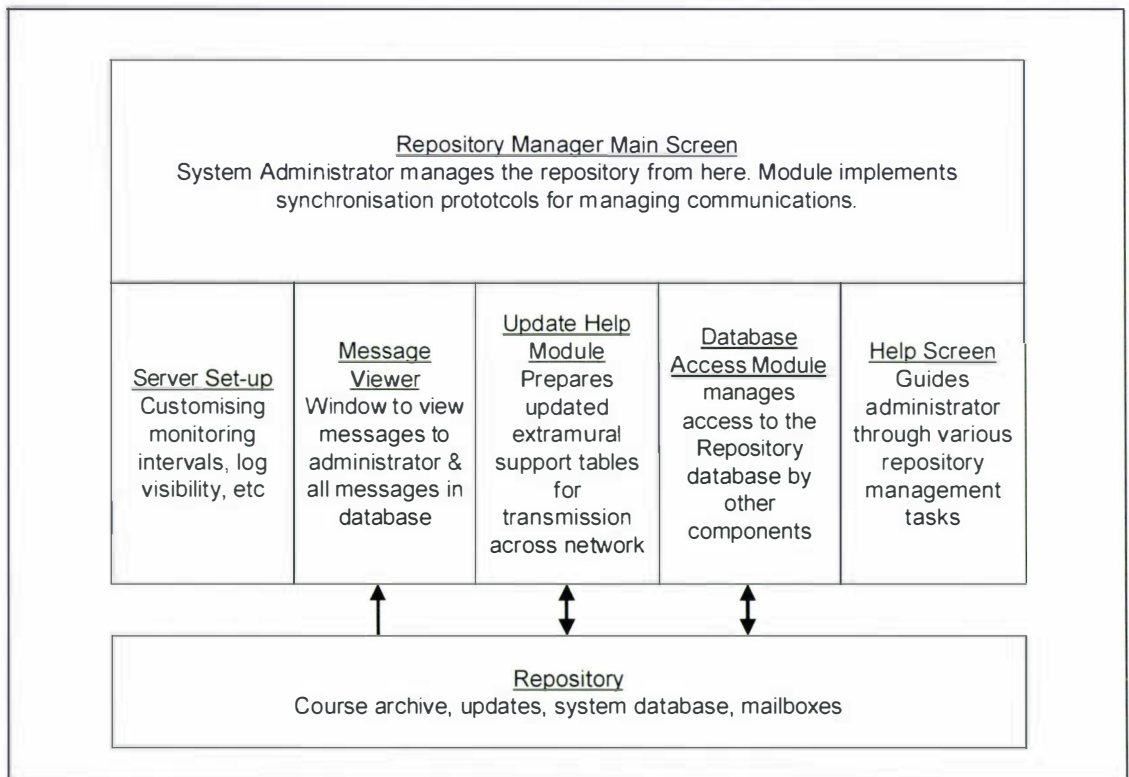


Figure 6.25: Repository Manager architecture.

6.6.2 Asynchronous messaging

To meet IMMEDIATE's specification, asynchronous electronic messaging has been implemented using a message-queuing technique that automatically handles communications from multiple users on a first-in, first-out basis. These messages may be between course participants, between the tutor and the system administrator, between system components and the tutor or administrator, or between constituent applications at the system-to-system level.

Only the administrator can view all mail held in the database.

Messaging protocols

A set of communication protocols had to be designed and implemented at both the client and server end, to ensure the integrity of communications over an unreliable network, to which multiple users may be connected concurrently. These protocols operate on top of the File Transfer Protocol. Messages are aggregated into a single file for transmission, which is parsed into individual messages by the receiving application. The message transfer process entails a two-way exchange of messages between the user and the Repository.

Because messages are stored in relational tables they must preserve the constraints on a replicated database distributed over multiple computers:

- Each message is unique within a single copy of the database so as to avoid redundancy or a primary key conflict.
- Duplication of messages between different copies of the database must be possible.
- Copies of all messages connected with a particular user must be stored on that user's computer. But all messages do not need to be stored on all computers.

At a minimum the protocols had to ensure that:

- any message lost through a network interruption will be re-transmitted the next time that user connects;
- transmissions from multiple users can be interwoven without difficulty; and
- no message will be transferred twice.

A set of protocols has been designed to meet these requirements (Figure 6.26).

These protocols, together with the messaging-queuing and file-parsing mechanisms

discussed next, proved sufficient to meet these requirements.

Central to the protocols is a two-step synchronisation process to ensure that all the messages that need to be sent, and only those that need to be sent, are transmitted from each end of the connection, while allowing for overlapping communications from different users (Figure 6.27).

Message-queuing

Message-queuing applications communicate by the client passing a message to an intermediate structure in which incoming requests are queued until the server is ready and able to deal with them. Any response from the server is returned via a similar mechanism. Through the prototyping process, the following mechanisms were determined for implementing message-queuing in IMMEDIATE.

Client-to-server communications are handled through a Repository folder named 'MessageQueue' to which all users are assigned access. All incoming communications are placed in this folder, which is monitored by the Repository Manager at a regular interval. The monitoring interval is determined by a timer component (thread) to which a *checkMsgQueue* method is attached. The administrator can adjust this interval.

Server-to-client communications are handled through an individual 'MessageBox' folder located in each user's Repository root directory.

The MessageQueue folder is organised as a priority queue of messages in the following manner:

1. At the end of each monitoring interval, the *checkMsgQueue* method is called. The files in the MessageQueue folder are sorted by name. These file names will look like '67888.txt' or '67864.txt'.
2. The first item, which will be the one with the earliest date_time stamp (smallest integer), is processed.
3. Once processed, the response is named either 'answer.txt' or 'messages.txt' in line with the messaging protocols, and placed in the user's MessageBox. This naming convention ensures that any older versions of these responses inadvertently left behind will be overwritten by the new ones.

Precondition:

The user has successfully established a connection via the FTP server.

Message Format:

- All transmissions are in the form of text files, in which the first five lines constitute a header which must follow a set format as below.

<TO RCVR>	Who the transmission is addressed to. Must be the actual receiver of the transmission, and be the System, the Tutor or a name on the class list
<FROM SNDR>	Who has sent the transmission. Must be the System, the Tutor or a name on the class list.
<MESSAGE TYPE>	May be a last received request/answer, a list of messages or a test transmission
<GROUP NUMBER>	The Work Group to which the user belongs. -1 means the tutor. The default group is 0.
<DATETIME STAMP OF LAST TRANSMISSION SNDR RECEIVED FROM RCVR>	Default = 0.0, which means send all messages for the sender in the repository database

- Each message within the body of a communication is separated by a new line containing only the string '[END]'.
- A transmission from a client application is named with the string form of an integer representing the date and time that the transmission was composed and sent, as in '<date_time_as_no>.txt'.
- A transmission from the Repository Manager will be named either 'answer.txt' or 'messages.txt' in line with the steps in the handshake protocol

Message Transfer

Messages must be updated via a two-step synchronisation and transfer process:

- The client application sends a transmission to the Repository requesting the date-time stamp of the last message successfully transferred **from** the client **to** the Repository.
- The Repository sends a transmission to the client containing the date-time stamp of the last message successfully transferred **from** the client **to** the Repository. This message must be named 'answer.txt'.
- The client sends a transmission containing the date-time stamp of the last message successfully transferred **to** the client **from** the Repository, and all the outgoing messages with a date-time stamp later than the date-time received from the Repository.
- The Repository sends the client a transmission containing all the incoming messages for that client having a date-time stamp later than the date-time received from the client. This message will be named 'messages.txt'.

Figure 6.26: Messaging protocols

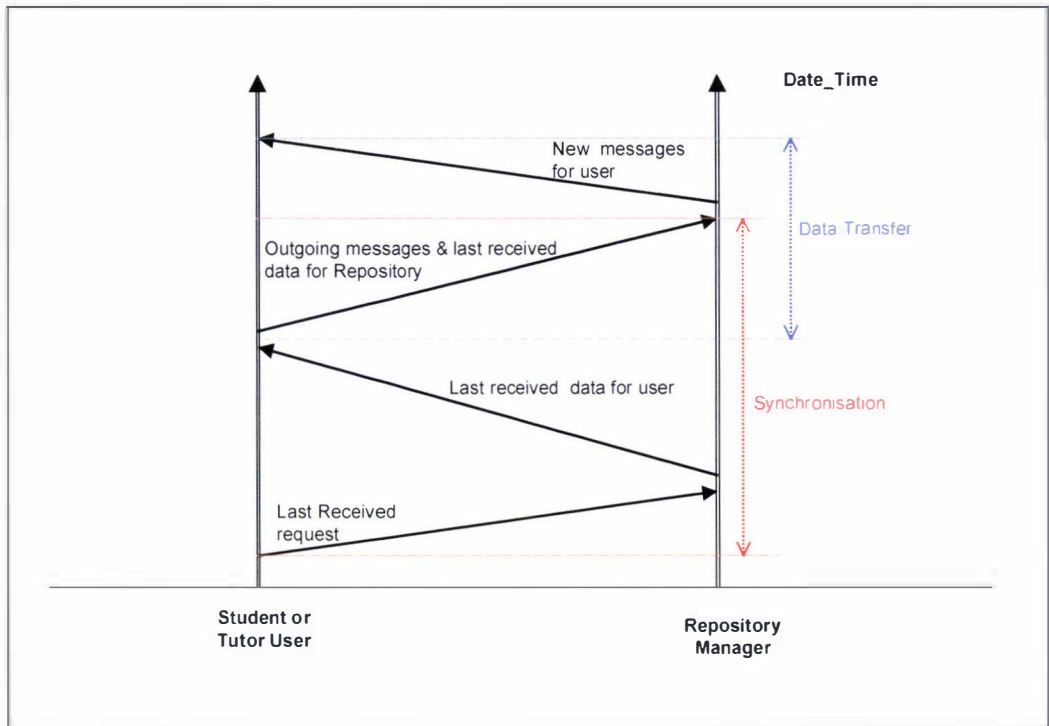


Figure 6.27 (a): Two-step transmission protocol.

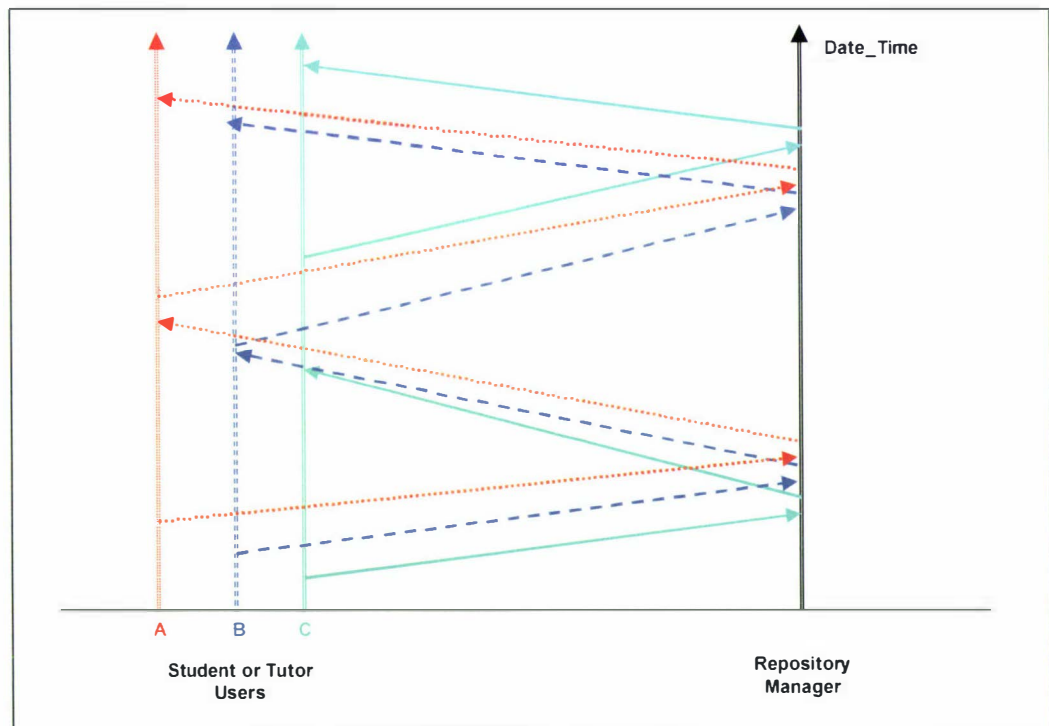


Figure 6.27 (b): Transmission protocols accommodate overlapping users.

From the user end:

1. The Learning Shell checks its Message Box in the Repository at regular intervals for the required file.
2. When the file is found it is moved to the user's machine and deleted from the server.
3. If no file arrives within a specified time period, or the connection is broken, then the transaction is aborted by the Learning Shell and the user is notified and requested to try again later.

File-Parsing

All transmissions within the IMMEDIATE system are text files in which the first five lines constitute a header and the remainder the body, as defined in Figure 6.26. These files must be composed at one end of the transmission and parsed at the other. The code for achieving this at the Repository Manager end of transmissions is included in Appendix H3.

When the Repository Manager finds a file in the Message Queue end, it passes it to the *processMsgList* procedure. The processing steps taken by this procedure are:

1. Process the transmission header
2. If transmission is a message list, then
 - a. Update the Repository with the messages
 - b. Return a list of messages to sender based on Last_Received information provided
3. Else if transmission is a Last_Received request, then
 - a. Return Last_Received information to sender

Processing the header is simply a matter of parsing the first five lines of the file for the necessary information.

Messages are stored in a relational database on each computer within the IMMEDIATE network. They are not transmitted as individual text files but as a list of messages each separated by an '[END]' symbol. This list includes all the outgoing messages in the sender's database that have not already been successfully added to the recipient's database.

All messages include a date_time stamp, recording when they were composed (i.e. the "Entered" field in the Messaging table in Figure 6.19). A Last_Received request from a client, viz. a Learning Shell or authoring application, is fulfilled by the Repository finding

the incoming message from that client with the most recent date_time stamp, using an SQL query like:

```
select MAX( entered ) from Messaging where F_From = : F_From;
```

where the ':From' variable takes the value of the client user name. And the user responds with a Last_Received message to the Repository Manager based on the result of:

```
select MAX( entered ) from Messaging where F_From <> : F_From;
```

where ':F_From' again takes the value of the client user name.

To compose a message list, the sender uses the Last_Received data to run an SQL query on the local copy of the Messaging table (Figure 6.28). The result set is then written to a text file, with each field written to a new line and '[END]' separating the messages. Updating the Messaging table at the receiver end is just a matter of reversing this process.

(a) Query run at client end:

```
select *
from Messaging
where ( F_From =:F_From or MsgType = "Q")
      and entered> :entered;
```

(b) Query run at Repository end:

```
select *
from Messaging
where entered> :entered and F_From <> :F_From
and( F_To =:F_To or CopyTo = "All"
     or ( CopyTo = "MyGroup" and GroupNo =:groupNo ) );
```

Figure 6.28: SQL queries for composing message lists to be transmitted.

However, the main body ("Text" field in the Messaging table in Figure 6.19) of a message may be many (or zero) lines long. Thus, the primary role of the '[END]' symbol is to mark the end of the Text field.

An advantage of transmitting messages as straight text files is that they are platform-independent. It is not necessary that the Repository and Learning Shell databases have the same physical implementation. It is only necessary that they have the same logical structure.

The key methods for implementing the messaging system at the Repository end are shown in Appendix H3. Implementation at the Learning Shell end is similar. The interface for the Shell messaging module is included as Appendix H6.

6.6.3 Procedures for Resource Updating over the network

The updating of resources over the network is a semi-automatic process involving the tutor, the system administrator, and the student. The primary challenge has been to ensure the integrity of the update process – while minimising user intervention – by ensuring that *all* students receive *all* updates, and that more recent updates are not overwritten by older ones.

Database updates

The updating of the Extramural Support system, which involves updating a database, has been handled differently from the updating of other Learning Resources, which involve adding or replacing files. This was to simplify the transfer process at the Learning Shell and to minimise any demands placed upon the student.

The Repository copy of the Support database is directly updated by the course tutor. The administrator uses a Repository Manager facility that converts each updated table into a text file and transfers it to a HelpUpdates folder in the Repository, in preparation for downloading to student machines with other course updates.

Data file updates

The course authoring application automatically transfers updated resource files to a special location in the Repository. The updated resource files and updated Support tables are then zipped (compressed) into an appropriately named executable resources file and placed in the UpdatesArchive folder. These zipped files are named consecutively in the format 'Resources_<NextAvailableNo>.exe'. In the prototyped version the zipping process is performed by the administrator.

Learning Shell

The operations outlined above greatly simplify the updating procedure from the student user's perspective.

On logging on to the Learning Shell the user may choose to update learning resources from the Internet or from a disk (Figure 6.29). Whenever the student opts to update from the Internet the Learning Shell automatically connects to the Repository, checks for new updates, and then automatically downloads and installs any updates not previously installed.

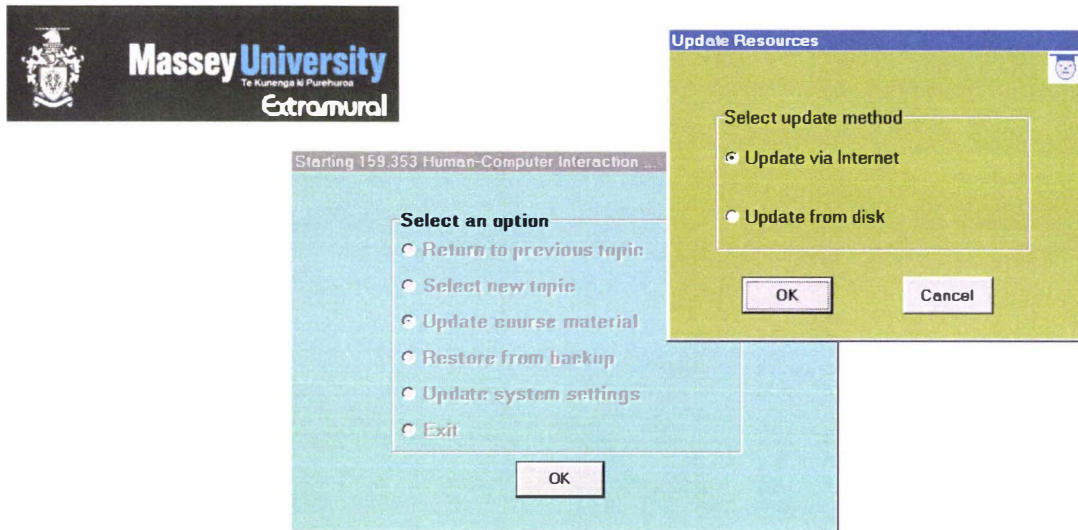


Figure 6.29: Learning Shell update resources screen.

This has involved implementing the following algorithm in the Learning Shell (Appendix H4):

1. Connect to FTP server.
2. Change directory to the Repository Updates_Archive folder.
3. Get file number of last resource update stored in Student Model.
4. Download all update files with more recent file numbers to an Updates folder.
5. Execute file with lowest update number.
6. Wait for the user to unzip the file by default to a temp folder.
7. Move the updated resources to the Learning Shell and delete the temp folder.
8. Repeat steps 5 to 7 until there are no more update files to unzip.
9. Update the student model with the most recent file number successfully unzipped.

10. Return control to the user.

As part of this process, updated Extramural Support files are placed in the HelpUpdates folder (Figure 6.9). At the end of the start-up procedures and before the course is opened, the Controller's UpdateExtramuralSupportFiles routine is automatically called. If there are any files in the HelpUpdates folder, then they are used to update the Extramural Support database and then deleted (Appendix H5).

The most urgent updates – e.g. assignment results or query responses – involve small text files suitable for transmission over a narrowband Internet connection. Provision has been made for larger updates – e.g. multimedia files or executables – to be stored separately in the CD_Only folder to be distributed by one-way broadband (e.g. CD, satellite download or fast internet).

The key to maintaining the integrity of the update process was to ensure that update files were downloaded consecutively from oldest to newest, and that the separate file zipping application had successfully completed its task and the updates had been installed, before the Shell updated the Student Model. The file naming system addressed the first issue, while the second was solved by attaching the updating code to a Delphi timer component (Appendix H).

Updating from a disk is a simplified version of this procedure without the unzipping step, and only requires loading the disk in the computer and clicking an on-screen button with the mouse.

6.6.4 A challenging task

Implementing the Communications Manager proved to be one of the more challenging tasks in the overall implementation of IMMEDIATE. This reflected both the inherent complexity in getting multiple separate applications to successfully communicate and cooperate over a network, and the need to simplify and minimise the associated tasks for the student user.

The goal of minimising the load on the student has been successful. In the implemented version, the only requirement demanded of the student user is to confirm the unzipping processing during the transfer of learning resources, by clicking on an 'OK' button to accept all the default parameters. If, however, the student does something else then an error may occur and the transfer will not succeed and will have to be repeated.

The updating process at the student end should be completely automated by adding a

file compression/decompression utility to the Learning Shell similar to the ones developed for automatic file and folder manipulation. This would also simplify the zipping process at the Repository end. Such utilities exist but were not available for this project.

The techniques developed for updating course resources could also be used to update the Learning Shell executable itself. This makes maintenance straightforward. Improvements to Shell modules could be incorporated, and the updated software distributed, with course content.

Further information on the communications management implementation is included in Appendix H.

6.7 *Authoring application*

This section summarises the experience in developing IMMEDIATE's course authoring and management application. The primary prototyping goal was to demonstrate a mechanism by which a teacher can author and update a course for the Learning Shell.

Given time constraints and IMMEDIATE's focus on proving the learning computer concept at the student interface, the design of the authoring application was very much the RAD process itself. Wherever possible, code developed for the Learning Shell was re-used in this prototype. Nevertheless, it still required a significant programming effort, with the functional authoring prototype running to nearly 7000 lines of source code. Its main components are depicted in Appendix I.

The prototyping steps followed were:

- Define course authoring for IMMEDIATE.
- Propose an authoring mechanism.
- Implement sufficient functionality to demonstrate the mechanism.
- Evaluate the results.

Each of these steps is now discussed in turn.

6.7.1 Course Authoring and Re-use

For the learning computer approach to be feasible, there had to be a practical way for a teacher to author and update course materials, and maintain the learning support system, without the teacher being required to understand how the learning computer works. To solve this problem it was useful to consider how a teacher prepares an

extramural course.

For IMMEDIATE, extramural distance teaching is viewed as an extension of internal university teaching. The teacher, drawing upon his/her classroom experience, prepares a study guide for the extramural student. This study guide is fleshed out with additional materials from the internal course – lecture notes and handouts, supplementary reading materials, etc – together with materials specifically directed to external students.

IMMEDIATE's task was to transfer that authoring process to a computer-based extramural environment. In support of multidimensional learning, it should provide for adding further learning materials beyond what can be offered in a paper-based course.

From this perspective, IMMEDIATE does not offer templates for improving the educational effectiveness of individual learning materials. It offers a mechanism for quick assembly and effective organisation of pre-authored material into a complete extramural course.

6.7.2 Authoring mechanism

The proposed mechanism was a course authoring and management application built around the same core data structures and algorithms as the student application, especially the System Tree. This provided a framework for the teacher to define a course structure and its study modes from the available components, and to add or edit learning materials in the appropriate format and location, using a "drag and drop" graphical interface.

A greater level of complexity was permitted for the authoring application than for the Learning Shell. It could be assumed that while course authors are not programmers, they are more experienced computer users than extramural students, and are backed up by university computing support services.

The authoring application is built upon the complete separation of functionality and data in the Learning Shell. Students would be supplied with a compiled version of the Learning Shell with pre-installed learning components, and a folder system containing the reference models, the learning support and messaging database, and all learning materials for the particular course.

A central copy of this folder system is stored on the university repository server (Figure 6.23). The authoring files are stored on the teacher's computer. At the end of each authoring session, updated files are automatically transferred from the authoring

application on the teacher's computer to the appropriate Shell directory in the Repository. From the Repository, the learning materials are transferred to the students' computers by CD or the Internet.

This authoring process was broken down into three main stages:

- define the overall course structure (table of contents).
- define the study modes that will be available to the student.
- add the course learning materials.

6.7.3 Implementation

The implementation of the principal components of the authoring application is now described. Sample screen shots are attached as Appendix I1 & I2.

Table of contents

The authoring application provides a direct manipulation interface in which the teacher creates a Table of Contents (TOC). This is an instance of the Learning Shell's System Tree data type which models the basic structure of the extramural course. The application then saves the TOC to file in the appropriate format to be read by the Learning Shell, using the System Tree's operations.

Study modes

At the start of the authoring process, an interface is presented containing a System Model initialised to the core study modes, which must be included in any course, and their constituent learning components. These approximate to the basic elements of a correspondence-based extramural paper. They are also necessary for the Learning Shell to function correctly.

The teacher may then add components to these core modes or create new study modes, by selecting from a list of all the pre-installed learning components available in the compiled Learning Shell.

Learning resources

Once the TOC and Study Modes have been defined, learning materials may be added or updated using an Add Learning Material form (Figure 6.30, Appendix I2).

IMMEDIATE provides a simple drag and drop mechanism for correctly adding (re-using) pre-authored learning materials such as lectures or tutorial programs without

requiring any special programming skills or understanding of how the system works.

Tools and templates have been provided for authoring learning modules specific to the Learning Shell, such as self-assessment tests and definitions of key concepts and learning goals for each topic. These templates provide the vehicle for adding such modules to the course.

The Add Learning Material screen displays the TOC, the learning components available to the course, and the teachers folder system where the pre-authored materials are stored. Selecting a topic in the TOC with the mouse causes a list of all the learning resources that have been added to that topic to be displayed in a separate box below. The nodes in the TOC are marked with coloured crosses and ticks to indicate which sections and topics have had the minimum resources added for proper functionality. A course is not ready for distribution until the root course node is marked with a green tick.

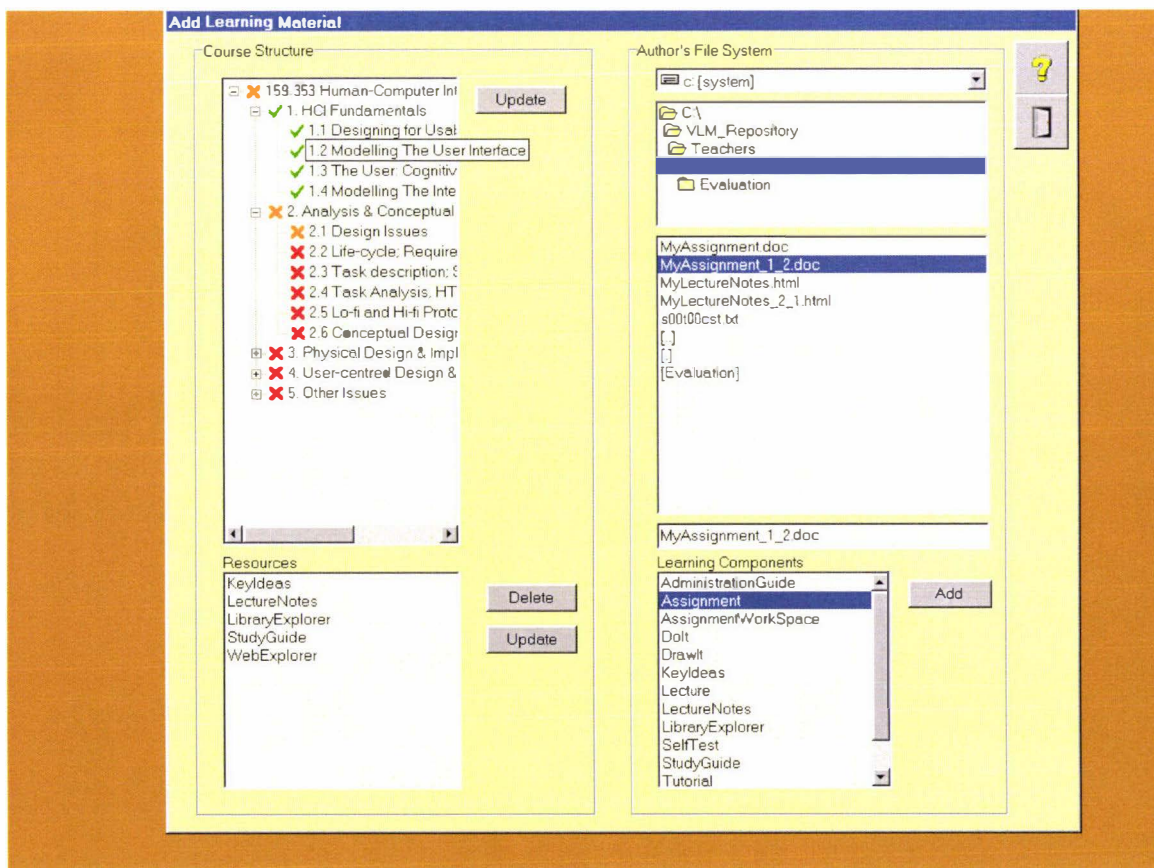


Figure 6.30: Interface for adding and updating learning materials.

To add a resource the author selects the topic and component for which the resource will be added, and then the appropriate data item in their directory system. This data item may be a file, or a folder with sub-folders. If a folder is selected, then the user is

asked to identify the index or start up file within it.

The application stores information about each component. If special processing is required, such as adding multiple files or authoring a resource, the user is guided through that process. The system uses the Learning Shell Resources Model module to code the resources so they can be identified by the Shell and transfers them to the appropriate position in the Repository archive at the end of the authoring session.

Authoring and Updating the Support System

Several facilities have been provided to enable tutors to author and update the Extramural Support system: When a course is authored, the authoring tool requires that a Key Ideas resource be provided for every topic in the course. A facility is provided for entering each key concept or learning goal in a topic, and its elaboration, into the course database. From this facility, tools may be accessed for linking concepts to more general themes (i.e. relationships) which will be displayed in the Concept Map, and for adding book and web references.

While only the key learning goals are added to the Key Ideas component during the authoring stage, additional concepts can be added to the database or edited at any time via the Group Work interface, through which the tutor monitors and responds to course discussion and queries. From here the tutor may classify discussion items by up to three keyword phrases for easy reference. They may also be edited and added to the support database.

6.7.4 Evaluation of authoring tool

To fulfil the key requirements for a computer-based distance learning environment, IMMEDIATE must support authoring and re-use.

The component-based architecture of the Learning Shell supports code re-use by a programmer. But it also provides a flexible means by which a teacher can select the components and define the study modes with which he/she wishes to teach the course.

This functionality is achieved by building the authoring system around the basic data structures of the Learning Shell. Using these data structures the authoring tool embeds algorithms that automatically position and name learning materials so that the Learning Shell will function correctly.

For the teacher, authoring a course is primarily a question of systematically re-organising their teaching materials from existing internal and external courses into a suitable form to be integrated into the Learning Shell. This information engineering

does not require any specialised programming or other technical knowledge.

The authoring process follows similar lines to what is necessary to transform internal teaching materials into a successful paper-based extramural course. Where it goes further, is in the greater range of materials and media that can be integrated into a computer-based course, in providing a template for structuring the course, and in ensuring that the author provides all the essential learning elements for every module.

At the teacher level, reusability primarily means that a competent teacher without previous programming experience is able to author and update a course using a generic shell provided by a programmer. Reusability features supported include:

- Modes can be defined and built to templates using re-usable components with settable properties.
- Internal teaching materials can be re-used. Lectures, for example, can be recorded in class and provided to external students together with notes provided to internal students.
- Tutorials and other interactive learning programmes in a runnable format (e.g. Web or Windows) can be picked up and linked into the course.
- Discussions and queries can be edited and made available to students as part of the help system.
- A facility for updating courses is built into both the authoring and learning applications at minimal effort for student or teacher.

For the student, the Learning Shell works for any course. The Shell is independent of the course content. Using the Shell for a different course simply involves loading a different set of resource files.

The current prototype operates with only one course at a time. However, with a separate folder structure for each paper's learning resources and database, and a few modifications to the source code so that the Shell can be pointed to the correct folder system at logon, it could be used for simultaneously studying several courses.

The authoring application was used successfully to install a section of a real extramural course for the user testing of IMMEDIATE discussed in Chapter 7.

6.8 The IMMEDIATE prototype

Once the constituent applications had been coded and tested individually, they then needed to be tested in relation to each other. An important part of the incremental

prototyping process has been assembling the applications into the overall IMMEDIATE system, and then testing and modifying them so that they worked together to support all the required functionality of the Learning Shell.

To begin with, IMMEDIATE was assembled and debugged on a single computer. Then, the university-end components and the Learning Shell were installed on separate PCs and modified to run over a peer-to-peer LAN.

This section concentrates on the main challenges to get IMMEDIATE to run over a network. These were:

- Integrating the three component applications (student, authoring and repository management) and the FTP server into a working prototype running on an Ethernet LAN. On the network level this was quite straightforward. It involved linking two PCs running Windows 98 using a 10Base2 Ethernet coaxial bus (Tanenbaum, 1996, p. 277), and configuring the PCs to communicate over TCP/IP protocols. Within the client applications (Shell and authoring), the hard coding of the FTP server's IP address was changed to a parameter to be read from a set-up file.

As an application, the Learning Shell was easily installed, together with its folder system and data files, from CD or the Internet. It could then be accessed from an icon on the Windows Desktop.

The Delphi installation utility was used to create a set of installation files, which were used to install the Shell on the second PC. This highlighted some issues caused by variations in graphics capabilities (e.g. available colours) and screen resolution. A lowest common denominator approach has to be taken to the use of colour in the Shell. In relation to varying screen resolutions, all interface components must be designed for the lowest resolution, an algorithm has to be introduced to resize screen objects to match the resolution on the user's computer, or the user has to be guided to set the screen resolution to the optimal level. For the purposes of this initial prototype, the latter strategy was assumed.

- Installing a section of an extramural computer science paper, using the authoring application. This proceeded smoothly. But as the course author was also the system developer, it could only be a test of the application's functionality, not its usability.

The main change introduced at this stage was to provide for a wider range of materials to be displayed by learning components, for instance, multiple web pages instead of a single document. This entailed altering the resources directory

structure of the Learning Shell. And it required additions and alterations to the authoring system to support the functionality added to the Learning Shell e.g. enabling a hierarchy of folders (as with web pages) to be added to course material in the same manner as a single file.

Improvements were also introduced at this stage for updating the Extramural Support database in response to student queries and communications. This is discussed under “Authoring and Updating the Support System” (6.7.3).

- A self-evaluation of the prototype for functionality using the set of scenarios that defined the main functional requirements of the Learning Shell (Appendix D).
- Modifying and debugging the prototype until it satisfied the essential functional requirements in these scenarios. This is discussed in the following sections.

6.8.1 Modifications to the prototype

Considerable effort was directed at this stage to perfecting the methods by which the three core applications communicated with each other over the network so as to accurately transfer data and maintain consistency across the overall system. Modifications of all three component applications were made to simplify updating of course materials and the extramural support database by the student, and to provide more visual feedback. Many of these changes have already been discussed.

Some changes also had to be made at this stage that were required by technical limitations of the tools used. For example, the database engine that Delphi provided for free distribution with an application supported a smaller subset of SQL functionality than the version provided with the programming environment itself.

(a) failed query:

```
select MAX( entered ) from Messaging
where F_From = : F_From;
```

(b) replaced by selecting the first member of result set from this alternative:

```
select entered from Messaging
where F_From = :F_From
order by entered desc;
```

Figure 6.31: SQL query rewritten to run outside Delphi environment.

For instance, updating of messages between the Learning Shell and Repository databases relied on use of the MAX function in SQL (6.5.2). But the call to MAX function failed in the distributed version. This meant that when the Learning Shell was

installed on a separate computer synchronisation of data between the databases failed. Because this problem only occurred outside the Delphi IDE, it was very difficult to locate. Once located, however, it was easily solved by rephrasing SQL queries where necessary, as in Figure 6.31.

The final network specification for IMMEDIATE is shown in Figure 6.32.

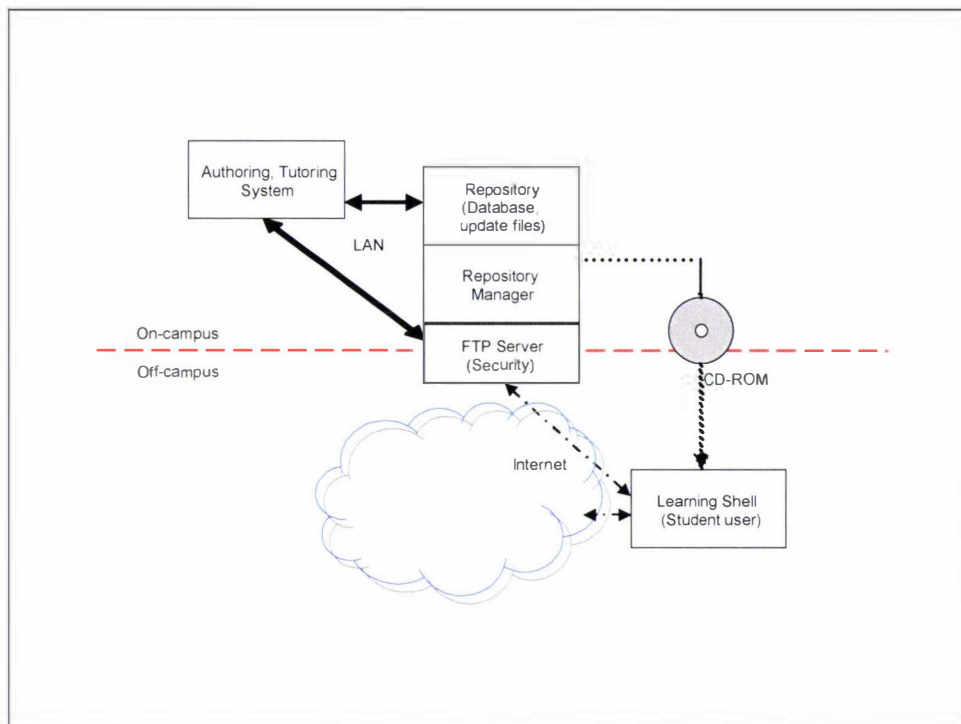


Figure 6.32: IMMEDIATE prototype implementation.

6.8.2 Debugging

The difficulties in troubleshooting a relatively simple SQL problem highlights how challenging it can be to locate and fix errors in programming logic and semantics in a project of this scope.

One of the advantages of software development in Delphi is the programming environment's comprehensive set of integrated debugging features, including being able to step through source code line by line while the program is running. However, implementing some of the requirements of the Learning Shell inhibited debugging using the Delphi tools.

For example, shutting out Windows operating system features while the Learning Shell is being used had the side effect of shutting out Delphi's debugging windows as well. This function of the Shell had to be switched off during debugging sessions.

And, for debugging the complex logic of the overall system – especially the interactions between the different applications -- it was often more helpful to trace component interactions using the well-proven method of writing status messages direct to the screen and to a log file as the application ran, using the method:

```
procedure tellUser( component, msg: string; logIt, tellIt: boolean );overload;
```

available in the Shell's System Utilities unit (Appendix F). Special debugging code is inherited or added to screen forms to support these features and enable Delphi's own debugging tools to work unimpeded.

This code is activated by a DEBUG variable in the controller object which activates a DEBUG variable in parent component forms, through which it is passed to its child forms. Alternatively, by directly activating the DEBUG variable in a parent component form, the programmer can troubleshoot that part of the code, while the rest of the application functions normally.

The tellUser method has also been used to provide an application-specific standard format to display a message to the student user where necessary, such as if an unexpected error occurs.

By using these debugging features to track user interactions with the Learning Shell, the Log File became an evaluation tool during user testing.

6.9 Conclusion

To implement IMMEDIATE for the purposes of evaluating the learning computer hypothesis, three separate applications have been built and then networked with an FTP Server into a working system. They are a Learning Shell, a Course Authoring and Management Application and a Repository Manager. This chapter has summarised that experience.

Prototyping IMMEDIATE proved to be a substantial programming task involving more than 25,000 lines of source code. Often, several different approaches needed to be tried before a satisfactory way of meeting a specification was found.

Perhaps the most challenging aspect of this prototyping work was developing and implementing a method for communicating and transferring data files over an unreliable network, while shielding the complexities of this process from the student user.

Using incremental prototyping, all the main requirements and functionality were successfully implemented. From an initial evaluation over a LAN, it was concluded that the prototype has demonstrated the technical feasibility of the learning computer

concept, at least under laboratory conditions.

Once IMMEDIATE was up, running and fully functional over a LAN, the next task was to install and test it under realistic field conditions. This aspect of prototyping and refining IMMEDIATE was combined with the user testing and is covered in Chapter 7.

Chapter 7

Use and evaluation of IMMEDIATE

The focus of this chapter is the testing and evaluation of IMMEDIATE with users in a small farming and fishing community in the North Island of New Zealand. This will be discussed under the following headings:

- Goals
- Strategy
- Installation and evaluation
- Results
- Conclusion

7.1 Goals of evaluation phase

Preece et al. (2002) define evaluation as “the process of systematically collecting data that informs us about what it is like for a particular user or group of users to use a product for a particular task in a certain type of environment” (p. 317). In an interview from the same book, Shneiderman distinguishes between an experiment and usability testing. The goal of an experiment is to confirm or refute a hypothesis using repeatable, quantitative means, while the goal of usability testing is to refine a product as quickly as possible, through identifying frequent problems (ibid., pp. 457-458).

In Chapter 3 it was hypothesised that to successfully provide a computer-based university-level distance-learning environment on an anywhere, anytime basis, three student-centred design strategies should be emphasised. They are: a distributed, rather than a centralised, network architecture; a user-initiated adaptable and collaborative, rather than system-initiated adaptive, approach to presentation and content; and a special-purpose, rather than a general-purpose, GUI environment. In Chapter 4 a learning computer was proposed as a conceptualisation of these strategies, which was then partially specified in Chapter 5 as the IMMEDIATE system.

The successful prototyping of IMMEDIATE was summarised in Chapter 6. This included testing the system on a peer-to-peer LAN. This experience furnished a more detailed specification for a learning computer implementation and provided provisional

support for the underlying hypothesis. To provide more conclusive evidence, it was important to evaluate IMMEDIATE under conditions more closely approximating those faced by its target users.

It was argued in Chapter 3 that extramural e-learning environments should be evaluated along the three dimensions of functionality, accessibility and usability (3.2). The goal of the evaluation phase was to confirm or refute the hypothesis, by evaluating IMMEDIATE along all three dimensions. This would require testing with users under realistic conditions, using quantitative and qualitative measures.

7.2 The Evaluation Strategy

Several key issues had to be decided in determining the evaluation strategy – what was to be evaluated, how, where, and by whom. In line with the goals, the evaluation would focus on assessing the completeness, accessibility, and usability of the Learning Shell functionality. The emphasis would be on the student as *user* rather than as *learner*.

The evaluation would combine aspects of a field trial and of more formal usability testing. It would be conducted with volunteer users representative of extramural students, who would work under field conditions reflecting the more challenging end of the spectrum in which distance learning takes place. If IMMEDIATE worked and was usable under these conditions, then it could reasonably be asserted that it would work and be usable anywhere, anytime. At the same time, each volunteer would be required to complete the same set of prescribed tasks under the same conditions. In this way, their interactions with the system could be monitored and compared in a measurable way.

The testing with users would have two facets. It would test the completeness and accessibility of the system functionality, primarily using quantitative data. And it would assess the usability of the Learning Shell interface, using quantitative and qualitative means. It would be important to demonstrate with quantitative data that the system was accessible, functional and usable in the field. But it would also be valuable to obtain the views of the users on IMMEDIATE and how it compared with other computer systems they have used.

7.2.1 Focus on student as user

The evaluation would focus on the student as user, i.e. how easily he or she could learn to use the system and carry out all the required tasks in the Learning Shell

environment. It would not address the issue of the student as learner – i.e. how well the user could learn the course subject matter.

IMMEDIATE has been designed to overcome the usability and accessibility problems associated with web-based and standalone educational software, by providing a system which makes minimal technical demands on the student user, both in terms of technological platform and computing experience. It does not introduce new learning or teaching methods that need to be evaluated.

What needed to be evaluated was the ability of the system to deliver a wide range of learning functionality in a usable form to remote students, and therefore that it was a workable, universal alternative to correspondence-based extramural study. The emphasis was upon the completeness of the functionality, on its *form* rather than its *content*.

7.2.2 Testing for functionality and accessibility

Evaluating IMMEDIATE for functionality and accessibility involved testing whether the three core components of the system – the Learning Shell, the Course Authoring and Management System, and the Communications Manager – could deliver the appropriate functionality in a timely manner to remote users, and assessing the reliability and performance of the system under these circumstances. For students to use IMMEDIATE for learning, all the three subsystems had to work together effectively.

This required installing and running the entire IMMEDIATE system in the field with actual users, and observing the results. This would be combined with the usability testing and observation.

Event logs would be used at both the client and server ends of the network to record quantitative data verifying that all functions worked and were accessible, and to identify and analyse any problems that might arise.

7.2.3 Usability testing

The centrepiece of IMMEDIATE is the Learning Shell. Therefore, considerable emphasis in the evaluation phase was placed upon assessing the Shell interface for usability. Would an extramural student studying at home alone be able to complete a set of typical tasks unaided (e.g. attend a lecture or participate in a group discussion)? How easily could they achieve this?

The evaluation method selected was scenario-based usability testing, because of its focus on what the user does (Pressman, 1997, pp.654-55). This usability testing would not be directed to identifying improvements in the interface, but to provide evidence in support of the learning computer concept. The scenarios and use cases drawn up as part of the design process would form the basis of the test scenarios.

Two criticisms of usability testing are that it emphasises first-time usage and cannot cover all the interface features in the space of a couple of hours (Shneiderman, 1998, pp.132). However, in the Learning Shell evaluation, the primary objective was to test the ability of an inexperienced user to get started with the system and complete the basic tasks in a couple of hours with minimal outside help.

7.2.4 Field testing

Usability testing would usually be conducted in a specially designed laboratory where the users can be carefully monitored. However, for the purposes of testing for functionality and accessibility, as well as usability, it would be difficult to reproduce all of the conditions faced by extramural students in a laboratory. Is the system runnable where the Internet is slow and unreliable? Are its features usable when more experienced help is not immediately at hand?

For this reason, the field test approach to usability testing (Shneiderman, 1998, pp.131) – where the system is installed, run and tested with users in a more realistic environment – was adopted.

The field test would involve the installation of the system over a telephone network, the selection volunteers, a pilot study with one user and an in-depth study with the remainder. Data would be collected for analysis using observation, log files, and interviews.

The criteria decided upon for the field test environment was that it be conducted using older computer hardware in a rural area where slow and unreliable Internet connections were the norm, with volunteer users working in isolation from one another. Equally important would be the choice of suitable volunteer users for the evaluation. In this way, the Shell could be evaluated under field conditions approximating the more difficult environments faced by university-level distance students to determine the universality of the underlying approach.

7.2.5 Purposive sampling

It is convenient to test educational software developed by university researchers with volunteers drawn from the university population at hand. Quite often these turn out to be undergraduate information science students. This choice of user would have been problematic for this evaluation, because the demographics of extramural student users differ in important respects from internal students – involving more who are “second-chance” learners, older, studying in isolation from their peers, or less computer-literate. It was necessary to verify that IMMEDIATE met the requirements of extramural learners.

It was therefore important to select volunteer users who more closely-matched the profile of the extramural student. It was decided to select them on the basis that they lived in a rural area, had distance learning experience and some knowledge of computers. The relevant information would be obtained by a questionnaire. This is an example of purposive sampling (Yin, 1984; Patton, 1990) where appropriate individuals who meet the specified requirements are selected. It was necessary to find individuals who had the potential to successfully complete the exercises under conditions similar to those faced by a distance student, working unassisted from a remote location.

7.3 *Installation and evaluation*

7.3.1 Test environment

During October 2003 the prototype was installed to run over the telephone network in an isolated coastal farming and fishing community in New Zealand. This rural community was chosen for the field test because its profile reflected some of the most difficult conditions that may be faced by those who undertake distance study from necessity rather than choice. From a telecommunications standpoint, these conditions are often more reminiscent of those associated with towns and villages in developing countries than to the “wired” urban environment in which the World Wide Web flourishes.

The selected community consists of approximately 40 permanent families scattered along a 10 kilometre stretch of coastline and throughout the surrounding hill country, about one hour’s drive (75 kilometres) from the nearest town of some 5000 people. The main economic activities are farming, fishing and forestry. The community is centred on a small school of 20-30 pupils located along a beach, and surrounded by a couple of

dozen holiday homes mostly owned by people living outside the district. High school students must board out of the district or study by correspondence.

From informal discussions with residents, it was evident that a number of adults, especially women, were involved in correspondence-based home study. In part this reflected a situation in which there were few local opportunities for female employment.

Conditions typical of many of the more remote parts of rural New Zealand – hilly terrain, stormy weather conditions, interference from agricultural equipment such as electric fences, and ageing, over-extended and unreliable telephone and electric power infrastructure – created a difficult environment for Internet computing. There was little advantage in obtaining the latest computer equipment. Anecdotal evidence indicated that while many families owned computers, they were often older models which they had taught themselves to use, if only to a limited extent. Often they were mainly utilised by the children in the household. Adult usage was mainly for email, games, some farm management and study tasks, and – for the patient – Internet browsing.

A drive around the district would show that most homes have satellite TV dishes. Broadband satellite reception for Internet downloading was therefore feasible. However, uploading by satellite broadcasting was still prohibitively expensive. Even if the means was found to fund a full broadband Internet service for the school, which would have to be satellite-based, there would still be no feasible means for linking up most of the farm households to it.

7.3.2 Installation of the system

In specifying IMMEDIATE (5.7) it was stressed that, while a wide variety of communication media such as satellite download were embraced by the learning computer concept, once the necessary hardware and their drivers were installed on a computer, all these media appeared to the user as either an internet connection or a portable storage device. Therefore, to simplify the prototyping process, the options implemented in the Learning Shell were updating by dialup internet connection or by portable disk. This limitation would only have a minimal impact in assessing the usability of the Learning Shell. It would only adversely affect the Shell's performance for those few functions which involved real-time Web access such as searching the university's library resources. Most web-based material accessed by the Shell is stored on the user's computer and could be updated by CD.

For the experiment, a client/server network was chosen with the communications management system installed on the network server, the teaching application installed

on a separate PC connecting to the server via a LAN, and the student application connecting from a remote location via the telephone network.

It was important that during the pilot study and the field test, both ends of the network be easily accessible to the researcher. Therefore, it was not feasible to install the server end at the researcher's university, several hour's drive from the field test locality. Instead it was installed at a farm building several kilometres inland from the user's locality. The farm was chosen because the necessary hardware was available there, and because it offered no advantage in the speed or reliability of the dialup connection over an internet connection to the university.

However, for the Learning Shell to connect to the server via an ISP, the server was required to obtain its own publicly-usable IP address. It was not possible to organise this for the trial. Therefore, the Learning Shell had to be set up with two dial-up connections – a connection to an ISP for accessing web pages, and another to the IMMEDIATE server. This required installing the university end applications on a LAN to which the Learner Shell connected as a dial-up client. The IMMEDIATE components could then communicate with each other using IP addresses reserved for private networks.

The installation steps were:

- Install the student application at the local school on a PC running Windows 95, using CD-ROM download. All volunteers would use the one PC. To preserve the integrity of each volunteer's data, each would have their own copy of the Shell and data files installed in a separate folder system. This would require the re-setting of the computer between each user.
- Establish a client/server LAN at the farm building using an operating system that supports the server end of dialup networking. Windows 2000 was chosen for this.
- Install the university end applications on the Windows 2000 LAN. The communications management subsystem (FTP server, Repository Manager) would be installed on the PC running the LAN server and the authoring application would be installed on another client PC.
- Set-up a dial-up connection between the student and university end over the local telephone lines.

Installing the Learning Shell at the school was a three-step process:

- Install a file compression and decompression utility. This is provided as a standard service in Windows XP, but earlier versions of the Windows operating system

usually require third party software to provide this service. In this case, WinZip was used.

- Load a full version of the database tool provided with the Delphi development environment. This would not normally be required because the database engine is part of the Shell application installation. However, installing the full database tool allowed direct access to the database independently of the Shell application, to locate and fix any unexpected problems should they arise during the pilot study and subsequent usability experiment.
- Install the Learning Shell application, including the folder structure and course materials. This involved activating a self-extracting compressed file from a CD, using the previously installed decompression utility.

All the steps in installing IMMEDIATE proceeded smoothly. The few minor problems that arose mostly related to the configuration and administration of the Windows 2000 network. No outside technical support was used at any point during the prototyping, installation and evaluation of IMMEDIATE, so a little experimentation was needed on occasion.

7.3.3 Pilot study

Four members of the rural community serviced by the school made themselves available to assist in evaluating the prototype. They were provided with an information sheet (Appendix K1) and signed a consent form. Each completed a copy of the questionnaire profiling his or her computer and learning experience (Appendix K2). Table 7.1 presents a summary of the questionnaire results.

All four volunteers had studied at the tertiary level and three were current or very recent tertiary distance students. One agreed to make herself available for the pilot study. The objectives of this pilot study were:

- To ensure sufficient course material had been incorporated into the prototype to enable all essential functionality to be accessed and tested by the user.
- To detect and fix as many usability problems as possible, that might otherwise impinge on evaluating the underlying conceptions of the Shell.
- To refine a set of exercises (scenarios) that takes the user through all major aspects of the system.

1. Username: ¹	Pilot User	User 1	User 2	User 3
2. Gender (circle one): Female / Male	F	M	F	F
3. Are you currently studying at university or polytechnic level? (circle one) Y / N	Y	N	N	Y
4. If not, have you studied at university or polytechnic level? (circle one) Y / N	NA	Y	Y	NA
5. Major subject	Education	Education	Communication	Marketing
6. Have you ever studied by correspondence (extramurally)? (circle one) Y / N	Y	N	Y	Y
7. Computer usage:				
In what year did you first use MS Windows?	1996	1996	1999	1992
In a typical week, how many hours, including work and leisure use, would you spend at a computer?	8	8	10	8-10
Now please detail how many hours per week on average over the last year you have used computers for the following tasks, by circling the appropriate number:	Hours per week	Hours per week	Hours per week	Hours per week
Write and send emails	1	1	2	5 or more
Browse the internet for news/information	1	1	4	5 or more
Write letters or reports with a word processing program (e.g. Word, WordPerfect, MS Works, etc.)	3	3	4	5 or more
Keep accounts and budgets with a spreadsheet program (e.g. Excel, Lotus 123, etc.)	0	0	0	0
For your own university or polytechnic studies	2	0	4	5 or more
To help others (e.g. your children) with their education	2	3	2	5 or more

Table 7.1: Summary of Participant Profile Questionnaires. ²

¹ For the purposes of logging onto IMMEDIATE the participants used the aliases "Mike", "Bob", "Mim" and "Pom". In the text, they are referred to as Pilot User, User 1, User 2 and User 3.

² All participants gave an estimate of the total time spent on the computer each week which was significantly less than the sum of the total time they estimated they spent on each task category. This suggested a considerable overlapping of tasks or that their estimates were very approximate. When asked about the discrepancy, two of the respondents expressed surprise and said it meant that they probably spent more time on the computer than they had thought.

Test scenarios

The scenarios covered setting up the student Learning Shell and exploring all aspects of its functionality, including accessing learning material in six different study modes: lecture, group work, tutorial, textbook, collaboration, and assignment. The volunteers were expected to complete these in two one-hour sessions spread over two days. The first exercises were quite detailed in their instructions, the later ones, progressively less so. The sequence of activities was as follows:

1. Initialise Learning Shell to own preferences
2. Log on to course
3. Join Group Discussion
4. Attend Lectures, Seek Help, and Complete Self-Assessment
5. Monitor The Assignment Discussion
6. Explore On-line resources
7. Complete Individual Tutorial

The "Attend Lectures" scenario is shown in Figure 7.1. A walkthrough of this scenario, demonstrating the use of the Learning Shell, is provided by Figures 7.2-7.13.

All seven scenarios are included in Appendix J.

To ensure that all essential features were included in the trial, the scenarios were checked against the nine potential features of computer-based learning systems (2.3) and the six dimensions of university learning (3.2.3) identified during the literature review phase of the project. The results have been summarised in Table 7.2.

Out of the pilot study a number of modifications were made. Further materials were added to the course, so that all scenarios were fully supported. Some inconsistencies between screens were identified and corrected, such as placing Help and other function buttons in the same place on all interface components.

A "Handy Hints" page (Appendix J1) was prepared listing key tips for navigating and using the system, the Help Screens were reorganised to emphasise hints for using each learning component, and the exercises were edited to present each user task more clearly and logically.

E-Learning Enhancement	Scenario
Allowing learning material to be accessed from almost anywhere, at any time.	All
Facilitating communication.	3, 5
Promoting co-operative work.	3, 5
Integrating the various media used to deliver distance education (video, audio, telephone, mail, graphics, and text) into a single multi-media environment.	4
Simulating real world situations, processes and problems.	7
Integrating materials from a variety of sources and locations.	6
Allowing the presentation of material to be adapted to suit the individual student.	2, 6
Acting like a human tutor.	7, 4
Helping the teacher to author teaching material.	5
Learning Dimension	
Learning by textbook	2
Learning by lecture	4
Learning by exploration	4, 6
Learning by collaboration	3, 5
Learning by doing	3, 7
Learning by tutorial	7

Table 7.2: Scenarios covered all e-learning enhancements and dimensions.

The docking function, introduced during the latter stages of prototyping, was removed from the scenarios as it was clearly confusing to a less experienced user. It was felt that, while the idea of organising the components of a study mode into a coherent screen was worth investigating further, it would only work if it were performed by the Shell itself. It was also at this point that a feature was added to the Group Work component enabling all newly received messages to be viewed immediately by the student, without their having to navigate to the appropriate point in the course.

7.3.4 Organising the field test

The three remaining volunteers participated in the in-depth study. They were representative of the demographics of the primary target users for IMMEDIATE, viz. mature students, studying alone from remote locations with limited communications infrastructure, and with a range of largely-self-taught computing experience.

Access a Lecture, Ask for Learning Support

(Prerequisite: Ensure the computer is connected to university)

{In this scenario, you access a video presentation, seek support in understanding one of the concepts raised in the lecture, and then evaluate your understanding of these concepts.}

1. Log on to **Massey Extramural**
2. Use the **Course Explorer** to navigate to the lecture at Topic 1.2. (Click the **Help** icon or press F1 if you need guidance).
3. Load and start the lecture (For this trial it is only a sample clip. In a real situation you would follow the presentation with the help of the slides contained in **Lecture Notes**.)
4. Use **Desktop Menu** to open **Key Ideas**. This is the gateway to **Extramural Support**.

Getting help with Key Ideas

5. You are confused about "metaphor". Select this concept and open **Extramural Support**.
6. Press F1 and read the **Help** page.
7. Ask **Extramural Support** to explain "metaphor".
8. RETRY for further information.
9. Response is inadequate. Click NO.
10. ASK the tutor for more information on "metaphor".
11. Open the **Concept Map**
12. Select the related item "conceptual models" there.
13. Ask **Extramural Support** to explain "conceptual models".
14. RETRY.
15. Click NO.
16. Exit **Help**.

Self-Assessment

17. You decide to register how well you understand Topic 1.2. Note the colour of the icons next to each concept in **Key Ideas**.
18. Find and start the **Self-Assessment** questionnaire. Give a range of responses over the questions. Rate "metaphor" and "conceptual models" as poor.
19. Update **Key Ideas**. Note the changes to the colours of the faces to the left of the concepts.
20. Open the **Course Explorer**. Note the changed colour at Topic 1.2.
21. Change study mode to Group Work. Update your messages. Wait until the process is complete before continuing.
22. Exit **Massey Extramural**. Do not backup your work.

<End of Lecture and Ask for Help scenario>

Figure 7.1 : Attend Lectures scenario.

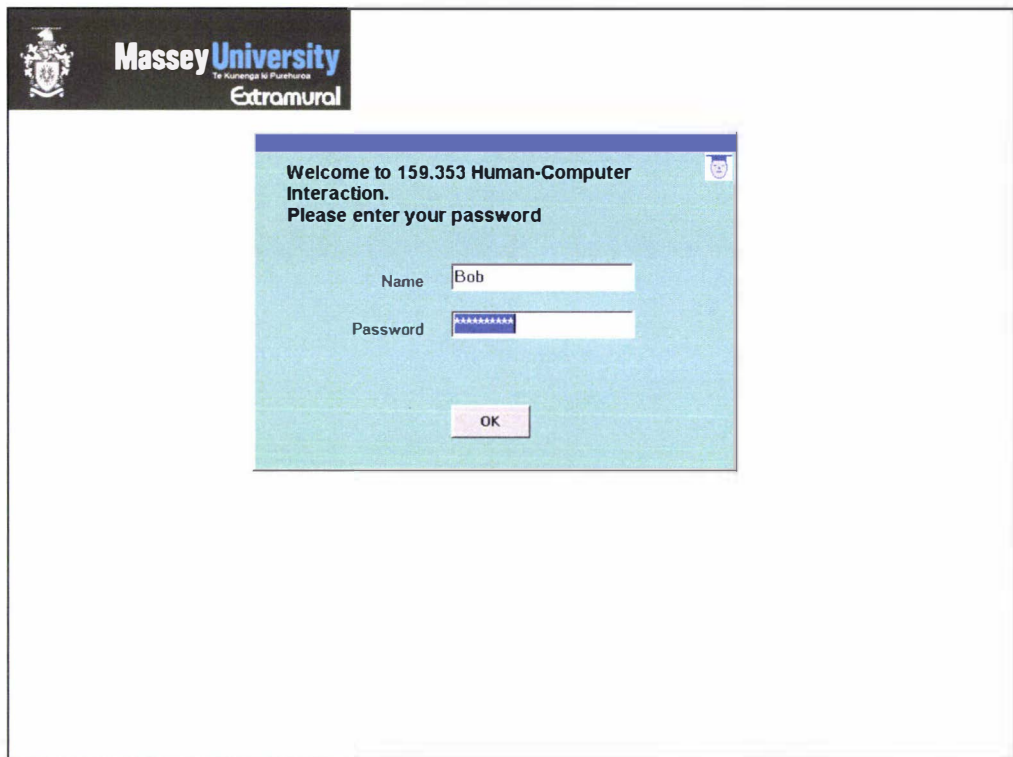


Figure 7.2: Logging on to the Learning Shell.

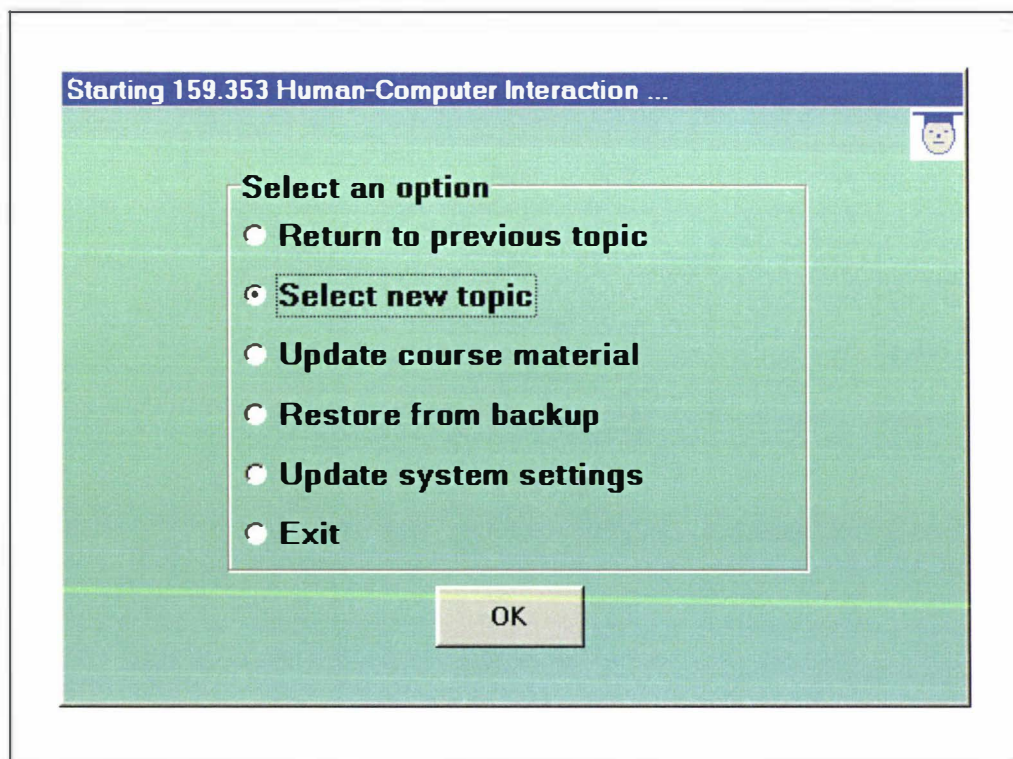
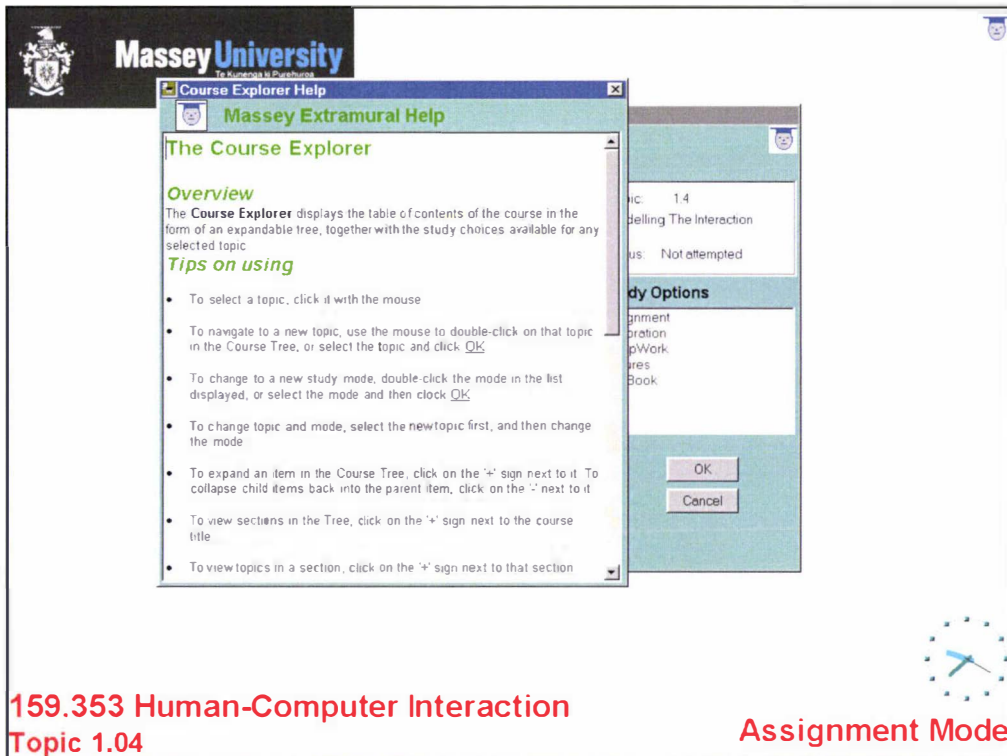


Figure 7.3: User Options form. "Select new topic" selected.



Massey University
Te Kōwhiri ki Pūwhiri

Course Explorer Help
Massey Extramural Help

The Course Explorer

Overview
The **Course Explorer** displays the table of contents of the course in the form of an expandable tree, together with the study choices available for any selected topic.

Tips on using

- To select a topic, click it with the mouse.
- To navigate to a new topic, use the mouse to double-click on that topic in the Course Tree, or select the topic and click **OK**.
- To change to a new study mode, double-click the mode in the list displayed, or select the mode and then click **OK**.
- To change topic and mode, select the new topic first, and then change the mode.
- To expand an item in the Course Tree, click on the '+' sign next to it. To collapse child items back into the parent item, click on the '-' next to it.
- To view sections in the Tree, click on the '+' sign next to the course title.
- To view topics in a section, click on the '+' sign next to that section.

Topic: 1.4
Modelling The Interaction
Status: Not attempted

Study Options

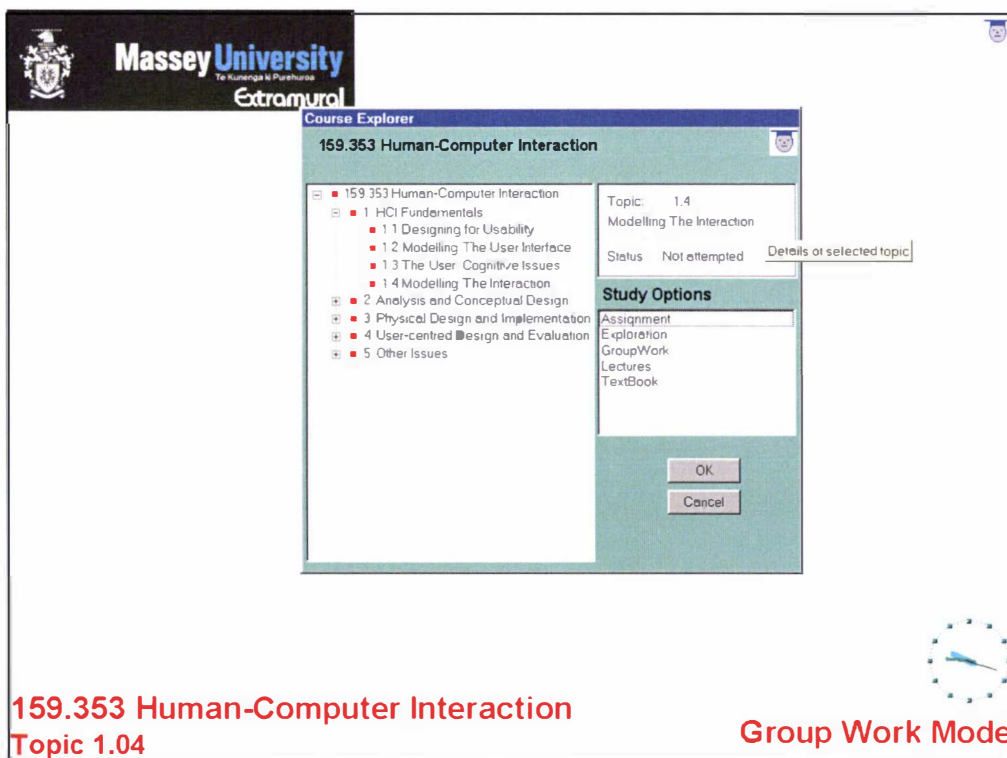
Assignment
Exploration
GroupWork
Lectures
TextBook

OK
Cancel

159.353 Human-Computer Interaction
Topic 1.04

Assignment Mode

Figure 7.4: Course Explorer. Clicking Help icon opens Help screen.



Massey University
Te Kōwhiri ki Pūwhiri
Extramural

Course Explorer
159.353 Human-Computer Interaction

- 159.353 Human-Computer Interaction
 - 1 HCI Fundamentals
 - 1.1 Designing for Usability
 - 1.2 Modelling The User Interface
 - 1.3 The User Cognitive Issues
 - 1.4 Modelling The Interaction
 - 2 Analysis and Conceptual Design
 - 3 Physical Design and Implementation
 - 4 User-centred Design and Evaluation
 - 5 Other Issues

Topic: 1.4
Modelling The Interaction
Status: Not attempted [Details at selected topic](#)

Study Options

Assignment
Exploration
GroupWork
Lectures
TextBook

OK
Cancel

159.353 Human-Computer Interaction
Topic 1.04

Group Work Mode

Figure 7.5: Navigating to Lectures Mode in Topic 1.2.

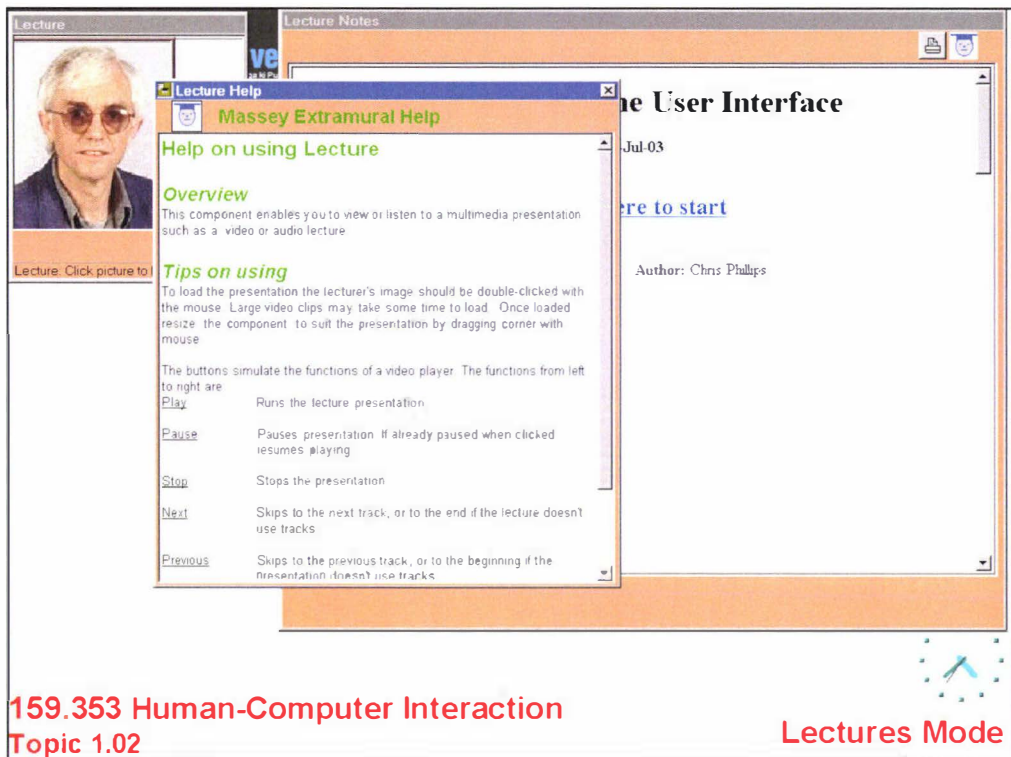


Figure 7.5: Help screen for Lecture component.



Figure 7.5: Right-clicking background with mouse opens Desktop Menu.

The screenshot shows a desktop application window titled "Lecture Notes". On the left, there is a "Lecture" window displaying a video of a person with the text "Lecture: Click picture to load". Below it is a "Key Ideas" sidebar with a list of topics: Conceptual models, Conceptual frameworks, Interaction paradigms, Interaction styles, Towards physical design, and metaphor. The main content area displays the title "2. Modelling the User Interface" with the date "08-Jul-03" and a link "Click here to start". Below this is a "Table of Contents" section listing various sub-topics like "Conceptual Design", "Scoping the Problem Space", and "Conceptual Models based on Activities". The author is listed as "Chris Phillips". At the bottom of the application, there is a "Computer Interaction" logo and the text "Lectures Mode".

Figure 7.6: Key Ideas opened via Desktop Menu.

The screenshot shows an application window titled "Extramural Support". The main area is a "Question" window with a dropdown menu set to "metaphor". The question text reads: "metaphor": Section 1.2 Metaphors support the application of the familiarity of the system implicitly by reference to familiar models. "metaphor": Section 1.1 the use of one idea or object to describe another. Metaphor is used widely in graphics to suggest that a computer screen is a physical space. A general physical space is a general physical space that is draggable. Metaphors are also useful to describe unrelated metaphors as a... Below the question is an "Ask For Help" dialog box with fields for "To:" (Tutor), "Cc:" (MyGroup), and "Subject:" (Section 1.2 "metaphor" Please help me on this). The message body says "Can anybody help me with 'metaphor'?". A "Help" dialog box is overlaid on top, displaying the message: "Your message has been saved, and will be sent next time you update your message list." with an "OK" button. At the bottom of the application, there is a "User Response" area with buttons for OK, Cancel, Retry, Ask, Map, Copy, New, and Exit.

Figure 7.7: Asking the tutor (and my group) for help with "metaphor".

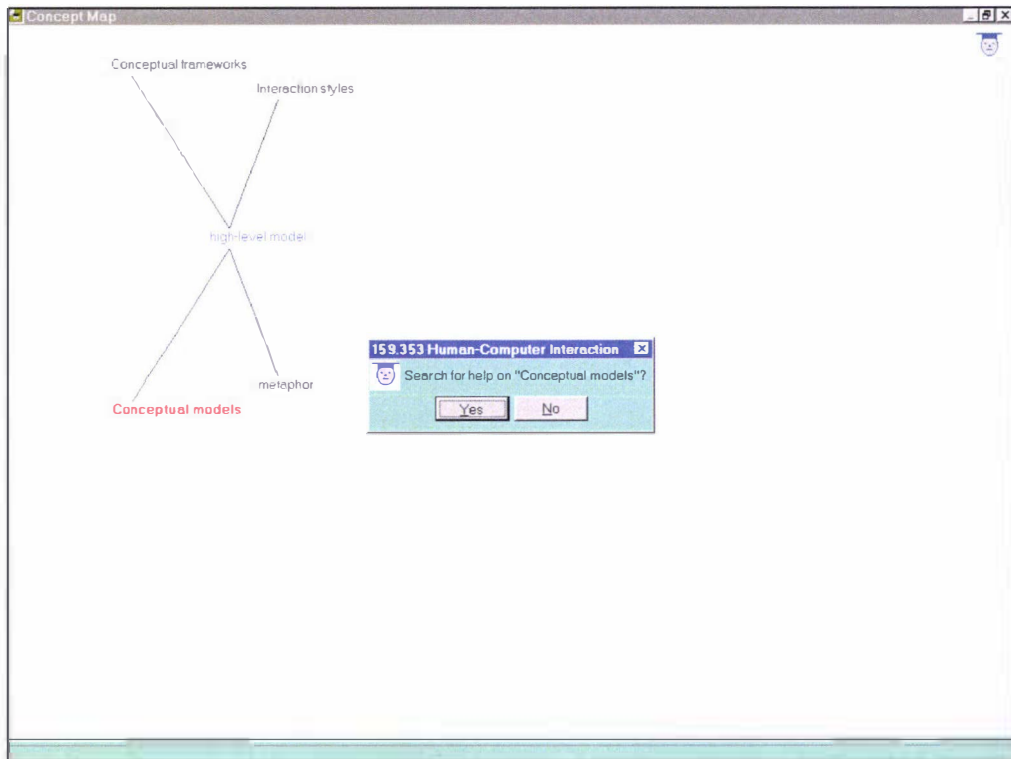


Figure 7.8: Selecting “conceptual models” in Concept Map.

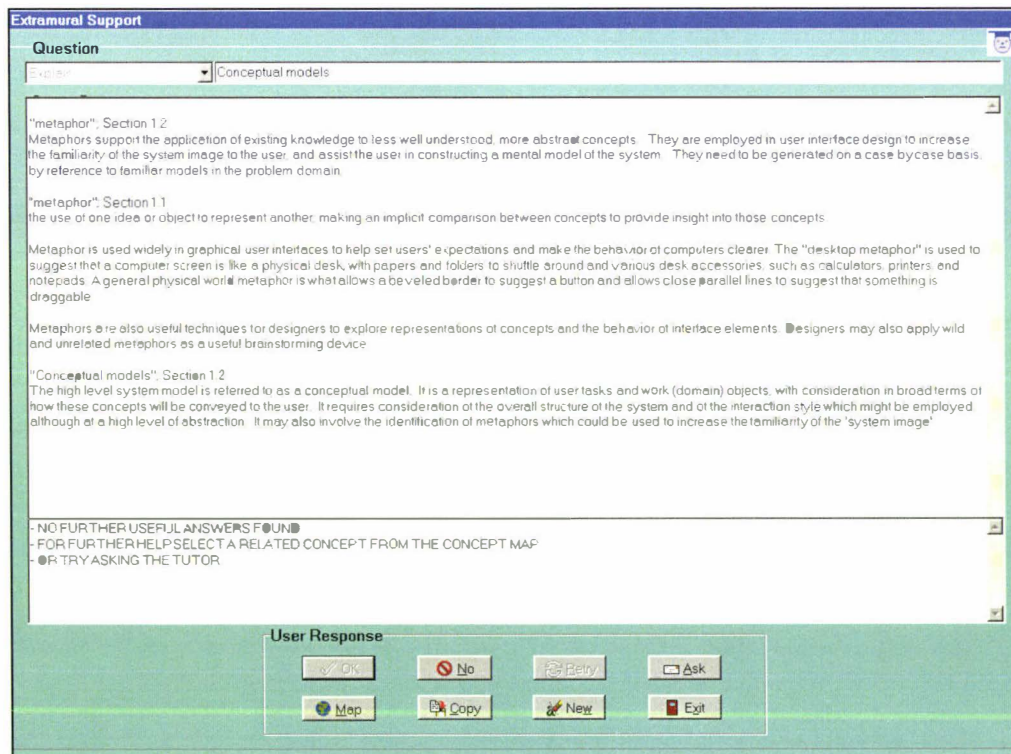


Figure 7.9: After Extramural Support has explained “conceptual models”.

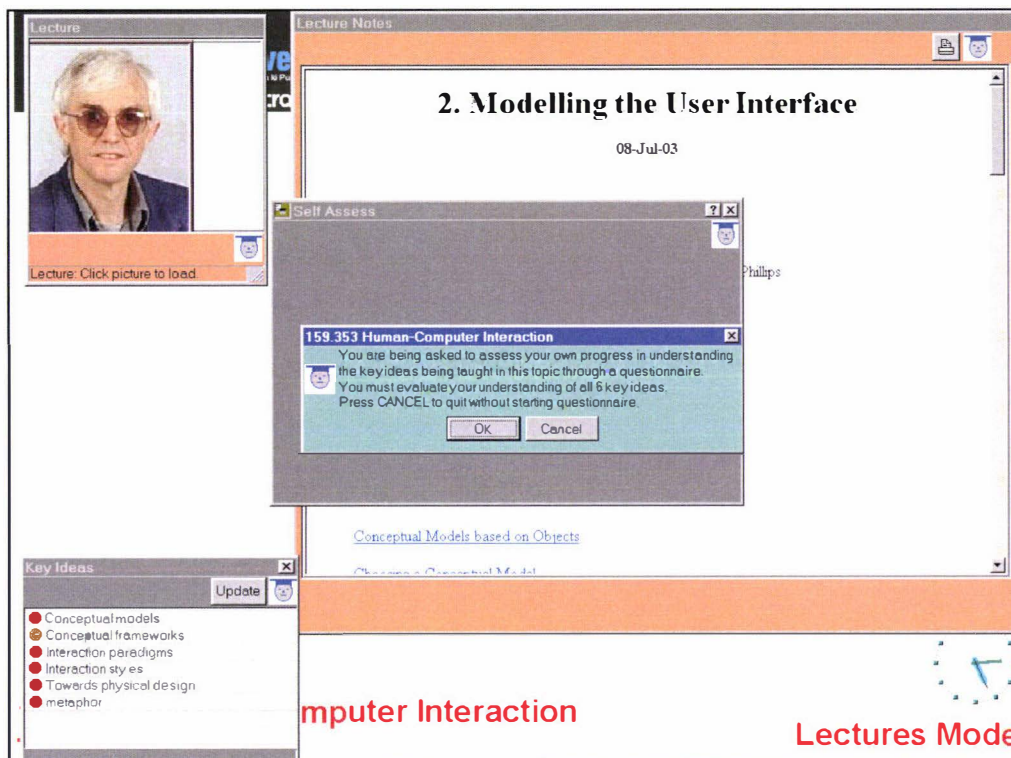


Figure 7.10: Starting Self-Assessment.

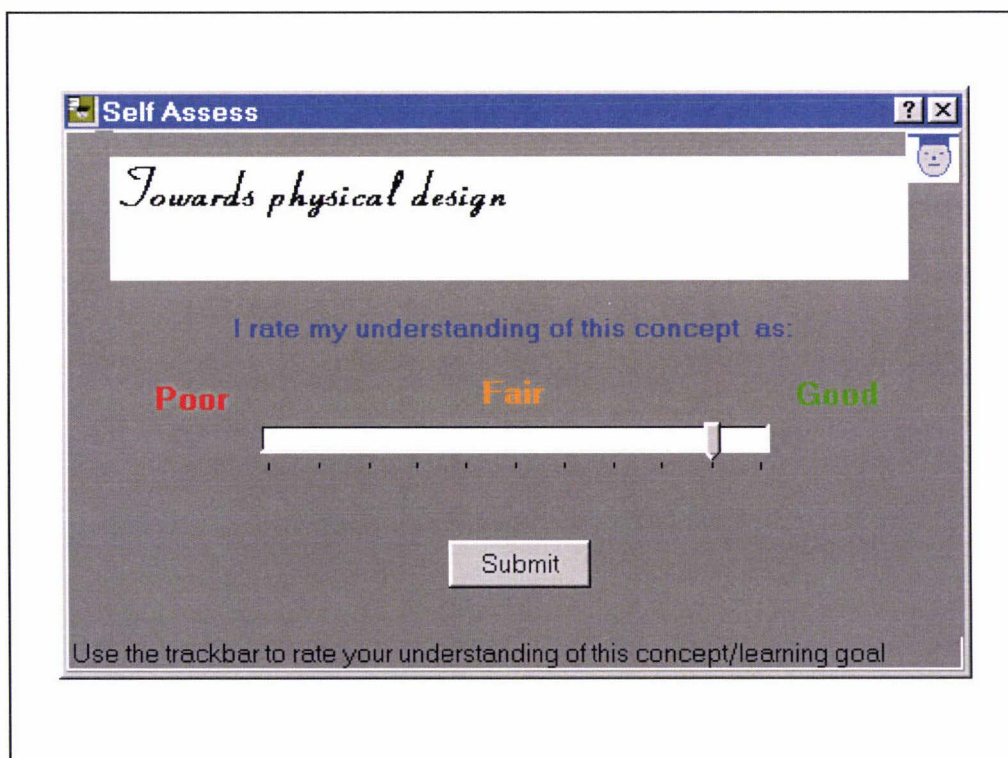


Figure 7.11: Self Assessment questionnaire component.

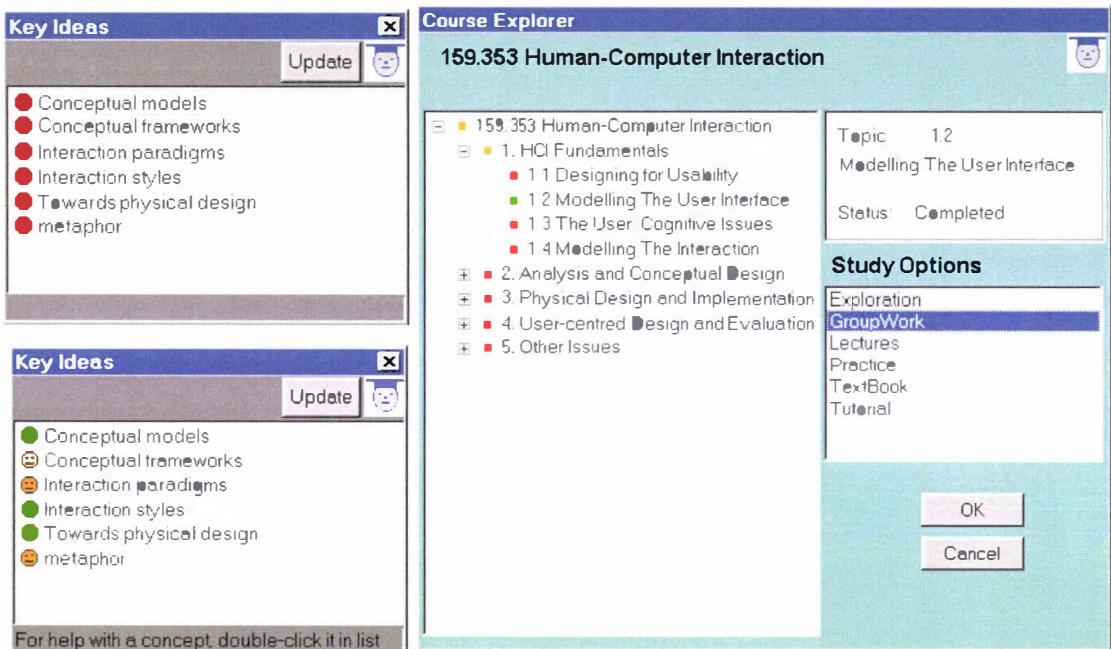


Figure 7.12: Key Ideas, Course Explorer are updated after Self-Assessment.

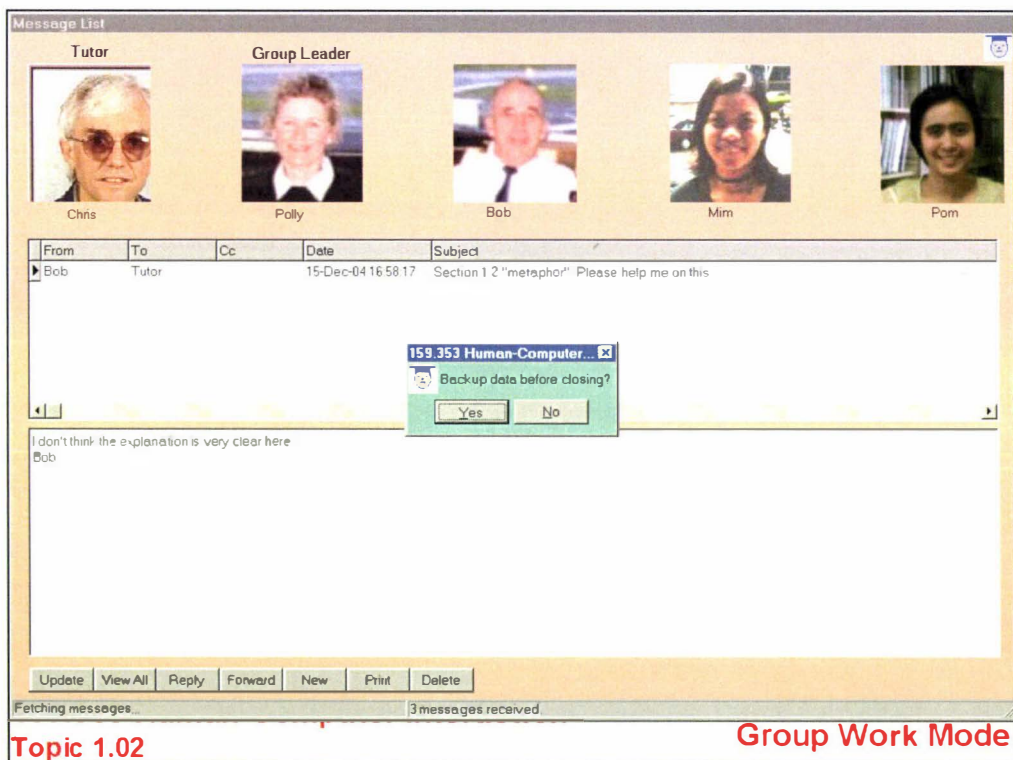


Figure 7.13: Exit after updating messages.

All three were adults in their late 20s to early 30s. Analysis of the questionnaire results (Table 7.1) showed that all the volunteers had had at least some computing experience ranging from 3-4 years to a decade. One (User 3) had been studying as a distance student with the NZ Open Polytechnic for a number of years and used her home computer to support her hard-copy-based course wherever possible. Another (User 1) had been an internal university student who now used computers for his work as a primary school teacher. The third and least-experienced computer user (User 2) had recently enrolled at the Open Polytechnic, but had subsequently put her studies on hold because of increased family responsibilities.

The three volunteers were asked to complete all seven scenarios. To ensure that no user was advantaged by knowledge of the course content, the material was taken from a third year university paper on a topic that none of them had studied. This was appropriate given the nature of the evaluation.

One issue that had to be faced was of training. When the system was fully developed, student users would be supplied with a CD-ROM containing the software and course materials, a demonstration video and a hardcopy guide. Since at this stage there was no demonstration video, some training had to be provided to the users.

Each user was given a demonstration of the student software as it would be shown in the video, and then completed the first two exercises under supervision. They were then left to complete the scenarios unassisted, except for the provision of the sheet of Handy Hints that would form part of the hardcopy guide.

Three forms of data collection were used during the experiment – logging their activities (Dix et al., 1993), observation and interviews (Patton, 1990, Scott et al., 1991). Where, when and what each user did during their sessions was tracked by the system and saved in a log file, primarily to analyse users' navigation paths and completion times, and record where they ran into difficulties, and to record and trace any errors in the functioning of the system itself. Because the student software is designed to support user-centred exploration and multiple navigation paths, a user should be able to complete a task even if they diverge from the shortest path that an experienced user may be expected to follow.

For each user, the following were observed while they worked through Scenarios 1 and 2:

- Their general demeanour before, during and after each scenario (confident, hesitant, anxious, frustrated).

- Their familiarity with the manipulation of screen objects with the mouse and keyboard (confident, hesitant, unconfident).
- Where they hesitated, where they used the Help icon, and where they asked for another demonstration.

Each participant was interviewed individually, immediately after they had completed each session (i.e. twice). Interviews were structured along the lines recommended in Scott et al. (1991, Ch. 14). A set of questions was prepared as the starting point for exploring each person's experience with the prototype. Questions, such as the difficulties faced, what aspects of the system they liked, and what they would want changing, were asked. At the same time, the interviews were allowed to digress where it helped to clarify the interviewee's experience with IMMEDIATE or other computer systems. These interviews were recorded and transcribed by the interviewer for subsequent analysis.

7.4 Evaluation Results

7.4.1 Functionality and accessibility

IMMEDIATE ran successfully and without major incident during the entire evaluation. The results were very positive, with all volunteers able to complete the exercises within the two one-hour sessions. Completion times, excluding wait times for downloads and updates over the network, ranged from just under 52 minutes for User 3 to a little over 89 minutes for User 2. All three stated that by the end of the exercises they were confident that they were able to use the system unaided.

The conclusion that all the features tested were functional and accessible was supported by the data collected in the log files.

Log Files

What each user did during their sessions was tracked by the Shell and saved in a log file. The log files traced the user's path through the scenarios by recording the state of the system after each significant interaction with the user. This included, the current time, topic, study mode and learning component, as well as the specific action called by the user and the system response to it (Figure 7.14).

An analysis of the log files revealed that, whilst each of the volunteers got held up at least once, they were all able to complete each scenario. Most difficulties were encountered in completing Scenario 3, which involved finding an Assignment, and then

asking the tutor a question concerning it using the electronic messaging system. User 3 completed this scenario in 16 minutes, which involved 19 logged events including accessing Help 6 times for 2 minutes in total. User 1 was only slightly slower in 18 minutes, but logged 38 events and accessed Help 10 times for a duration of 4 minutes. On the other hand, User 2 took 27 minutes, logging 26 events but only spent 1 minute in Help, accessing it 5 times. This user also had difficulties when accessing the Assignment mode the second time (Scenario 5).

```
13/10/03 5:17:07 PM: System launched
17 17 12: Logon: Pom logged on
17 17 24: UserOptions: Select new topic
17 17 24: CourseExplorer: Show
17 17 35: Controller: Change topic: 1.02
17 17 35: Controller: Change mode: Lectures
17 17 47: LectureNotes: Help requested for this component
17 17 52: LectureNotes: Help page closed
17 19 23: Desktop Menu: Learning Aids
17 19 25: KeyIdeas: Show
17 19 52: ExtramuralSupport: Concept = metaphor
17 19 53: ExtramuralSupport: Show
17 19 58: ExtramuralSupport: Help requested for this component
17 20 19: ExtramuralSupport: Help page closed
17 20 27: ExtramuralSupport: qWhat
17 20 40: ExtramuralSupport: User action: Retry
17 20 48: ExtramuralSupport: NotifyTutor system message saved.
17 20 48: ExtramuralSupport: User action: No
```

Figure 7.14: User interactions traced through log file (e.g. from Scenario 4).

Graphs were compiled to compare the time each user took to complete the scenarios (Figure 7.15) and the frequency with which they referred to the Help system whilst completing each scenario (Figure 7.16).

The log files also helped to locate one or two bugs remaining in the system, including an incorrect system status message that initially caused the volunteers to wait unnecessarily when updating their messages from the IMMEDIATE server.

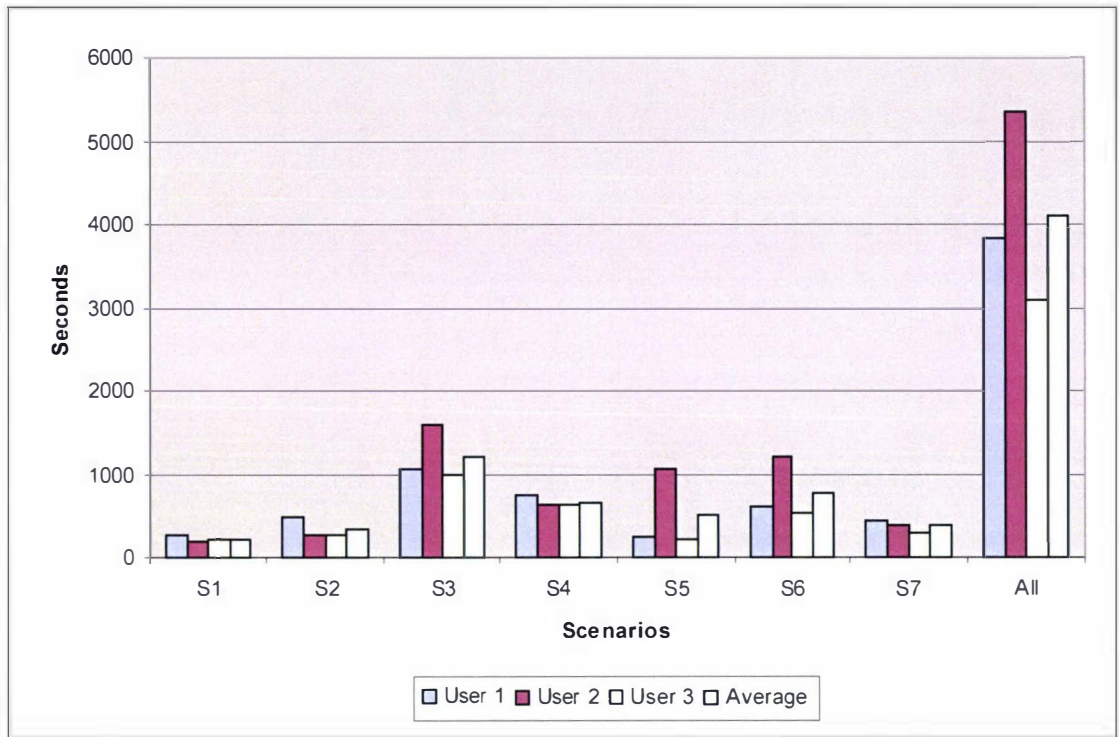


Figure 7.15: Comparison of scenario completion times.

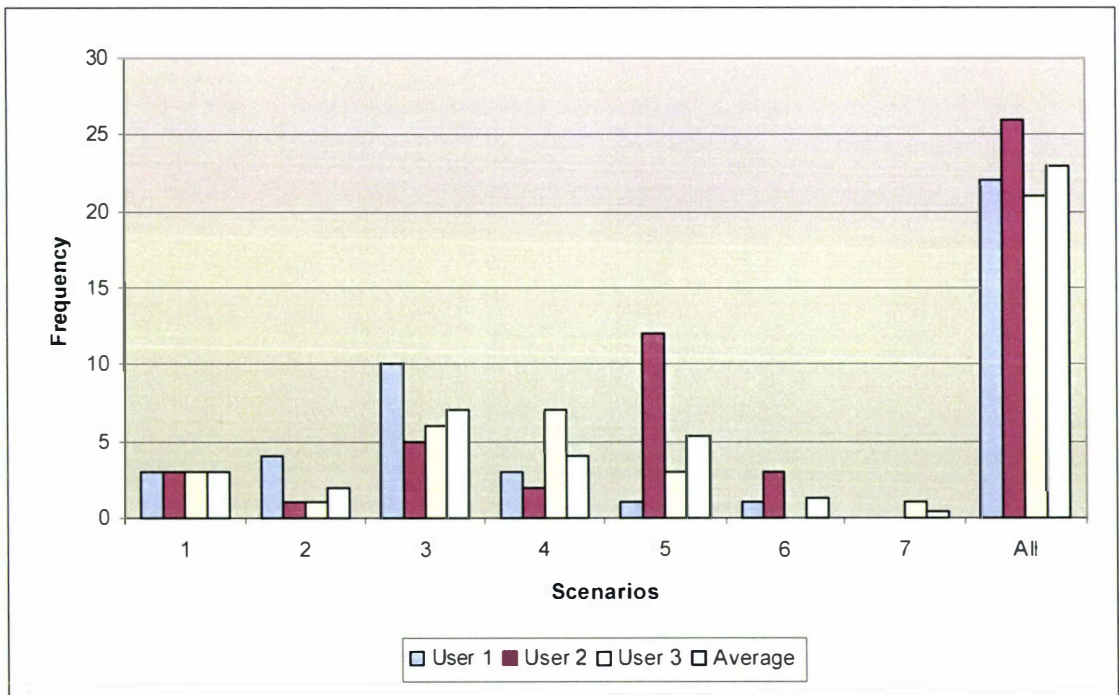


Figure 7.16: Comparison of help access frequency.

Authoring system

The field study provided an initial opportunity to assess the functionality of the university end of IMMEDIATE in a realistic setting. Supporting the three volunteer users involved:

- Sending messages from the authoring application to the student users, in response to their messages and queries;
- Updating the Extramural Support database from the authoring application, in response to student and system messages, for transfer to the Learning Shell;
- Adding new resources (assignment grades) to the Repository from the authoring application for update by the Shell.

These activities involved logging on to the FTP server over the LAN as a teacher user in order to send and receive messages from the authoring application, including communicating with the System Administrator. Updated help and learning resources files then had to be zipped at the Repository Manager and made available for downloading by the Learning Shell. These tasks were accomplished without any significant problems.

Communications performance and reliability

An important corollary of the user testing was to assess the performance and reliability of IMMEDIATE when it was run in the realistic environment provided by a rural telephone network. At the client end, all communications with the Repository server were recorded in the Learning Shell log. At the server end, client interactions with the Repository, including system responses, were logged by the Repository Manager (Figure 7.17) and the FTP server (Figure 7.18)³.

Analysing these files revealed that IMMEDIATE was able to handle communications between the students and the tutor, and allow learning resources to be updated, efficiently and reliability over the two days of the testing. The only unexpected delays detected were traced to a bug at the interface design, rather than any problem in the communications technology. No other issues adversely affecting IMMEDIATE's performance and reliability were identified.

³ In Figs 7.17 and 7.18, "Chris" is the course tutor.

7.4.2 Usability

The quantitative data provided by the logs, especially the comparable times taken by all users to complete all scenarios, provide evidence for the usability of the Learning Shell. However, the qualitative data, from the observations and the interviews, provided even more compelling evidence supporting the Shell's usability, and contrasting it with the volunteers' experiences using other computer systems.

The users were only observed working through the first two scenarios. The data from this observation was of somewhat secondary value because these two exercises were part of the supervised demonstration. However, it did reinforce the information from the volunteers' profile questionnaires regarding their previous computing experience.

```
13 06 57: Last received request from Mim
13 06 57: Last received request processed for Mim: 14 10 03 13 01 22
13 06 57: Deleting C:\VLM_Repository\Repository\MessageQueue\55099.txt...
13 06 57: Deleted
13 07 12: Message update requested by Mim
13 07 12: 3 messages received from Mim
13 07 12: 0 messages transferred to Mim directory
13 07 12: Deleting C:\VLM_Repository\Repository\MessageQueue\55112.txt...
13 07 12: Deleted
13 49 57: Last received request from Chris
13 49 57: Last received request processed for Chris: 07 10 03 21 30 31
13 49 57: Deleting C:\VLM_Repository\Repository\MessageQueue\57632.txt...
13 49 57: Deleted
13 50 12: Message update requested by Chris
13 50 12: 4 messages received from Chris
13 50 12: 4 messages transferred to Chris directory
13 50 12: Deleting C:\VLM_Repository\Repository\MessageQueue\57638.txt...
13 50 12: Deleted
```

Figure 7.17: Extract from Repository Manager log.

```

[L 2003 10 14 13:06] 00004 Mim cntr User from 192.168.0.4 logged in
[F 2003 10 14 13:06] 00004 Mim data C:\VLM_Repository\Repository\MessageQueue\55099.txt
Receiving.
[F 2003 10 14 13:06] 00004 Mim data C:\VLM_Repository\Repository\MessageQueue\55099.txt Received
file successfully. Size: 44 bytes. 0.043 Kbytes/sec
[F 2003 10 14 13:07] 00004 Mim data C:\VLM_Repository\Students\Mim\MessageBox\answer.txt Sending.
[F 2003 10 14 13:07] 00004 Mim data C:\VLM_Repository\Students\Mim\MessageBox\answer.txt File sent
successfully. Size: 44 bytes. 0.043 Kbytes/sec
[F 2003 10 14 13:07] 00004 Mim data C:\VLM_Repository\Repository\MessageQueue\55112.txt
Receiving.
[F 2003 10 14 13:07] 00004 Mim data C:\VLM_Repository\Repository\MessageQueue\55112.txt Received
file successfully. Size: 1813 bytes. 1.771 Kbytes/sec
[F 2003 10 14 13:07] 00004 Mim data C:\VLM_Repository\Students\Mim\MessageBox\messages.txt
Sending.
[F 2003 10 14 13:07] 00004 Mim data C:\VLM_Repository\Students\Mim\MessageBox\messages.txt File
sent successfully. Size: 44 bytes. 0.043 Kbytes/sec
[L 2003 10 14 13:07] 00004 Mim cntr User from 192.168.0.4 logged out
[L 2003 10 14 13:49] 00005 Chris cntr User from 192.168.0.2 logged in
[F 2003 10 14 13:49] 00005 Chris data C:\VLM_Repository\Repository\MessageQueue\57632.txt
Receiving.
[F 2003 10 14 13:49] 00005 Chris data C:\VLM_Repository\Repository\MessageQueue\57632.txt
Received file successfully. Size: 47 bytes. 0.046 Kbytes/sec
[F 2003 10 14 13:49] 00005 Chris data C:\VLM_Repository\Teachers\MessageBox\answer.txt Sending.
[F 2003 10 14 13:49] 00005 Chris data C:\VLM_Repository\Teachers\MessageBox\answer.txt File sent
successfully. Size: 46 bytes. 0.045 Kbytes/sec
[F 2003 10 14 13:49] 00005 Chris data C:\VLM_Repository\Repository\MessageQueue\57638.txt
Receiving.
[F 2003 10 14 13:49] 00005 Chris data C:\VLM_Repository\Repository\MessageQueue\57638.txt
Received file successfully. Size: 6612 bytes. 6.457 Kbytes/sec
[F 2003 10 14 13:50] 00005 Chris data C:\VLM_Repository\Teachers\MessageBox\messages.txt Sending.
[F 2003 10 14 13:50] 00005 Chris data C:\VLM_Repository\Teachers\MessageBox\messages.txt File sent
successfully. Size: 2477 bytes. 2.419 Kbytes/sec
[L 2003 10 14 13:50] 00005 Chris cntr User from 192.168.0.2 logged out

```

Figure 7.18: Extract from FTP Server log.

Users 1 and 2 were clearly inhibited by previous negative experiences with computers, including the fear of crashing the system if they did something wrong. Interestingly, an analysis of their interviews and log files show that, once they had gained some confidence in the robustness of the Learning Shell, these two users took a very exploratory approach, popping in and out of things almost at will.

For example, when interviewed, User 2 described her experience of “being able to go into the menu like that and surf around and see where everything is...rather than going to Help.”

All participants completed the first two scenarios without any major difficulties. While User 1 and User 3 were confident with their use of mouse and keyboard from the outset, User 2 was noticeably more hesitant.

All participants paused at what was the most challenging task in these opening scenarios (Scenario 2.4): finding the Course Explorer immediately after logging on, and then using it to navigate to a section of the course. User 3 opened Help, as suggested in the scenario, read it through until she saw what to do, and carried on without further delay. User 1 followed the same path, but rather more slowly and cautiously. User 2 was even more hesitant, first trying to figure out how to proceed from what she saw on the screen, without accessing Help.

The data from this observation was of most value when correlated with data on the participants' previous learning and computing experience supplied by the user profile questionnaires, with the transcripts of the interviews, and with additional observations made during the course of the evaluation as a whole. It confirmed that they represented a range of computing experience with User 3 the most confident, and User 2 the least so.

In addition, User 3, the quickest in completing the scenarios, as the most experienced distance learner of the three appeared to have the clearest mental picture of the system and what she was trying to accomplish with it. She also appeared to make the most judicious use of the Help facility.

Interviews

The most valuable form of data collection, from the usability perspective, proved to be the interviews which were undertaken with each user individually at the end of each one-hour session. These were semi-structured interviews (Preece et al., 2002, p.394) in which each volunteer was asked the same set of questions as the starting point for exploring their experience with the prototype (Appendix K3). The interviews averaged between 20 and 30 minutes.

No major usability problems were identified during the interviews. Those problems that did emerge were relatively minor interface design issues and errors missed (or introduced!) in the pilot study, and the odd technical hiccup related to limitations in the test environment. An example of the former was a confusing status message displayed

after users' had downloaded their messages from the server that has already been alluded to. Examples of the latter were using the same machine for each user, and requiring separate dial-up connections to access the "university" (i.e. the farm building several kilometres back in the hills where the Windows 2000 server was installed) and the Internet, when normally they would be accessed through the same connection.

The most substantial proposal to improve usability came from User 1 who favoured a "Back" key or undo facility. There were times, he said when he had made an error in working through a scenario and "I knew that I had to go back but there was no way to go back except exit completely, but I [only] needed to go one step backwards."

An interesting point noted by User 2 was that her mental picture of what she could do in the Learning Shell was constrained by her prior experience of what was and was not possible in Windows. For instance, she initially struggled to understand how you could easily move between different sets of programs (i.e. study modes). "And that's the thing, you can't do that generally on a computer because you lose something or you've got to close it all down completely and put it away before you open something else up, I find. Well, at least, that's how I do things, whereas this time I didn't have to do that. I could to and fro and I think I struggled to adjust to that."

As has already been noted, the most experienced distance student, User 3, completed the scenarios consistently faster than the other two volunteers and her observations on the Learning Shell often exhibited the clearest understanding of its learning features. This supported the conclusion that there was a good match between many of the Shell's features and the real world of extramural study, helping distance students to understand these features intuitively, and improving its usability as a learning environment.

The complete interviews were transcribed and then analysed for significant, common themes relating to the goals of the evaluation. Full transcripts are attached as Appendix K4. Major themes identified were:

- Frustration with the complexity of existing computer systems.
- The problem of poor internet service.
- The effects of personal isolation.
- Support for the Learning Shell.
- The importance of training and help.
- Simplifying searches to avoid information overload

All participants were enthusiastic about the usability of the Learning Shell, contrasting what they accomplished through it with the difficulties they often faced completing tasks in the Windows/Internet Explorer environment.

Complexity of existing systems

One of the most important themes to emerge from the interviews was the frustration of the interviewees when trying to use their home computers for study and other tasks. At times the interviews read like a litany of the problems users face negotiating the complexities of the Windows environment. The interviewees alluded to:

- *Feeling overloaded with unnecessary features and functions.* User 2 observed that computers were “not as simple as they should be”, while User 1 talked of “so many hidden things that you stumble across” in Windows and of still not knowing what many features were for. User 3 noted that “with a normal computer, and other software, there's a lot of stuff that you just don't need. And to get around to doing the task that you want to do, it's harder to get there because you've got obstacles in the way.”
- *Getting lost in a maze of too many open windows.* User 1 summed this up in explaining his resort to “alt+ctrl+del”: “You've got to get out to start again, because you have no idea where you have gone and what you've done and the computer won't tell you what you've done...” User 2 talked of running into problems by “being too busy, trying to go here and do this and do that and not closing the boxes that need to be closed and running things on top of other things and stuff like that.” User 3 thought that people “lose patience, because you have to go different ways around to do different things. You have to close boxes down, you've got to go back to the start when you log off, you've got to find your way back around again.”
- *Frustration at not being able to complete tasks.* User 3: “Half the problem with doing a lot of things on a computer is that a lot of it is just irrelevant stuff, and you get sick of waiting just to get to your original stuff. I think that is where a lot of people think, I can't be bothered doing that anymore.” User 2 spoke of trying to do something using her computer and just giving up: “I would end up turning it off.”
- *Not getting the help they needed from the help system.* User 3 explained how she was unable to find the appropriate help to complete a task in a spreadsheet: “And I knew that I could do it, that it was possible to do it, but to actually find out how to do it was really hard. Especially when the help is [supposed to be] designed to do that.” In her experience, help systems often presented her with “irrelevant drivel”.

User 1 found that accessing the help system on his home computer didn't solve his problem because "quite often it's not the scenario where you have got yourself... you have done something different."

- *Being overwhelmed by too much information.* The interviewees found that general-purpose tools like help systems and search engines presented too much information to the user, which they found difficult to narrow down for their specific purpose. "Especially when you search the Internet, you type in say 5 key words or 3 key words and then it comes up with 250,000 or a million possibilities and you think 'Gee. I 'm not going to go through all that'"(User 3). "I get 300 sites of which 275 of them are useless. It's a lot slower on the internet the other way than your way, because I have to physically open it and have a look at it to see if its relevant or not, and then go on to the next one and it's a long process" (User 1).

These experiences offer strong support for one of the major premises on which the learning computer concept is based viz. existing computer systems are too complex for effective distance learning.

Poor Internet service

Another theme to emerge related to the quality of rural Internet connections. The usability problems the interviewees faced in the general-purpose Windows environment were compounded by extremely poor Internet service. This was especially the case for User 2 and User 3 who lived back on hill country farms. User 3 spoke of "waiting, waiting" for items to download. User 2 told of waiting half an hour for two emails and being told to wait 31 hours for a web page: "I have seen it at 1 [kilobit per second]. I know that it should be 26,000 or 24,000 and it is 1, literally just one. You can get a connection. You can dialup and get online. But you can't get anything. Not even a home page. And you're paying your \$2.50 per hour for nothing!"

Any web-based learning system relying upon downloading learning materials from a central server simply cannot work under the circumstances the interviewees described.

Isolation

These usability and accessibility problems are especially challenging when it is combined with the pressures of study in isolation from one's peers.

User 3, with more than 10 year's computer experience, reported how she would be unable to figure out how to complete assignment tasks on the computer by herself. "It

was really hard to find out what you were supposed to be doing. So I just leave it." It was easier to shut down the computer and complete the assignment on paper.

User 2 spoke of her frustrations trying to obtain information for her course off the Internet because "you've got to be a brain box to use it." For her it was sometimes easier to drive the 70 kilometres into town. "It's easier to do the basic stuff, do the hard yards, like go to the library and things like that and photocopy pages out of a library book... Because its just easier. But it should be easier on a computer but I find that it is not."

In an on-campus environment, the problems they confronted – how to copy text from one document to another, or how to format an Excel chart correctly – would be quickly sorted out with the help of more experienced peers. But in their isolated situation, these problems rendered their computers unusable.

The experiences described by the volunteers strongly supports the premise advanced in this thesis that the isolation of distance students compounds the usability problems of general-purpose computing environments. Standalone systems may overcome the delivery problems of web-based courseware. But they exacerbate the isolation problem.

Usability of Learning Shell

The dominant theme to emerge from the interviews in relation to the Learning Shell itself was an overwhelming endorsement of its usability. In contrast to the general frustration the interviewees expressed with their normal computing environment, they all embraced the Learning Shell enthusiastically and indicated they would be keen to use it for their own studies.

User 1 thought the Shell was "user-friendly", and that it would be especially useful for a new user who had purchased a computer to study online and "this is the first time they have had to come to grips with a computer as well as their new course. It's not intimidating at all." They would quickly grasp that they couldn't do any damage by experimenting.

For User 2, "If I was to be studying at Massey and there was a program like that, that would probably be one of the main reasons why I would study online."

User 3, the most experienced computer user and distance student, emphasised: "Well I could see that I could use it for [my study] quite easily. I think. God. It would just make it a lot easier."

The Learning Shell is distinguished from a general-purpose window and web environment by its minimalist, integrated interface that manages many housekeeping tasks normally left to the user, and limits what the user can do themselves in order to free the user for learning. The volunteers commented favourably on this approach in the course of their interviews.

User 3, for example, thought that compared to her own computer "this is a lot easier because you just close everything at the same time, you don't have to worry about closing boxes here and there, and when you log back on, it comes straight back exactly to where you started from. It was a lot easier."

User 1 liked it that there were "limited...buttons, shall we say, that you can use" for navigating the system.

User 2 observed: "And even when I went ... where I shouldn't have been it was very easy for me to get out and start again. I just found it so easy, whereas normally, if I was on my computer and something like that happened, I would end up turning it off...and going back to it another day because I'd get frustrated."

Some of the favourable characteristics of the Learning Shell the volunteers singled out were: simplicity, ease of use, speed, relevance, robustness, friendliness, transparency, and consistency.

- *Simplicity.* The interviewees liked the way the Shell simplified tasks they often struggled with in the Windows and Web environments, such as managing transitions between different screens and programs. "If I wanted to [email the tutor with a question on my course], I had to close down, hook up to the Internet, bring up all that. It would take a long time, and that's when I think, I can't be bothered doing all that. I'll just struggle through. Whereas with this one it was right there, and you could just easily do it" (User 3). For User 2, "Half the time I get frustrated when I have to work with my computer because it is not as simple as it should be. And this is as simple as it should be." User 3 noted that one of the advantages of this simplicity is that "you don't have to remember a lot of things."

User 1 liked the way that complexity was hidden from the user. He would never have "gathered that it was a complex computer program." Otherwise he "would have panicked straightaway." He also liked the fact that only three basic operations had to be memorised to use the entire system: "And that it's very easy to find your way around the whole thing with those three keys. To me that's the most appealing thing about it."

- *Ease of use.* User 2 said that she had completed many tasks that she would have given up on at home as too hard: "But I find it really odd that I'm doing this and it's so easy." User 3 linked this ease of use to the simplicity of the environment as "you don't have to worry about all the bells and whistles and everything else going on because it is all just there".
- *Speed.* "It's quick...I'm just thinking of computer programs in general, basically. Some, when you are jumping from one field to another...Oh I have got to shut that down, load that up. And not with changing disks, but just in general. But here, going from that one to the other was very quick to get from here to there. It was a matter of within 5 seconds I was gone from one to the other. There's no sitting and 'please wait' while it does this and that..." (User 1).
- *Relevance.* User 1 and User 3 particularly contrasted the relevance of the information they received from the help and search facilities associated with the Shell: "At least here the content is relevant to what you are doing" (User 3).
- *Robustness.* In User 1's view a new user "within half an hour will have the grasp that they can't do any damage" and that "if you get yourself in trouble, it's very easy to get yourself out of trouble." User 3 "felt confident enough to whiz around, because you've got all the help things". User 2 "knew that I could go into that and I could go wherever I wanted and I am not going to get stuck anywhere."
- *Friendliness.* User 1 felt "that it is really user-friendly compared to my computer" and "not intimidating at all".
- *Transparency.* User 1 noted that the interface is "very transparent... it's all very clear what you use to do what you want it to do. There's no hidden features."
- *Consistency.* "It's pretty much... standardised. Because you are using the same three keys for every area. So you go into a new area and these three keys are applicable. Whereas if you go into different things on say your normal computer, well sometimes this is for that and that is for that, and that and that..."[User 1]

Training and Help

The interviews also highlighted the importance of the initial training and the help system to the overall usability of the Learning Shell.

An important element in the success of the usability experiment was in the training documentation that accompanied the software: the demonstration (video), the Handy Hints sheet, and the set of training exercises (scenarios). Each of the participants

emphasised the importance of these, with the least experienced user ("I would be your best guinea pig ever!"), User 2, being most enthusiastic. She repeatedly stressed the importance of the clear instructions making it possible for her to complete tasks in the Shell beyond what she had thought herself capable of: "But obviously it's not over my head ... if someone gives you instruction and its easy to follow you can pretty much do anything, can't you? You don't have to be extremely intelligent to be able to do things like this."

The evaluation also showed the effectiveness of linking Help to each component in the system and of emphasising a "Handy Hints" approach. This provided a simple mechanism for delivering just-in-time, just-enough, context-sensitive help to the user. User 1 and User 3 emphasised that the way the help system was organised, it was always relevant and easy to find out what they needed to use a feature of the Shell: For User 1 "if I click on Help I have found every time what I need to do and its been able to solve it for me." User 3 noted, "the content is relevant to what you are doing".

For User 2, completing the scenarios on the Learning Shell was her first ever instruction in basic computing skills. Prior to the experiment she had "never, ever done" tasks like copying and pasting which are the ABCs of computing. From this perspective, the Learning Shell is a stripped-down version of general-purpose GUI like Windows that implements a subset of its functionality. By training users in basic skills that also apply in the general-purpose environment, such as copying and pasting between applications, using the Shell has the side effect of introducing users to general-purpose GUIs and providing a bridge to their more complex environment.

Many of the improvements to the Shell's training and help documentation, such as the "handy hints" emphasis, were introduced as a result of the pilot study. They had a big impact on the success of the usability testing, which underscores the value of including a pilot study in the overall evaluation process.

Simplified search mechanism

Another theme emphasised by the users was the effective support the Shell provided for finding additional learning material through Extramural Support.

The extramural learning support system is one of the more complex aspects of the Learning Shell. It is essentially a database of useful definitions, book references and web sites, and a mechanism for searching that database to provide the student with material relevant to what they are currently studying. Interacting with this support

system was one of the more challenging tasks the volunteers were required to do. All were able to complete these tasks.

While they could not judge the content of the search results, their comments were nevertheless valuable in highlighting some of the strengths of the learning computer approach, compared to the search mechanisms they normally would be faced with.

For User 3, "It was easy and it was fast. And that is my two main things at home... Because you didn't have to waste all that time in sorting through the information that you are given as to what's relevant and what's not."

User 1 also found it easier because "obviously it's trimming it down to what's relevant and what's not. So I presume for web sites that I have been given there, that they will be useful for what I want, not may be useful, will be useful."

In future the completed system will also need to be used by students who are actually enrolled on a paper whose content is included within the shell. This will provide a more meaningful test of the effectiveness of the help and search facilities. It will also allow the issue of student learning to be addressed.

7.5 Conclusion

An evaluation of IMMEDIATE was carried out under realistic field conditions in a remote farming and fishing community with the software running over the telephone network. It was a major undertaking setting up an evaluation of this kind in a rural area and finding suitable participants. Whilst there were only three users in the in-depth evaluation, they were representative of distance learners in remote environments.

The number of users was sufficient to demonstrate the practicability of the concept. It showed that the Repository Manager could handle communication between the university and student end, that the Course Authoring and Management system could assist a remote student, and that the system was able to operate over a two day period without any problem. The system proved fully functional and accessible to users, providing speedy access in contrast to the long delays that occur when using the Internet in similar circumstances.

Whilst the users took varying lengths of time to accomplish the tasks, all were able to complete them even though the course material was unfamiliar. In their comments the participants all acknowledged the benefits of a system of this kind. They were able to run this distributed system over a telephone network and demonstrate that the main functions could all be used.

The user testing demonstrated that the learning computer approach was feasible and offered accessibility and usability advantages. By extension it can be presumed that it also offers learning advantages through the range of functionality it offers, the minimisation of distractions and ease of use offered by the specialised interface, and the speed of access to learning material. However, this was not able to be directly tested in the course of the evaluation.

Something highlighted in this study was the depth of frustration felt by isolated computer users at being often unable to complete what should be quite simple tasks. In contrast, the three volunteers who took part in the in-depth usability evaluation were confident that in the two one-hour sessions, they had gained enough experience and confidence to be able to use IMMEDIATE effectively for their own study. Indeed the overwhelming level of enthusiasm shown by the participants had not been anticipated.

The experiences and impressions of the volunteers captured in the interviews strongly support the key thesis of the research concerning the advantages of organising a distance learning environment as a special purpose computer, tailoring the interface to the task, and managing transitions between tasks, so as to minimise the demands upon the user.

The consensus of the volunteer users in favour of the Learning Shell, summed up in User 2's comment: "If I could do my study using this type of program it would be so easy" is the most compelling argument in favour of the learning computer approach as realised by IMMEDIATE.

Chapter 8

Conclusion

In this chapter the research that has been conducted is reviewed in light of the initial objectives that were set. What has been achieved vis-à-vis these objectives? What has been learnt in the process? And what future work needs to be done to further advance these objectives?

8.1 Summary of research

The starting point for this research was anecdotal evidence and personal observation that existing web-based e-learning systems are too difficult to use, and require too much sophisticated technology, to work in all distance learning situations, and too limited in their functionality to offer a significant learning advantage over correspondence-based courses, especially when the latter are combined with course web-sites and email. This would be particularly true for rural communities in the developed and developing countries for whom, paradoxically, correspondence schooling was originally developed.

Furthermore, it was observed that less experienced users frequently struggle to complete even basic computer tasks in an unsupervised environment, and it was surmised that this could be a serious impediment to computer-based distance learning.

Preliminary research, including informal meetings with experts and trying out some educational software, supported these concerns. This suggested that e-learning – rather than offering new opportunities to narrow the gap between the world's haves and have-nots – may perpetuate and even widen educational inequalities, through introducing a two-tier system based on access to, and ability to use, sophisticated computer and communication technology.

These observations initiated a search for ways in which computer technology can be applied to distance learning so as to offer a learning advantage over correspondence-based courses, without incurring a disadvantage in terms of accessibility. The objectives for this research were:

- To identify the ways in which computer systems can assist distance learning and to what extent this potential has been realised in existing e-learning technologies.
- To identify threads of current research relevant to overcoming the shortcomings of e-learning systems, drawing from the broad spectrum of Computer Science including Artificial Intelligence, Software Engineering and Human Computer Interaction,
- To develop new ideas, and extend existing ones, for fulfilling the potential for computers to effectively support distance learning.
- To build and test a prototype which implements these key ideas.

The research conducted to meet these objectives can be broken into three main phases viz. investigating the problem through a literature review, conceptualising a solution, and testing the conceptualisation through prototyping and evaluation with users. A formal iterative research methodology was used, utilising qualitative and quantitative methods.

8.1.1 Literature review

The first two research objectives have been met through an extensive literature review, which included conference papers, journal articles, books, web sites, industry publications and news reports. This began with a systematic review of the origins and evolution of distance education and of computer-based learning research. From this review, it became clear that “distance learning” covered a broad range of scenarios – from specialised in-house industrial training to television-based, mass-oriented cultural improvement campaigns - with very different requirements. For the purposes of this research, it was necessary to focus upon computer-based delivery of extramural courses – university-run programmes for students unable to attend normal classes – designed for individual home study. Nine contributions that networked computers can make to improve extramural learning were identified, ranging from facilitating communication among students to providing individualised tutoring help when a teacher is not available.

In the next step, the rich record of contemporary computer-based, and especially web-based, learning research was reviewed. From this review it was concluded that the potential for computers to improve the extramural study environment is far from being realised. In fact, very little research was found that was explicitly directed towards the

specific problems of this sector. Research tended to be weighted more towards the needs of the education provider than towards those of the distance student.

Much of the literature is critical of the limited functionality of commercial web-based courseware and learning management systems; a smaller, but significant body of research addresses usability problems; and a very small section addresses accessibility problems. But most research is centred upon innovations to web-based systems, which incorporate more sophisticated teaching or course management methods, which are more suited to training scenarios than higher education, or which assume the hi-tech urban communications environment within which only a small minority of the world's population resides. From the student's standpoint, they tended to assume levels of computer literacy and access to high-end multi-media and broadband technology that was more in line with what can be expected for on-campus courses, than for individual students studying from home.

8.1.2 Conceptualisation and specification of a computer for learning

From the review of current research, eight guidelines were synthesised for developing effective extramural e-learning environments. In addition, three broad user-centred strategies were identified as showing promise as possible ways to implement such an environment. These strategies emphasise localised over centralised functionality, specialised over general-purpose tools, and user-initiated adaptability over system-initiated adaptivity. It was hypothesised that by following the design guidelines and combining these three strategies – without making any presumptions about technological platform – a workable way could be found to meet all the requirements for an extramural e-learning environment that offers a significant improvement over correspondence-based courses.

In order to meet the third objective, the next step was to conceptualise a learning system that could test out this approach. The idea emerged of a learning computer – which integrates all the features a student needs for learning into a simplified, specialised, individualised, network user interface. In visualising the learning computer, analogies were drawn with many everyday objects and activities. The central metaphor, however, was that of the multi-dimensional university community of learning, which the learning computer extends to the extramural student. Within the learning computer, each learning dimension or study mode is defined by a set of elements, each of which corresponds to a basic constituent of university study.

Having visualised the new system in this way, it was then necessary to specify a prototype in more detail that was in line with the design strategies and guidelines of the underlying hypothesis. In particular it meant specifying the technological platform on which it would be built, and the services that would need to be employed at the university end to support it. The key decisions made at this stage were:

- That the learning computer would be implemented as the client in a client/server network, in which most functionality is distributed to the client side. Communications and learning content would be updated by periodically linking to a central repository of messages and learning resources via the Internet and, where necessary, via alternative media such as CDs delivered through the post; rather than as a browser client downloading documents from a web server.
- That the student interface would be implemented on the Windows platform as a specialised direct manipulation graphical Learning Shell replacing the operating system GUI; rather than using courseware's approach of adding functionality to a general-purpose web browser. This Shell would be assembled on a modular basis, in which each learning element or other interface object is implemented as a reusable software component, using the Delphi rapid application development environment.
- That user interactions with the Shell, including navigation, would be simplified by managing them through an intermediate system layer between the interface and the operating system, which references a tree data type that models the modular, hierarchical structure of an extramural course. This layer would also model the student to provide the reference point for individualising the Shell to a specific user.
- That learning assistance would be provided to the student through integrating communications, collaboration and support, by means of a replicated SQL-compliant database which is maintained by the course tutor, and can be queried by the student via a user-friendly interface. In this way, the student can obtain assistance through discussion with other students, asking the tutor, or querying the support database; and the outcomes can be logged, edited and incorporated into the database, as a means of dynamically improving the support system.

To implement these decisions and prove the overall concept, a course authoring and management application and a communications management component needed to be built and integrated with the learning computer into a network. This overall prototype

has been called IMMEDIATE (Integrating MultiMEdia in a DIstAnce learning and TEaching environment).

8.1.3 Evaluation through prototyping and user testing

Incremental prototyping was used to evaluate and refine the main elements of the design specification and then to integrate them into the operational IMMEDIATE network. This prototyping confirmed that the method proposed for developing a computer-based learning environment was workable. In its simplest form, IMMEDIATE provides an integrated, easy-to-use, structured environment for the extramural student to navigate his/her study guide, communicate with others electronically, and access a database of helpful definitions, references and URLs. At the same time it provides a framework in which the full functionality for e-learning – i.e. the nine ways in which networked computer systems can support extramural study – can be integrated as appropriate. IMMEDIATE provides a dynamic authoring and management system which enables a course to be updated, improved and re-used after it has been deployed.

The second step in the evaluation and testing process was to get the IMMEDIATE network up and running, first over a LAN and then over a rural telephone-based communication system, so as to test it with users. It was especially important to evaluate IMMEDIATE under as realistic conditions as possible – including lower-end technology, unreliable communications infrastructure, and users representative of the more inexperienced and geographically-isolated among extramural students. To achieve this, the prototype was installed and tested over the rural telephone network with volunteer users in a small New Zealand farming and fishing community. The evaluation here was for functional completeness, accessibility and usability, using scenario-based user testing. This evaluation was prepared by a pilot study.

IMMEDIATE performed very favourably under these conditions. The volunteers' response to the learning computer was enthusiastic, contrasting what they could accomplish with it to the difficulties they faced with conventional systems. It was concluded that the user testing gave strong support to the thesis that distributive, specialised and adaptable strategies can be successfully combined to provide a learning environment that is more widely-accessible and usable than web-based courseware, and that offers a learning advantage for extramural students over correspondence and email-based courses.

8.2 Contribution to knowledge

The central contribution to knowledge of this research has been the development and demonstration of the *learning computer* as a new concept distinct from either a LMS or a LCMS in several important respects.

First, the learning computer does what the web-based LMS's and LCMS's have so far been unable to do viz. deliver distance education on an anybody, anywhere, anytime basis. It demonstrates a method by which it is possible to combine the benefits and power of standalone computing with the collaborative potential of the network, in an easy-to-use extramural study environment. This approach is universally-applicable, and is especially effective in supporting the individual distance student in circumstances where an LMS fails, and the only realistic option has been correspondence study.

Anybody. The learning computer combines the technological advances represented by networked PCs with the simplicity and ease of use of the earlier dedicated computer systems like TICCIT or PLATO, providing a dedicated learning environment. The navigation, file management, networking and other housekeeping functions the user would have to perform themselves in a system built upon a general-purpose environment are embedded into the learning computer itself, requiring minimal input from the student.

Anywhere. By distributing most content and functionality to the client side, the IMMEDIATE architecture enables the learning computer to be used almost anywhere, and minimises the student's vulnerability to slow or unreliable Internet connections. Because it only requires periodic background updating and synchronisation when and where a network connection is available, it is able to utilise the speed and functionality of standalone systems without sacrificing the communication and collaborative strengths of the internetworked PC.

Anytime: All the collaborative and communication functionality of the learning computer, including group work and learning support, is implemented using an asynchronous model and a replicated database that can be accessed and used anytime, regardless of whether a network connection is currently available. Students can access their course content at anytime.

The second and most fundamental difference from a LMS or LCMS is that the distance students themselves, rather than the education providers, are the primary target users. It demonstrates the importance of designing e-learning systems with student

requirements paramount, and shows how this can be achieved. The learning computer is first and foremost a specialised learning environment, in which network and education provider features are designed as support services rather than as authoring, teaching or course administration productivity tools. An integrated, holistic approach is taken to e-learning, in which the usability of the student's overall computing environment is considered as important to successful learning outcomes as the learning content itself.

The learning computer embodies a set of guidelines for designing course-oriented educational software as a student-centred learning system. These guidelines have been successfully followed in the implementation of the IMMEDIATE prototype. It demonstrates the potential for specialised environments to address usability concerns resulting from the increasing complexity of general purpose environments like the PC as they encompass more and more functions and features.

The third unique facet of the learning computer is its support for the trend towards *ubiquitous computing*, i.e. invisibly integrating computer power into everyday life through embedding dedicated computing devices into common appliances such as washing machines, automobiles or watches. The idea of specialised embedded computers to simplify and improve learning as found in Bork (2001a) and Hoppe et al. (2000), focuses upon the use of specialised *hardware*. The IMMEDIATE prototype furnishes additional support to this approach. At the same time, it contributes a method for implementing the approach with specialised *software*, as a graphical user shell over the most widely-used family of operating systems, Microsoft Windows. The Learning Shell offers a method for implementing a “*what*” interface centred upon the task in the user's domain rather than a “*how*” interface focussed upon the mechanism for achieving that task. This addresses a long-recognised challenge for simplifying and improving the usability of computer systems especially in industrial applications, as discussed in Gentner et al. (1990).

This research has also furnished additional support for other conclusions drawn by researchers relating to the efficacy of applying the strategies of specialisation, localisation and adaptability to the development of e-learning systems. In particular:

- *Interface design*: This research has underscored the design principle that simpler is often better. IMMEDIATE was conceptualised from the requirements for an extramural e-learning environment, without making any presumptions about technology. This opened the door to considering alternatives outside the prevailing web-based framework for e-learning, and revisiting some basic elements of user-

oriented interface design that are at the core of the learning computer conception. By combining an emphasis on interface design – especially user-centred adaptable and collaborative strategies – with an emphasis on e-learning as an extension of the human teacher and the university, the framework for an integrated, individualised, multi-dimensional, easy-to-use learning environment has been laid. This supports the findings of Murray et al. (2000) that “good interface design and passive but powerful user features can sometimes provide the benefits that are ascribed to more sophisticated or intelligent features” (para. 41).

- *Distributed functionality*: Learning Management Systems conventionally require the student to maintain a live connection to a central web server, which dynamically generates learning material and then uploads it as web pages to the student's browser application. Supporting a student without a live link is complex because key functionality resides on the server. TILE (Gehne et al., 2001) addresses this by duplicating some web server functions on the student's computer so that some data can be updated via CDs, and the student can work offline when a connection to the server is unavailable. IMMEDIATE extends this approach by relocating all learning functionality to the student machine so that only periodic updates and synchronisation with the server are needed, either by the Internet or CD.

IMMEDIATE is innovative in using the Internet as a delivery mechanism rather than as an integral part of student's interactions with their learning material. Web-based material can be accessed and displayed alongside any other material, and a messaging system similar to email is integrated into the Learning Shell, but no key functionality relies upon maintaining a live connection to the Internet. This implementation enables the Learning Shell to utilise the fuller functionality of a standalone computer while incorporating the communication and collaborative learning capabilities of web-based systems.

The IMMEDIATE network functionality is achieved using an FTP Server, and custom message-queuing and database synchronisation protocols, which require only the exchange of compact text files. Because there is no direct connection between the databases on the client and the server, they need only be compatible at the logical rather than the physical level. Large multimedia files can be sent via alternative media such as satellite or post if necessary. IMMEDIATE's architecture provides a simpler and more flexible framework than web-based learning management systems, and can accommodate almost any kind of learning material,

while giving especially strong support to student-centred discovery and collaborative learning methods.

- *Form vs. content*: Some educational software designers such as Smulders (2003) have stressed the importance of designing for *form* in order to make learning the *content* possible. Much current research into improving e-learning systems focuses upon ways of modelling the content viz. the knowledge a course is trying to teach, how it is going to teach it, and what the student understands of it. The success and simplicity of IMMEDIATE is based upon modelling *form* (structure) in the first instance, and then modelling *content* only as necessary within that framework. IMMEDIATE introduces the idea of reusable learning components as an extension of Delphi's drag and drop visual components.

Learning components differ from software components (controls) in that they encapsulate a single domain task rather than a single software task. They differ from reusable learning objects in that they implement the *form* rather than the *content* of a learning module. The learning components are the centrepiece of a modular construction which supports reuse on three levels – the student (multiple courses), the teacher (authoring, updating and re-using learning materials), and the programmer (reusable code).

- *Knowledge-base*: Typically, extramural university courses are highly modular, broken down into units and sub-units with learning goals and assessments attached to each. By using the *form* of the course to define a reference model for the Learning Shell, a powerful mechanism is created for navigation, integration, synchronisation and individualisation of the Shell. Many courseware systems have a facility for linking student discussions to a particular topic in a course. With IMMEDIATE, students can do more than view and contribute to discussions linked with a particular topic. The reference model underpins a dynamic, context-sensitive and adaptable system for helping the student to learn the course content, by providing a framework for organising a database of discussions, definitions, references and URLs, into a coarse-grained knowledge-base.

The student can query and search this knowledge-base through a user-friendly interface, and discuss the results with others. Through the Learning Shell tracking these discussions and recording student satisfaction with the system's responses, and its integration with the course authoring and management application, IMMEDIATE supports dynamic updating of the knowledge-base by the course tutor

in light of those responses. This contribution builds on research initially carried out within the TILE group (Heinrich et al., 2000; Heinrich et al., 2001).

8.3 Future Work

Developing the IMMEDIATE prototype was a large and complex engineering task involving specifying the system, writing thousands of lines of source code, locating and customising reusable software components, installing and configuring the elements of a client/server network, installing modules of an actual extramural course, and evaluating the system with users. During the course of this work it was necessary to keep clearly in mind that the principal objective of the project, despite its intensely practical side, was to evaluate a hypothesis, rather than the development of a practical educational tool.

The conclusion of this research is that the prototyping and user testing of IMMEDIATE offers strong support for the underlying hypothesis, and that this approach to providing an extramural e-learning environment merits further exploration and refinement. Because of the pervasively-modular structure of IMMEDIATE, the “nuts and bolts” of the current prototype provide the basis for further research, without precluding a systematic revision of all existing elements and the incorporation of new ones.

Three areas of IMMEDIATE for further research and refinement are:

- *Learning Shell Interface:* The heart of IMMEDIATE is the student interface represented by the Learning Shell. During the successful evaluation phase, some areas were noted where the interface could be modified to improve usability, such as a greater ability to step back from (undo) a particular course of action, and making the learning support facility accessible from any interface component in a similar manner to the help facility. As part of developing an improved version of IMMEDIATE for further learning trials with students, a thorough review of the interface should be carried out, including drawing upon independent usability experts.
- *Individualisation:* Evidence from the IMMEDIATE prototype supports the view that a special-purpose educational computer makes it easier to realise many of the individualisation goals of computer-based learning researchers. In line with the Learning Shell's user-centred approach, individualisation is mainly achieved through visual cues to guide students to adapt the system to their individual needs. Most scope for increased individualisation relates to the learning help (Extramural

Support) system. This includes improving the mechanism for natural-language querying of the knowledge base, mapping text phrases to key learning concepts, and integrating more sophisticated algorithms and strategies for searching the knowledge base, including suggesting where help is needed based on the student's personal learning history.

The learning help system is one of the more innovative aspects of the Learning Shell. It needs to be tested in further trials by students who are actually enrolled in the paper whose content is included within the Shell. This will provide a more meaningful test of the learning effectiveness of the student support and search facilities, and what advantage if any would be offered by adding these more sophisticated algorithms and tools.

- *Course authoring and management tool:* This part of IMMEDIATE took full advantage of the capabilities of a Rapid Application Development tool like Delphi to quickly assemble an application for the purposes of prototyping. The focus was on working out mechanisms by which a non-programming teacher could construct and manage a course, wherever possible re-using software modules from the Learning Shell. The tool has not been tested with users beyond utilising it for the authoring and updating tasks associated with the evaluation phase. A future task is to evaluate and refine the authoring application for functionality and usability, through testing it with non-programmer teachers.

New directions

Since this research project began in 2001, new technologies have emerged or gained greater prominence, as researchers have sought to improve the effectiveness of educational software or raise teachers' productivity. For instance, the application of XML to learning and the development of reusable learning objects based on XML are receiving increased attention.

The component-based architecture of the Learning Shell makes the incorporation of new learning technologies straightforward. As part of reviewing and refining the Shell's learning components for pedagogical effectiveness and usability, incorporating XML-based techniques and modules should be explored.

IMMEDIATE implements the learning computer concept for the Windows platform. This involves hiding the Windows GUI from the learner, through disabling some features, and through the programmer following certain constraints in developing new learning components. The next step is to customise the Learning Shell so that it can be installed

on and run from a portable storage device as a *mobile* learning computer. The increasingly-affordable USB-based “memory stick” portable storage devices – through their compact size, large storage capacity, speed and aesthetic look – offer promising possibilities for implementing this approach.

A Linux-based implementation of a learning computer for the PC would carry a greater programming overhead while offering a cleaner approach, because the learning shell can be built as a GUI directly on top of the Linux kernel. Future work will investigate the feasibility of implementing a Linux-based learning computer along two lines:

- As a single-use learning computer. Because the learning computer works with lower end information technology, this has practical potential for re-using older PCs that might otherwise be dumped.
- As a self-booting portable device which can be plugged into any PC to temporarily convert it into a specialised learning computer. This offers a way for a learner to take full advantages of the learning computer approach, while sharing a general-purpose PC with other household members and other tasks. Kawato et al. (2003) have researched running Linux-based learning systems in a Windows environment, by launching them from a bootable, auto-configuring CD-ROM, based on the Knoppix system¹. Black Dog is a compact mobile Linux-based computing device that attaches via USB to Linux, Windows, or Mac computers, commandeering the host's keyboard/video/mouse/modem functions to run it's own applications (LinuxDevices, 2005). Black Dog offers a promising technological basis for implementing the mobile learning computer concept.

IMMEDIATE's approach to improving the usability and accessibility of a complex software system has application to other domains, which would benefit from specialised, easy-to-use, task-oriented interfaces, e.g. computerised systems for operating industrial machinery, or integrated farm management systems. This will also be explored further in the future.

¹ <http://www.knopper.net/knoppix/index-en.html>

References

- Alessi, S. M. and Trollip, S. R. (1991): *Computer-Based Instruction: Methods and Development*. Prentice Hall. Englewood Cliffs. Second Edition.
- Allen, S. (2003): Massey amasses record student numbers. *Dominion Post*. December 5. Wellington.
- American Heritage (1992): *The American Heritage Dictionary of the English Language*, Third Edition, Houghton Mifflin Company. Electronic version included in Microsoft Bookshelf, 1996-1997 edition.
- Anastaides, P.S. (2003): The Future of the Book, the Book of the Future. *The 3rd IEEE International Conference on Advanced Learning Technologies*. Athens, Greece pp.246-247. IEE Computer Society.
- Anderson, J. R., Corbett, A. T., Koedinger, K., and Pelletier, R. (1995): Cognitive tutors: Lessons learned. *The Journal of Learning Sciences*, **4**, 167-207.
- Aniebonam, M. (2000): Effective Distance Learning Methods as a Curriculum Delivery Tool in Diverse University Environments. *Communications of the Association for Information Systems*. **Vol. 4(8)**. October. United States.
- Arora, V., Suphasindhu, N., Baras, J. S., Dillon, D. (1996): "Effective Extensions of Internet in Hybrid Satellite-Terrestrial Networks". *AIP Conference Proceedings, March 1, 1996*. **Vol. 361(1)**. pp. 339-344.
- Bennet, S., McRobb, S., and Farmer, R. (1999): *Object-Oriented Systems Analysis and Design using UML*. McGraw-Hill. London.
- Benyon, D. R. and Murray, D. M. (1993): Adaptive Systems: From Intelligent Tutoring To Autonomous Agents. *Knowledge-based Systems*, **6 (3)**. Butterworth-Heinemann Ltd.
- Bergstedt, S., Wiegrefe, S., Wittmann, J., and Moller, D. (2003): Content Management Systems and e-Learning Systems – A Symbiosis? *The 3rd IEEE International Conference on Advanced Learning Technologies*. Athens, Greece pp.155-159. IEEE Computer Society.
- Blyth D. (2001): We're waiting to be taken to the next Web level. *NZ Infotech*. **511**. Oct. 15.

- Boada, A., Cervera, A., Prieto, J. (2003): SMS Technology as an Academic Communication Tool. A Case Study: The Open University of Catalunya (UOC). *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5. pp.527-31.
- Bonelli, S. (ed.). (1998): *Internet Complete*. SYBEX. San Francisco.
- Bork, A. (1996): Rebuilding Universities with Highly Interactive Multimedia Curriculum. *International Journal of Engineering Education*. **Vol. 12**, pp. 320-332.
- Bork, A. (2001a). Tutorial Learning for the New Century. *Journal of Science Education and Technology*. **10(1)**, 57-71.
- Bork, A. (2001b): What is Needed for Effective Learning on the Internet. *Educational Technology and Society. Special Issue on Curriculum, Instruction, Learning, and the Internet*. **Vol. 4, No 3**, pp. 139 – 144.
- Bork, A., Ibrahim, B., Levrat, B., Milne, A. and Yoshii, R. (1992): The Irvine-Geneva course development system, education and society. In Aiken, R. (Ed.), *Information Processing 92*, **Vol. II**, Elsevier Science Publishers, New York. Available at <http://www.ics.uci.edu/~bork> as at 12/12/04.
- Bork, A., Ibrahim, B., Levrat, B., Milne, A., and Yoshi, R. (1992): The Irvine-Geneva Course Development System. Aiken, R. (ed). *Education and Society*. Information Processing, **2**. Elsevier. Netherlands.
- Borland (2004): *Introducing Borland Delphi 8 for the Microsoft .NET Framework*. A Borland White Paper. http://www.borland.com/products/white_papers/pdf/delphi_for_dotnet.pdf as at 5/1/05
- BPTA (2001). British Polymer Training organisation web site. <http://www.bpta.co.uk> as at 10/05/01.
- Brandon Hall (2005): LMSs and LCMSs Demystified. http://www.brandonhall.com/public/resources/lms_lcms/lms_lcms.htm as at 21/9/05.
- Brooks, C., Cooke, J. and Vassileva, J. (2003): Versioning of Learning Objects. *The 3rd IEEE International Conference on Advanced Learning Technologies*. Athens, Greece. pp. 296-297. IEEE Computer Society.
- Brookshear, J. G. (2003): *Computer Science: an Overview*. Seventh Edition. Benjamin-Cummings Publishing. California.

- Brusilovsky, P. (1999): Adaptive and Intelligent Technologies for Web-based Education. *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching*. (ed. C. Rollinger and C. Peylo). **4**, 19-25.
- Brusilovsky, P. and Miller, P. (2001): Course Delivery Systems for the Virtual University. *Access to Knowledge: New Information Technologies and the Emergence of the Virtual University* (ed. T. Tschang and T. Della Senta). Amsterdam: Elsevier Science.
- Brusilovsky, P., Eklund, J., and 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, 14-18 April 30 (1-7)*, pp. 291-300.
- Brusilovsky, P., Schwarz, E., and Weber, G. (1997): Electronic Textbooks on the World-Wide Web: From Static text to Interactivity and Adaptivity. *Web-Based Instruction* (ed. Khan B). Educational Technology Publications. Englewood Cliffs.
- Callear, D. (1999): Intelligent Tutoring Environments as Teacher Substitutes: Use and Feasibility. *Educational Technology*. **39(5)**. September-October. New Jersey.
- Cantu, M. (1999): *Mastering Delphi*. Sybex. San Francisco.
- Carbonnel, J. R. (1970): AI in CAI: An Artificial Intelligence Approach to Computer-Assisted Instruction. *IEEE Transactions on Man-Machine Systems*, **MMS-11(4)**, December.
- Carroll, J. M. (2000): Making Use: Scenarios and scenario-based design. *Interfacing Reality in the New Millenium. OZCHI 2000 Conference Proceedings*. CHISIG. Sydney. December 4-8. pp. 38-48.
- CDLP (2004): "What is Distance Learning? *California Distance Learning Project*. <http://www.cdlponline.org/index.cfm?fuseaction=whatis&pg=4> as at 8/5/04.
- Cerri, S.A., Gouarderes, G., and Paraguacu, F. (eds.) (2002): *Intelligent Tutoring Systems. 6th International Conference, ITS 2002*. Proceedings. Biarritz, France and San Sebastian, Spain. June.
- Chan, C. W. (2003): Virtual Understanding from Virtual Laboratory. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5. pp. 1017-18.
- Chaudhary, S.S. (1992): Television in Distance Education: The Indian Scene. *Indian Journal of Open Learning* **1(1)**: pp. 23-31.

- Chee, Y.S. (2001): Networked Virtual Environments for Collaborative Learning. *Proceedings of the International Conference of Computers in Education/ SchoolNet 2001*. Seoul, Korea. November 12-15. **1**. pp. 3-11.
- Chee, Y.S., Law, N. Lee, K. T., and Suthers, D.(eds.) (2003). The "Second Wave" of ICT in Education: from Facilitating Teaching and Learning to Engendering Education Reform. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5.
- Cheese, P (2003): What Keeps Universities from Embracing e-Learning? *LTI Magazine*. November 5. www.ltimagazine.com/ltimagazine as March 2004.
- Cheng, K.W.E., Chan, C.L., Chan K.W. (2001): Development of a Web-based Virtual Power Electronics Experiment. *Proceedings of the International Conference of Computers in Education/ SchoolNet 2001*. Seoul, Korea. November 12-15. **1**. pp. 47-50.
- Cheung, E.K.W. (2001): Empirical Study of Students' Perceptions in E-Learning. *Proceedings of the International Conference of Computers in Education/ SchoolNet 2001*. Seoul, Korea. November 12-15. **2**. pp. 719-724.
- Chi, M. T. H., Feltovich, P. J. and Glaser, R. (1981): Categorisation and Representation of Physics Problems by Experts and Novices. *Cognitive Science*. **5**. pp 121-152
- Cisco (2000): *Cisco Networking Academy Program*. Available at <http://www.cisco.com/warp/public/779/edu/academy> as at 1/5/01.
- Clark, D. (1999a): *A Time Capsule of Training and Learning: Correspondence Schools*. Northwestern University. Last updated 22/1/00. <http://www.nwlink.com/~donclark/hrd/history/correspondence.html> as at 1/2/05.
- Clark, D. (1999b): *A Time Capsule of Training and Learning: B. F. Skinner (1904 - 1990)*. Northwestern University. Last updated 22/1/00. <http://www.nwlink.com/~donclark/hrd/history/skinner.html> as at 1/2/05.
- Clark, D. (1999c): *A Time Capsule of Training and Learning: Cognitive Science*. Northwestern University. Last updated 22/1/00. <http://www.nwlink.com/~donclark/hrd/history/cognitive.html> as at 1/2/05.
- Clark, D. (1999d): *A Time Capsule of Training and Learning: Computer Based Training (CBT)*. Northwestern University. Last updated 22/1/00. <http://www.nwlink.com/~donclark/hrd/history/cbt.html> as at 1/2/05.

- Clark, D. (1999e): *A Time Capsule of Training and Learning: Constructivism*. Northwestern University. Last updated 22/1/00. <http://www.nwlink.com/~donclark/hrd/history/constructivism.html> as at 1/2/05.
- Collins D. (1995). *Designing Object-Oriented User Interfaces*. Benjamin/Cummings. Redwood City.
- Comer, D. E. (1999): *Computer Networks and Internets*. Second Edition. Prentice Hall. New Jersey.
- Connolly, T., Begg, C. and Strachan, A. (1999): *Database Systems: A Practical Approach to Design, Implementation, and Management*. Second Edition. Addison-Wesley. Harlow.
- Cook, J. F. (1998): *Distance Education for Rural Development: The Third Wave*. Agricultural Extension and Rural Development Department. University of Reading.
- Cooper, A. and Saffo, P. (1999): *The Inmates are Running the Asylum*. SAMS. Indianapolis, Indiana.
- Corbett, A. T., Anderson, J. R., and Patterson, E. G. (1990): Student Modelling and Tutoring Flexibility in the Lisp Tutoring System. *Intelligent Tutoring Systems* (ed. Frasson C. and Cauthier G). Ablex. Norwood, New Jersey.
- Cunningham, S., Ryan, Y., Stedman, L., Tapsall, S., Bagdon, K., Flew, T., and Coaldrake, P. (2000): *The Business of Borderless Education*. Department of Education, Training and Youth Affairs, Commonwealth of Australia.
- DeCloque, P. (Ed) (2000). Illustrated History of Computer Assisted Language Learning. *History of CALL Web Exhibition*. <http://www.history-of-call.org> at 10/05/04
- Devedzic, V. Spector, J.M., Sampson, D.G. and Kinshuk. (eds.) (2003): *The 3rd IEEE International Conference on Advanced Learning Technologies*. Athens, Greece. IEEE Computer Society.
- Dewire, D.T. (1998): *Thin Clients: Delivering Information Over the Web*. McGraw-Hill. New York.
- Dietinger, T., & Maurer, H. (1998). GENTLE - (GEneral Networked Training and Learning Environment). In T. Ottmann, T. & I Tomek (Eds) *ED--Media/ED--Telecom'98*, Charlottesville, VA. AACE, Association for the Advancement of Computers in Education. pp. 358-364.
- Dix, A., Finlay, J., Abowd, G., & Beale R. (1993). *Human Computer Interaction*. Prentice Hall

- Dodds, P. (2003). *ADL Background*. Updated March 1. http://www.rhassociates.com/adl_background.htm as at 8/5/04.
- Donovan, M. and Macklin, S. (1999): The Catalyst Project: Supporting Faculty Uses of the Web...with the Web. *CAUSE/EFFECT Journal*, **22(3)**.
- Drucker, P. (2000): Putting More Now Into The Internet. *Forbes Global*, 15/5/00. Online version available at <http://www.Forbes.com> as at 1/5/01.
- Dyer, L. (2004): Alternative Nation. *Australian PC Authority*. July. **80**.
- Education Review (2004): UK e-university"fiasco". March 31-April 6. Reprinted in *EXMSS Off Campus*. June, 2004.
- Ellis, C., Gibbs, S., and Rein, G.L. (1991): Groupware: some issues and experiences. *Communications of the ACM*. **34 (1)**. pp. 680-689.
- Ellsworth, J., Barron, B. et al. (1997): *The Internet 1997 Unleashed*. Fourth Edition. Sams.Net Publishing. Indianapolis.
- Extramural News (2002): Wired up and ready to go. *Extramural News*. May. Office of the Principal Extramural and International, Massey University. Palmerston North.
- Flynn, I.M. and McHoes, A.M. (1997): *Understanding Operating Systems*. Second Edition. PWS Publishing Company.
- Fowler, H.W. and Fowler F.G. (1974): *The Concise Oxford Dictionary of Current English*. Fifth edition revised by E.McIntosh. Oxford.
- Gehne, R., Jesshope, C.R. and Zhang, J. (2001): Technology Integrated Learning Environment - A Web-based Distance Learning System. *Proceedings of IASTED International Conference 2001, Internet and Multimedia Systems and Applications*. Hawaii, USA. ISBN 0-88986-299-0. pp. 1-6.
- Gentner D. R. and Grudin, J. (1990): Why Engineers (Sometimes) Create Bad Interfaces. *CHI'90 Proceedings*. April. ACM.
- Gerdt, P., Kommers, P, Suhonen,J. and Sutinen, E: (2002): StoryML: An XML Extension for Woven Stories. *Intelligent Tutoring Systems. 6th International Conference, ITS 2002*. Biarritz, France and San Sebastian, Spain. June. Proceedings. pp. 893-902.
- Gledhill, V. (1981): *Discovering Computers*. Science Research Associates. Sydney.
- Goldberg, M. (1997): Communication and Collaboration Tools in WebCT *Proceedings of the Conference on Enabling Network-Based Learning*, May 28 - 30. Espoo, Finland.

- Goldman, J. E., Rawles, P. T. and Mariga, J. R. (1999): *Client/Server Information Systems*. John Wiley & Sons. New York.
- Gonzalez, A. J. and Dankel, D.D. (1993): *The Engineering of Knowledge-Based Systems: Theory and Practice*. Prentice Hall. Englewood Cliffs. New Jersey.
- Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A. and Vassileva, J. (1998): The Intelligent Help Desk: Supporting Peer Help in a University Course. In B.Goettl, H.Halff, C.Redfield, V.Shute (eds.) *Intelligent Tutoring Systems, Proceedings ITS'98*, San Antonio, Texas, LNCS No 1452, Springer Verlag: Berlin pp.494-503.
- Greer, J., McCalla, G., Vassileva, J., Deters, R., Bull, S., and Kettel, L. (2001): Lessons Learned in Deploying a Multi-Agent Learning Support System: The I-Help Experience, *Proceedings of AI in Education AIED'2001*, San Antonio, IOS Press: Amsterdam, pp. 410-421.
- Hall, J. and Sery, P.G. (2000): *Red Hat Linux for Dummies*. IDB Books Worldwide. Foster City.
- Harmon, P. and King, D. (1985): *Expert Systems*. Wiley Press. New York.
- Harold, E. R. and Means, W. S. (2002): *XML in a Nutshell. A Desktop Reference*. O'Reilly & Associates. Sebastopol, California. 2nd Edition.
- Hart, J. (1997): *Win32 System Programming*. Addison-Wesley. Reading.
- Hasegawa, S., Kashihara, A. and Toyoca, J. (2002): An e-Learning Library on the Web. *Proceedings of 2002 International Conference on Computers in Education (ICCE 2002)*. Auckland, Dec 3-6, IEEE Press. pp.1281-82.
- Hayashi, T., Watanabe, K., Hayashida, Y., and Kodo, H. (2001): Remote Lecture Based on Instruction with Blackboard Using High-Quality Audio-Video Stream. *Proceedings of the International Conference of Computers in Education/ SchoolNet 2001*. Seoul, Korea. November 12-15. **2**. pp. 910-913.
- Healey D and Stevens K. (1998): Student Perceptions of Telecommunications Technologies for Accessing Learning Opportunities in Two Northern Canadian Schools. *Journal of Distance Learning*. **4(1)**. Distance Education Association of New Zealand.
- Heinrich E and Kemp R. (2000): A Flexible Scheme For Representing And Retrieving Multimedia Contents In Computer-Based Educational Systems. *IWALT 2000: International Workshop of Advanced Learning Technologies*. Kinshuk, Jesshope C & Okamoto T (eds.). IEEE Computer Society. Los Alamitos.

- Heinrich, E., Johnson, R., Luo, D., Maurer, H. and Sapper, M. (2001). Learner-formulated questions in technology-supported learning applications. *Proceedings of Ed-Media 2001*, Tampere, Finland. Montgomerie, C. and Viteli, J. (eds.). AACE, Vancouver, USA. ISBN 1-880094-42-8. p 43.
- Herman, M. (2004): Microsoft tackles the divide. *Dominion Post*, July 1.
- Higgins, A. (1998): Winds of Change and Paradigm Shifts: Correspondence, Distance and Open Learning. *Journal of Distance Learning*. **4(1)**. Distance Education Association of New Zealand.
- Hill, W. F. (1997): *Learning: A Survey of Psychological Interpretations*. Longman. New York. Sixth edition.
- Honeyball J. (2002a): Flights of fancy. *PC Authority*. **50**. January.
- Honeyball, J. (2002b): .Net - The Microsoft development world. *Australian PC Authority*, April, **53**.
- Hope, W. (2002): Evidence of Unequal Access. *NZ Infotech*. 530.
- Hoppe, U., Lingnau, A., Machado, I., Paiva, A., Prada, R. & Tewissen, F. (2000). Supporting Collaborative Activities in Computer Integrated Classrooms – the NIMIS Approach. *Proc. of 6th International Workshop on Groupware CRIWG 2000*, Madeira, Portugal, 18 - 20 October. IEEE CS Press
- Hoppe, U., Verdejo, F., and Kay, J.(eds.) (2003): *Artificial Intelligence in Education: Shaping the Future of Learning through Intelligent Technologies*. IOS Press. Amsterdam.
- Hsieh, P, Halff H & Redfield C. (1999): Four Easy Pieces: Development Systems for Knowledge-Based Generative Instruction. *International Journal of Artificial Intelligence in Education*. **10**.
- Hurley, M. A. (1998): Features and Functions Overload. *Information Management & Computer Security*, **6(4)**. MCB University Press.
- Huynh, M. Q., and Umesh, U. N. (2003): E-Learning As An Emerging Entrepreneurial Enterprise In Universities And Firms. *Communications of the Association for Information Systems*. **12**. pp. 46-68.
- IHug (2004): High Speed Satellite Internet. <http://www.ihug.co.nz/ultra> as at 5/11/04.
- Inprise (1999a): *Borland Delphi 5 Developer's Guide*. Inprise Corporation, Scotts Valley.

- Inprise (1999b): *Quick Start: Borland Delphi 5 for Windows 98, 95 & Windows NT*. Inprise Corporation, Scotts Valley.
- Ishikawa, T., Matsuda, H, and Takase, H. (2002): Agent Supported Collaborative Learning using Community Web Software. *Proceedings of the International Conference of Computers in Education 2002*. Auckland, New Zealand. December 3-6. **1**. pp. 42-43.
- Ishizuka, T., Horita, T., Moriya, K., Ishihara, K., Yamada T. (2003): PDA Classroom Drill Type Learning System with the Real-time Total Functions of Learning History. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5. pp. 570-75.
- JADE. (2000): *Technical Overview on JADE*. Aoraki Corporation. Christchurch. Available at <http://www.discoverjade.com/jade/techover.htm> as at 25/3/02.
- Jesshope, C., Heinrich, E., and Kinshuk (2000). Technology Integrated Learning Environments for Education at a Distance. *Supporting the Learner through open, flexible and distance strategies: Issues for Pacific Rim Countries, Wellington, New Zealand*.; DEANZ 2000 Conference, 26-29 April 2000, Dunedin, New Zealand. Online publication, <http://www.deanz.org.nz/conf.htm>.
- Jesshope, C., Shafarenko, A. and Slusanschi, H. (1998): Low-bandwidth multimedia tools for web-based lecture publishing. *IEE Engineering Science and Educational Journal*. **7(4)**. pp.148-54.
- Jo, J. H., Moon, K-S, Jones, V and Cranitch, G. (2001): Innovations in e-Learning with Wireless Technology and Personal Digit Assistant. *Proceedings of the International Conference of Computers in Education/ SchoolNet 2001*. Seoul, Korea. November 12-15. **1**. pp. 41-46.
- Jonassen, D., Mayes, J. T. and Mcaleese, R. (1993): A Manifesto for a Constructivist Approach to Technology in Higher Education. T. Duffy, D. Jonassen, & J. Lowyck (Eds), *Designing constructivist learning environments*. Heidelberg, FRG: Springer-Verlag.
- Jones, D and Pritchard, A. (1999): Realising the Virtual University. *Educational Technology*. **39(5)**. September-October. New Jersey.
- Jong, B.S., Lin, T.W., Chan, T.Y., and Wu, Y.L. (2003): Using VR Technology to support the Formation of Cooperative Learning Groups. *The 3rd IEEE International Conference on Advanced Learning Technologies*. Athens, Greece. pp. 37-41. IEE Computer Society.

- Kawato, T., Sasaki, H., and Takeya, M. (2003): Application of the Bootable CD-ROM with a Linux System to the Programming Education. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5 pp. 109-11.
- Keall, C. (ed.) (2001): 16 Education programs. PC World Plus CD-ROM. *New Zealand PC World*. August.
- Kearsley, G. (2003): Explorations in Learning & Instruction: The Theory Into Practice Database. <http://tip.psychology.org/index.html> as at 10/5/04.
- Kinshuk, Jesshope, C and Okamoto, T (eds.). (2000): *IWALT 2000: International Workshop on Advanced Learning Technologies*. December. IEEE Computer Society. Los Alamitos.
- Kruse, K. (2002): E-Learning and the Neglect of User Interface Design. *E-Learning Guru*. http://www.e-learningguru.com/articles/art4_1.htm as at 12/12/04
- Kubarek, D. (1999): Introducing and Supporting a Web Course Management Tool. *Syllabus magazine*. June. <http://www.cit.cornell.edu/atc/cst/SyllabusWeb/syllabus.pdf> as at 13/6/04.
- Larson, K., and Czerwinski, M. (1998): Web page design: implications of memory, structure and scent for information retrieval. *Proceedings of CHI'98*, p25-32.
- Laurillard, D. (1993): *Rethinking University Teaching*. Routledge. London.
- Learnz2001. (2001): Overview of Learnz2001: Island Odyssey: http://www.learnz.org.nz/2001/programme_2001.htm as at 1/5/01.
- Lennox, D. (2000): Improving the Top Line Using e-Learning. *WBT White Paper*. Maryland. <Http://www.wbtsystems.com> as at 10/5/01.
- LinuxDevices (2005): Pocketable Linux server creates plug-and-go Linux desktop. *LinuxDevices.com*. August 16. <http://www.linuxdevices.com/news/NS8562564746.html> as at 20/08/05.
- Lopez, N., Nunez, M., Rodriguez, I., and Rubio, F. (2002): Including Malicious Agents into a Collaborative Learning Environment. *Intelligent Tutoring Systems. 6th International Conference, ITS 2002*. Biarritz, France and San Sebastian, Spain. June. Proceedings. pp. 51-60.
- Luger, G.F. and Stubblefield, W.A. (1998): *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Third Edition. Addison Wesley Longman. Harlow, England.

- Macromedia (2005): *Macromedia Authorware 7*. Produce rich-media courseware for e-learning. <http://macromedia.com/software/authorware/> as at 21/09/05.
- Marland, P. (1997): *Towards More Effective Open & Distance Teaching*. Kogan Page. London.
- Martin, B. and Mitrovic, A. (2003): ITS Domain Modelling: Art or Science? *Artificial Intelligence in Education*. Hoppe, H.U. et al. (eds.). IOS Press. pp. 183-90.
- Mazza, R. and Dimitrova, V. (2003): CourseVis: Externalising Student Information to Facilitate Instructors in Distance Learning. *Artificial Intelligence in Education*. Hoppe, H.U. et al. (eds.). IOS Press. pp. 279-86.
- Massey Extramural. (2004): Massey University Extramural web site. <http://extramural.massey.ac.nz/welcome.htm> as at 1/2/04.
- Massey University (2000): Online Learning At Massey University. *Report of the Information Technology and Distance Education Taskforce*. Massey University. October.
- Mayes, J. T. (1993): Commentary: Impact of cognitive theory on the practice of courseware authoring. *Journal of Computer Assisted Learning*, **9**, 222-228. <http://apu.gcal.ac.uk/clti/papers/TMPaper12.html> as at 10/5/04.
- Merceron, A. and Yacef, K. (2003): A Web-Based Tutoring Tool with Mining Facilities to Improve Learning and Teaching. *Artificial Intelligence in Education*. Hoppe, H.U. et al. (eds.). IOS Press. pp. 201-08.
- Microsoft (1997): *Encarta 97 Encyclopedia*. <http://www.encarta.com>.
- Microsoft (2005): Overview of the .NET framework. *.NET Framework Developers Guide*. Ms-help://MS.NETFrameworkSDK/cpguidenf/html/cpovrintroductiontonetframeworks as at 5/01/2005.
- Militant (2001): "Without culture there can be no freedom". *The Militant*, **65(16)**. April 23. New York.
- Mills, S. (ed). (1997): *Turning Away From Technology. A New Vision For the 21st Century*. Sierra Club Books. San Francisco.
- MIT (2005): MIT OpenCourseWare. <http://ocw.mit.edu/OcwWeb/Global/AboutOCW/about-ocw.htm> as at 21/09/05.
- Mitrovic, A. and Hausler, K. (2000): Porting SQL-Tutor to the Web. *Proc. ITS'2000 workshop on Adaptive and Intelligent Web-based Education Systems*, pp. 37-44.

- Mitsuhara, H., Ochi, Y., Kanenishi, K. and Yano, Y. (2002): A Web Retrieval Support System with a Comment Sharing Environment: Toward and Adaptive Web-based IR System. *Proceedings of 2002 International Conference on Computers in Education (ICCE 2002)*. Auckland, Dec 3-6, IEEE Press. pp.1218-22.
- Moran, T. P. (1981): An Applied Psychology Of The User. *Computing Surveys*. **13(1)**. March.
- Mostow, J., Roth, S., Hauptmann, A. G. and Kane, M. (1994): "A Prototype Reading Coach that Listens", *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, American Association for Artificial Intelligence, Seattle, WA. August, 1994. pp. 785-792.
- Munro, A., Johnson, M., Surmon, D. and Wogulis, J. (c.1995): Specifying Interactive Graphical Behaviours in RIDES. A Working Paper in Progress. Behavioural Technology Laboratories, University of Southern California. <http://btl.usc.edu/rides/authInterf/behav.html> as at 10/5/01.
- Murray, T. (1998a) A Model for Distributed Curriculum on the World Wide Web? *Journal of Interactive Media in Education*, 1998.
- Murray, T. (1998b): Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. *Journal of the Learning Sciences* (Special Issue on Authoring Tools for Interactive Learning Environments). **7(1)**. pp. 5-64.
- Murray, T. (1999): Authoring Intelligent Tutoring Systems: An analysis of the state of the art. *Int. J. of AI and Education*. **10 (1)**, pp. 98-129.
- Murray, T., Condit, C. And Haugsjaa, E. (1998): MetaLinks: A Preliminary Framework for Concept-Based Adaptive Hypermedia. *Workshop proceedings for ITS-98 workshop on WWW-Based Tutoring*. San Antonio, August, 1998.
- Murray, T., Condit, C., Piemonte, J., Shen, T., & Khan, S. (2000). Evaluating the Need for Intelligence in an Adaptive Hypermedia System. *ITS 2000*. pp.373 -382.
- Najjar, L. J. (1990). *Using color effectively* (IBM TR52.0018). Atlanta, GA: IBM Corporation.

- Nakabayashi, K, Maruyama, M, Koike, Y, Fukuhara, Y & Nakamura Y. (1996): *An Intelligent Tutoring System on the WWW Supporting Interactive Simulation Environment with a Multimedia Viewer Control Mechanism*. NTT Information and Communication Systems Laboratories, Tokyo 180, Japan.
<http://teis.virginia.edu/aace/conf/webnet/html/321/321.htm>
- Naughton, P. and Schildt, H. (1999): *Java 2: The Complete Reference*. Third Edition. Osborne/McGraw-Hill. Berkeley.
- Newell, A and Simon, H A. (1972): *Human Problem Solving*. Prentice-Hall.
- Nielsen, J. (1994): Enhancing the explanatory power of usability heuristics. *Proceedings of ACM CHI'94*. pp.52-58.
- Nielsen, J. (2001): Jakob Nielsen on e-learning. *Elearning Post*. January 15.
<http://www.elearningpost.com/features/>.
- Noble, D. (1997): Digital Diploma Mills Part 1: The Automation of Higher Education. October. <http://communication.ucsd.edu/dl/> as at 1/5/01.
- Noble, D. (1998a): Digital Diploma Mills, Part II: The Coming Battle over Online Instruction. March. <http://communication.ucsd.edu/dl/> as at 1/5/01.
- Noble, D. (1998b): Digital Diploma Mills, Part III: The Bloom is off the Rose. November. <http://communication.ucsd.edu/dl/> as at 1/5/01.
- Noble, D. (1999): Digital Diploma Mills Part IV Rehearsal for the Revolution. November. <http://communication.ucsd.edu/dl/> as at 1/5/01.
- Norton, A.-L. (ed.) (1994): *The Hutchinson Dictionary of Ideas*. Helicon. Oxford.
- Novak, J. D. and Gowin, D. B. (1984): *Learning how to learn*. Cambridge University Press, New York 1984
- Nunes, M.B., and McPherson, M. (2003): Constructivism vs. Objectivism: Where is the difference for Designers of e-Learning Environments? pp. 496-500. *The 3rd IEEE International Conference on Advanced Learning Technologies*. Athens, Greece. IEEEComputer Society. pp. 496-500.
- NZSC (2004): *The Correspondence School of New Zealand* web site.
<http://www.correspondence.school.nz/> as at 1/2/04.
- NZWireless(2004): web site <http://www.nzwireless.org/> as 21/7/04.

- Okazaki, Y., Feng, X. Y., Okamoto, M., and Kondo, H. (2003): Stroke Segmentation and Symbol Identification in On-line Handwriting Mathematical Expressions for a WWW-based Intelligent Tutoring System. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5. pp. 715-18.
- One News (2002): Students receive mobile phones. *nzoom.com*. http://onenews.nzoom.com/news_detail/0,1227,92163-1-8,00.html as at 6/4/02.
- Oppermann, R., Rashev, R. and Kinshuk (1997): Adaptability and Adaptivity in Learning Systems. *Knowledge Transfer (Volume II)* (Ed. A. Behrooz), pAce, London, pp. 173-179 (ISBN 1-900427-015-X).
- Osier, D., Grobman, S. and Batson, S. (1997): *Teach Yourself DELPHI 3 in 14 Days*. Sams Publishing. Indianapolis.
- Paiva, A. and Machado, I. (1998): Vincent, an Autonomous Pedagogical Agent for on-the-Job Training. *Proc. 4th International Conference, ITS '98*. Goettl, B., Half, H., Redfield, C. and Shute, V. (eds.) San Antonio, Texas. Aug 16 -19. Springer. Berlin.
- Park, O., Perez, R. and Seidel, R. (1987): Intelligent CAI: Old Wine in New Bottles, or a New Vintage. *Artificial Intelligence and Instruction*. Kearsley, G. (ed.). Addison-Wesley. Reading.
- Patel, A. and Kinshuk. (1997): Intelligent Tutoring Tools in a Computer-Integrated Learning Environment for Introductory Numeric Disciplines. *Innovations in Education and Training International*. **34(3)**. August. Journal of Staff and Educational Development Association. Kogan Page. London.
- Patton, M. Q. (1990). *Qualitative Evaluation Methods*. Beverly Hills, CA.: Sage Publications.
- PBS (2004): Public Broadcasting Service Distance Learning Week-Timeline <http://www.pbs.org/als/dlweek/history/1960.htm> as at 1/02/04.
- Petzold, C. (1996): *Programming Windows 95*. Microsoft Press. Redmond.
- Pinkwart, N., Hoppe, H.U., Bollen, L. and Fuhlrott, E. (2002): Group-Oriented Modelling Tools with Heterogeneous Semantics. *Intelligent Tutoring Systems. 6th International Conference, ITS 2002*. Biarritz, France and San Sebastian, Spain. June. Proceedings. pp. 21-30.
- Prebble, T. (2001a): New Zealand. *Open and Distance Education In The Asia Pacific Region*. Jegede, O. and Shive, G. (eds.) pp. 381-399.

- Prebble, T. (2001b): Contact Courses- Who Needs Them? *Extramural News*. July. Massey University. New Zealand.
- Preece, J. (1994): *Human-Computer Interaction*. Addison-Wesley. Wokingham.
- Pressman, R. (1997): *Software Engineering: A Practitioner's Approach*. International Editions. McGraw-Hill. New York.
- Queensland College (2002). NuMaths. Queensland College of Tertiary Studies. <http://maths.newzealand.edu> as at 8/3/02.
- Quinn, V. (2002): That about WAPs up 2001. Australian PC Authority, January.
- Rasseneur, D., Delozanne, E., Jacoboni, P., and Grugeon, B. (2002): Learning with Virtual Agents: Competition And Cooperation in AMICO. *Intelligent Tutoring Systems. 6th International Conference, ITS 2002*. Biarritz, France and San Sebastian, Spain. June. Proceedings. pp. 61-70.
- Reeves, T. C. and Harmon, S. W. (1994): Systematic Evaluation Procedures For Interactive Multimedia For Education And Training. *Multimedia Computing: preparing for the 21 Century*. (ed. Reisman, S.) Idea Group. Harrisburg.
- Reeves, T.C. (1992): Effective Dimensions of Interactive Learning Systems. *Information Technology for Training and Education Conference*. Brisbane
- Reynolds, T. J. and Gutman, J. (1988): *Laddering Theory, Method, Analysis, and Interpretation*. Journal of Advertising Research, February/March. pp.11-29.
- Ritter, S. (1997): PAT Online: A Model-tracing tutor on the World-wide Web. *Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society*, Kobe, Japan, 18-22 August.
- Ritter, S., Anderson, J., Cytrynowicz, M. and Medvedeva, O. (1998): Authoring Content in the PAT Algebra Tutor. *Journal of Interactive Media in Education*. 8 Oct. **98(9)**. Open University, United Kingdom.
- Roseth, B. (2001): Catalyst aim: Better tools for better classrooms. University Week. March 8. University of Washington.
- SCORM (2002): Sharable Courseware Object Reference Model <http://www.rhassociates.com/scorm.htm> as at 24/6/04.
- Scott, C. A., Clayton, J. E., & Gibson, E. L. (1991). *A Practical Guide to Knowledge Acquisition*. Reading: Addison-Wesley.

- Searle, S. (2001): Digital television a possible answer to rural internet woes. *Rural News*. 16/4/01. Auckland.
- Seidel, R. and Chatelier, P. (eds.). (1995): *Learning Without Boundaries*. Plenum Press. New York.
- Shneiderman, B. (1998): *Designing the User Interface*. Addison-Wesley. Reading, Massachusetts.
- Sinista, K. and Manako, A. (1999): Interactive Dictionary as an Information Wish-Maker. *Educational Technology*. **39(5)**. September-October. New Jersey.
- Smulders, D. (2003). Designing for Learners, Designing for Users, http://elearnmag.org/subpage/sub_page.cfm?section=3&list_item=11&page=1 as at 4/5/04.
- Stern, M. K. and Woolf, B. P. (1998): Curriculum Sequencing in a Web-based Tutor. *Proc. 4th International Conference, ITS '98*. San Antonio, Texas. Aug 16 - 19. Springer. Berlin
- Stirling, P. (2001): The Mouse Trap. *The Listener*. July 14.
- Swart, B. (2001): Migrating Your Delphi 5 Projects to Kylix. *Borland Software Corporation*. www.Borland.com.
- Tagg, R. and Freyberg, C. (1997): *Designing Distributed and Cooperative Information Systems*. International Thompson Computer Press. London.
- Tanenbaum, A. S. (1996): *Computer Networks*. Third Edition. Prentice-Hall International. London.
- Tansey, F. (2003): The Standard Bearers Close Ranks. *Syllabus Magazine*. March 1.
- Taylor P. (2000): Beware geeks bearing modems. *The Times Higher Education Supplement*. September 1.
- Tiffin, J. (2002): The HyperClass: Education in a Broadband Internet Environment. *Proceedings of the International Conference of Computers in Education, 2002*. Auckland, New Zealand. December 3-6. 1. pp. 23-29.
- Tominaga, H., Hagihara, H., Matsubara, Y., Yamasaki, T., and Yano Y. (2003): Fill-in Type Drill-based Web Learning System for Basic IT Education. *Proceedings of the International Conference of Computers in Education 2003*. Hong Kong, China. December 2-5. pp. 764-66.

- TOPNZ. (2004): The Open Polytechnic of New Zealand web site. <http://www.topnz.ac.nz/> as at 1/2/04.
- Uglow, J. (2002): *The Lunar Men. The Friends who made the Future*. Faber and Faber. London.
- Underwood, J. D. M., and Underwood, G. (1990): *Computers and Learning: Helping Children Acquire Thinking Skills*. Blackwell. Oxford.
- Universal (2001): UNIVERSAL web site: <http://www.ist-universal.org/sommaire.htm> as at 10/5/01.
- UP (2004): University of Phoenix web site. <http://onl.uophx.edu/default.aspx> as at 8/5/04.
- UTTC. (2004): UT TeleCampus - Overview 2004. The University of Texas System Online www.telecampus.utsystem.edu as at 8/5/04.
- Vassileva J. (2001): "Distributed and United", Invited talk at ICCE'2001, Seoul, 12-15 Nov. Available at <http://julita.usask.ca/homepage/Agents.htm> as at 19-Feb-02.
- Vassileva, J. (1997): "Dynamic Course Generation on the WWW". *Artificial Intelligence in Education* (du Boulay, B. and Mizoguchi, R. eds).. IOS Press.
- Vizcaíno, A., du Boulay, B (2002): Using a Simulated Student to Repair Difficulties in Collaborative Learning. *Proceedings of the International Conference of Computers in Education 2002*. Auckland, New Zealand. December 3-6. 1. pp. 349-353.
- Watt, S. (2000): Course Web Sites. 7th February. *KMi-Occasional Papers-1*. Open University. United Kingdom. <http://kmi.open.ac.uk/publications/occasionalpapers.html> (as at 1/5/01)
- Web-Based Education Commission. (2000). *The Power of the Internet for Learning. Report to the President and Congress of the United States*. Washington D.C. December.
- Weber, G., and Brusilovsky, P. (2001): ELM-ART: An Adaptive Versatile System for Web-based Instruction. *International Journal of Artificial Intelligence in Education*. 12.
- Wells, N.D. (2000): *Linux! I didn't know you could do that...* Sybex. San Francisco.
- Wenger, E. (1998) *Communities of Practice: Learning, Meaning and Identity* Cambridge University Press.
- Whitten, J.L., and Bentley, L.D. (1998): *System Analysis and Design Methods*. Irwin McGraw-Hill. Boston. Fourth Edition.

- Woolf, B. (1990): 20 Years in the Trenches: What have we Learned? *Intelligent Tutoring Systems* (eds. Frasson, C & Gauthier G). Ablex. New Jersey. pp. 234-250.
- Woolley, D.R. (1994). PLATO: The emergence of on-line community. *Computer-Mediated Communication Magazine*, **1(3)**, July 1. p.5. Available online: <http://www.december.com/cmc/mag/1994/jul/plato.html>.
- Wright, H. (2002a): Getting up to speed. *NZ Infotech*. **525**. Feb 18.
- Wright, P., Mosser-Wooley, D. and Wooley, B. (2001). Techniques & tools for using color in computer interface design. *ACM Crossroads*. On-line journal. Available: <http://www.acm.org/crossroads/xrds3-3/color.html> as at 12/12/04.
- Wyld, H. C. (ed.). (1932): *Universal Dictionary of the English Language*. The Amalgamated Press Ltd. London.
- Yellen R E. (1999): The On-Line Trainee: A Forgotten End User. *Journal of End User Computing*. **11(3)**. July-Sept.
- Yin, R. K. (1984). *Case Study Research: Design & Methods*. First edition. London: Sage Publications.

Appendix A

References for e-Learning Systems Reviewed

A.1: Paper References for Initial System Comparison¹

<i>Tool</i>	<i>References</i>
Blackboard / Courseinfo	
CALAT	Nakabayashi, K, Maruyama, M, Koike, Y, Fukuhara, Y & Nakamura Y. An Intelligent Tutoring System on the WWW Supporting Interactive Simulation Environment with a Multimedia Viewer Control Mechanism. NTT Information and Communication Systems Laboratories, Tokyo 180, Japan. http://teis.virginia.edu/ace/conf/webnet/html/321/321.htm
CATALYST.	Donovan, M & Macklin, S. <i>The Catalyst Project: Supporting Faculty Uses of the Web...with the Web.</i> in <i>CAUSE/EFFECT</i> journal, Volume 22 Number 3 1999.
CILE /Byzantium Project	Patel, A and Kinshuk. <i>Tutoring Tools in a Computer-Integrated Learning Environment for Numeric Disciplines.</i> In: <i>Innovations in Education and Training International.</i> Volume 34(3). August 1997. Journal of Staff and Educational Development Association. Kogan Page. London.
CNAP	
DIY	Watt, S. <i>Course Web Sites.</i> 7th February 2000. KMi-Occasional Papers-1. Open University. United Kingdom. http://kmi.open.ac.uk/publications/occasionalpapers.html
DCG on WWW	Vassileva J. (1997) DCG + WWW: Dynamic Courseware Generation on the WWW, Proceedings of AIED'97, Kobe, Japan, 18-22.08.1997, IOS Press, 498-505. Vassileva J. (1995) Dynamic Courseware Generation: at the Cross Point of CAL, ITS and Authoring. in Proceedings <i>International Conference on Computers in Education, ICCE'95</i> , Singapore, 5-8 December 1995, 290-297.
Eon	Murray, T. (1998). Authoring Knowledge Based Tutors: Tools for Content, Instructional Strategy, Student Model, and Interface Design. <i>J of the Learning Sciences (Special Issue on Authoring Tools for Interactive Learning Environments).</i> Vol 7, No.1, pp. 5-64.

¹ Compiled between May 2001 and March 2002.

GENTLE-WBT/ Hyperwave eLearning Suite	Dietinger, T, Maurer, H. <i>GENTLE - (General Networked Training and Learning Environment)</i> . Graz University of Technology – Institute for Information processing and Computer supported new Media, AUSTRIA. 1998
Highly Interactive Tutorials	<p>Bork, A. Tutorial Learning for the New Century. <i>Journal of Science Education and Technology</i>. March 2001, Volume 10, No 1, pp. 57-71</p> <p>Bork, A. What is Needed for Effective Learning on the Internet. <i>Educational Technology and Society. Special Issue on Curriculum, Instruction, Learning, and the Internet</i>. Vol 4, No 3, 2001, pp 139 – 144.</p>
I-HELP	<p>Greer J., McCalla G., Vassileva J., Deters R., Bull S., Kettel L. (2001) Lessons Learned in Deploying a Multi-Agent Learning Support System: The I-Help Experience, <i>Proceedings of AI in Education AIED'2001</i>, San Antonio, IOS Press: Amsterdam, 410-421.</p> <p>Greer, J., McCalla, G., Cooke, J., Collins, J., Kumar, V., Bishop, A. and Vassileva, J. (1998) The Intelligent HelpDesk: Supporting Peer Help in a University Course, in B.Goettl, H.Half, C.Redfield, V.Shute (eds.) <i>Intelligent Tutoring Systems, Proceedings ITS'98</i>, San Antonio, Texas, LNCS No1452, Springer Verlag: Berlin pp.494-503.</p>
IMT/PICAT/PIMECH	
INDIE	<p>Dobson, W.D. and Riesbeck C.K. <i>Tools for Incremental Development of Educational Software Interfaces</i>. Northwestern University. http://www.cs.nwu.edu/~wolff/chi98/chi98.html;</p> <p>Dobson, D.J <i>Authoring Tools for Investigate-and-Decide Learning Environments</i>. PhD thesis. Northwestern University. 1998. Illinois. http://www.cs.nwu.edu/~wolff/thesis</p>
Interactive Dictionary	Sinista, K and Manako, A. <i>Interactive Dictionary as an Information Wish-Maker</i> . In: <i>Educational Technology</i> . Volume 39(5). September-October 1999. New Jersey.
InterBook	<p>Brusilovsky, P., Eklund, J., and Schwarz, E. (1998) Web-based education for all: A tool for developing adaptive courseware. <i>Computer Networks and ISDN Systems (Proceedings of Seventh International World Wide Web Conference, 14-18 April 1998)</i> 30 (1-7), 291-300.</p> <p>Brusilovsky, P. and Anderson, J. (1998) ACT-R electronic bookshelf: An adaptive system for learning cognitive psychology on the Web. <i>Proceedings of The 3rd World Conference of the WWW, Internet, and Intranet, WebNet'98</i>, Orlando, FL, November 7-12, 1998, AACE, pp. 92-97 (Top Paper Award).</p>

LEARNZ	
LISP Tutor	<p>Anderson, J. R., Corbett, A. T., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. <i>The Journal of Learning Sciences</i>, 4, 167-207.</p> <p>Anderson, J R, Corbett, A T, & Patterson, E G. Student Modelling and Tutoring Flexibility in the Lisp Tutoring System. <i>Intelligent Tutoring Systems</i> (ed. Frasson C & Cauthier G). Ablex. Norwood, New Jersey. 1990.</p> <p>Anderson, J.R. and Reiser, B.J. (1985). The LISP Tutor. <i>Byte</i>, 10, 4, 159-175.</p>
MANIC	<p>Stern M K & Woolf B P. Curriculum Sequencing in a Web-based Tutor. <i>Proc. 4th International Conference, ITS '98</i>. San Antonio, Texas. Aug 16 - 19, 1998. Springer. Berlin.</p> <p>A. Schapira, K. De Vries, C. Pedregal-Martin, "MANIC: An Open-Source System to Create and Deliver Courses over the Internet," <i>Proc. 2001 Symposium On Applications and the Internet (IEEE Computer Society Press)</i> (San Diego, Calif., USA, Jan. 2001).</p>
MDC	<p>Murray, Tom. A Model for Distributed Curriculum on the World Wide Web? Draft. (For the final version see the Journal of Interactive Media in Education, 1998)</p>
METALINKS	<p>Murray, T., Condit, C., Piemonte, J., Shen, T., Khan, S. (2000). Evaluating the Need for Intelligence in an Adaptive Hypermedia System. Submitted to ITS 2000.</p> <p>Murray, T., Condit, C. & Haugsjaa, E. (1998). MetaLinks: A Preliminary Framework for Concept-Based Adaptive Hypermedia. <i>Workshop proceedings for ITS-98 workshop on WWW-Based Tutoring</i>. San Antonio, August, 1998.</p>
PAT	<p>Ritter, S, Anderson , J, Cytrynowicz, M & Medvedeva, O. Authoring Content in the PAT Algebra Tutor. <i>Journal of Interactive Media in Education</i>. 8 Oct 98. 98(9). Open University, United Kingdom.</p> <p>Ritter, S. PAT Online: A Model-tracing tutor on the World-wide Web. <i>Proceedings of the workshop "Intelligent Educational Systems on the World Wide Web", 8th World Conference of the AIED Society, Kobe, Japan, 18-22 August 1997</i>.</p>
PLATO	<p>Woolley, D.R. (1994). PLATO: The emergence of on-line community. <i>Computer-Mediated Communication Magazine</i>, 1(3), 5. Available online: http://www.december.com/cmcmag/1994/jul/plato.html</p>

RIDES	<p>Monroe A, Johnson M, Pizzini P, Surmon D & Wogulis J. A Tool for Building Simulation-Based Learning Environments. <i>Behavioural Technology Laboratories</i>, University of Southern California. http://btl.usc.edu/rides/shortPapers/bldsim.html</p> <p>Monroe A, Johnson M, Pizzini P, Surmon D & Wogulis J. Specifying Interactive Graphical Behaviours in RIDES. A Working Paper in Progress. <i>Behavioural Technology Laboratories</i>, University of Southern California. http://btl.usc.edu/rides/authInterf/behav.html</p>
SQL-Tutor; SQLT-Web	<p>Mitrovic, A., Hausler, K. Porting SQL-Tutor to the Web. <i>Proc. ITS'2000 workshop on Adaptive and Intelligent Web-based Education Systems</i>, pp. 37-44, 2000.</p> <p>Mitrovic, A, Ohlsson, S. Evaluation of a Constraint-based Tutor for a Database Language. <i>International Journal of Artificial Intelligence in Education</i>, 10, 1999. Pp 238-256.</p>
TILE	<p>Gehne, Regina; Jesshope, Chris; and Zhang, Jenny. Technology Integrated Learning Environment - a Web-based Distance Learning System. Draft paper. Massey University. 2001;</p> <p>Jesshope C., Heinrich E. & Kinshuk (2000). Technology Integrated Learning Environments for Education at a Distance. <i>Supporting the Learner through open, flexible and distance strategies: Issues for Pacific Rim Countries</i>, Wellington, New Zealand: The Distance Education Association of New Zealand, 348ff.</p>
TOP-CLASS	<p>Lennox, Duncan. Improving the Top Line Using e-Learning. <i>WBT White Paper</i></p>
UNIVERSAL	<p>UNIVERSAL - Design and Implementation of A Highly Flexible E-Marketplace of Learning Resources. Draft conference paper.2001;</p> <p>Jerman-Blazic, B. The Usability Aspects of an Universal Brokerage and Delivery System for the Pan-European Education. Kinshuk, Jesshope, C and Okamoto, T (eds). <i>IWALT 2000: International Workshop on Advanced Learning Technologies</i>. December 2000 IEEE Computer Society. Los Alamitos.</p>
VINCENT	<p>Paiva A & Machado I. Vincent, an Autonomous Pedagogical Agent for on-the-Job Training. <i>Proc. 4th International Conference, ITS '98</i>. Goettl B, Half H, Redfield C & Shute V (eds) San Antonio, Texas. Aug 16 -19, 1998. Springer. Berlin. (Filed under proceedings)</p>

WEBCT	<p>Goldberg, M. Communication and Collaboration Tools in WebCT <i>Proceedings of the conference Enabling Network-Based Learning</i>, May 28 - 30, 1997, Espoo, Finland</p> <p>Goldberg, M & Salari, S. WebCT: UBC's Web course tool. <i>Campus Computing and Communications</i>, May/June 1996, University of British Columbia</p>
WITS	<p>Callear, David. Intelligent Tutoring Environments as Teacher Substitutes: Use and Feasibility. <i>Educational Technology</i>. Volume 39(5). September-October 1999. New Jersey.</p>
XAIDA	<p>Wenzel, B & Dirnberger, M Experimental Advanced Instructional Design Advisor (Xaida) Training: A Multidimensional Approach To Training Evaluation. <i>Mei Technology Corporation</i>. San Antonio, Texas. 1996. Available online at: http://www.ijoa.org/imta96/paper80.html</p> <p>Hsieh, P, Half H & Redfield C. Four Easy Pieces: Development Systems for Knowledge-Based Generative Instruction. <i>International Journal of Artificial Intelligence in Education</i>. 1999 (10).</p>
Overviews	
Adaptive and Intelligent Technologies for Web-based Education.	<p>Brusilovsky, P. Adaptive and Intelligent Technologies for Web-based Education. In: C. Rollinger and C. Peylo (eds.) <i>Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching</i>, 1999, 4, 19-25.</p>
Advanced Learning Technologies	<p>Kinshuk, Jesshope, C and Okamoto, T (eds). <i>IWALT 2000: International Workshop on Advanced Learning Technologies</i>. December 2000 IEEE Computer Society. Los Alamitos</p>
Authoring Intelligent Tutoring Systems	<p>Murray, T. Authoring Intelligent Tutoring Systems: An analysis of the state of the art. Draft paper. Computer Science Dept., University of Massachusetts, Amherst, 1999.</p>
Comparison of WebCT and CourseInfo	<p>Kanjilal, U. Web-based Distance Education: Considerations for Design and Implementation. <i>Indian Journal of Open Learning</i>. Vol. 9(3), pp 433-440. New Delhi.</p>
Universities online	<p>Jones, D and Pritchard, A. Realising the Virtual University. <i>Educational Technology</i>. Volume 39(5). September-October 1999. New Jersey.</p> <p>Brusilovsky, P. and Miller, P. Course Delivery Systems for the Virtual University. In: T. Tschang and T. Della Senta (eds.): <i>Access to Knowledge: New Information Technologies and the Emergence of the Virtual University</i>. Amsterdam: Elsevier Science. 2001.</p>

A.2: Web References for Initial System Comparison²

Tool	References
BLACKBOARD / CourseInfo	http://company.blackboard.com; http://support.blackboard.com http://www.c2t2.ca/landonline/shownote.asp?appRow=11 http://www.cit.cornell.edu/atc/cst http://www.cit.cornell.edu/atc/cst/why.shtml
CALAT	http://www.nttlabs.com/text_only_html/calat.html http://teis.virginia.edu/aace/conf/webnet/html/321/321.htm
CATALYST.	http://catalyst.washington.edu/; http://depts.washington.edu/~uweek/archives/2001.03.MAR_08/article7.html http://www.educause.edu/ir/library/html/cem9934.html
CILE / Byzantium Project	http://fims-www.massey.ac.nz/%7Ekinshuk/papers/ieti97.html; http://ifets.gmd.de/periodical/vol_1_2000/patel.html; http://www.mgt.uea.ac.uk/cti/events/events_past/Byzantium_Workshop.html
CNAP	http://www.cisco.com/warp/public/779/edu/academy
DIY	http://is157332.massey.ac.nz http://www.macromedia.com/software/authorware
DCG on WWW	http://julita.usask.ca
Eon	http://helios.hampshire.edu/~tjmCCS/papers/JLSEon/JLS96.html
GENTLE-WBT.	http://wbt-2.iicm.edu/product http://wbt-2.iicm.edu/gentle/papers/edmedia98.pdf http://wbt-2.iicm.edu/gentle/GentleIntro.htm

² Compiled between May 2001 and March 2002.

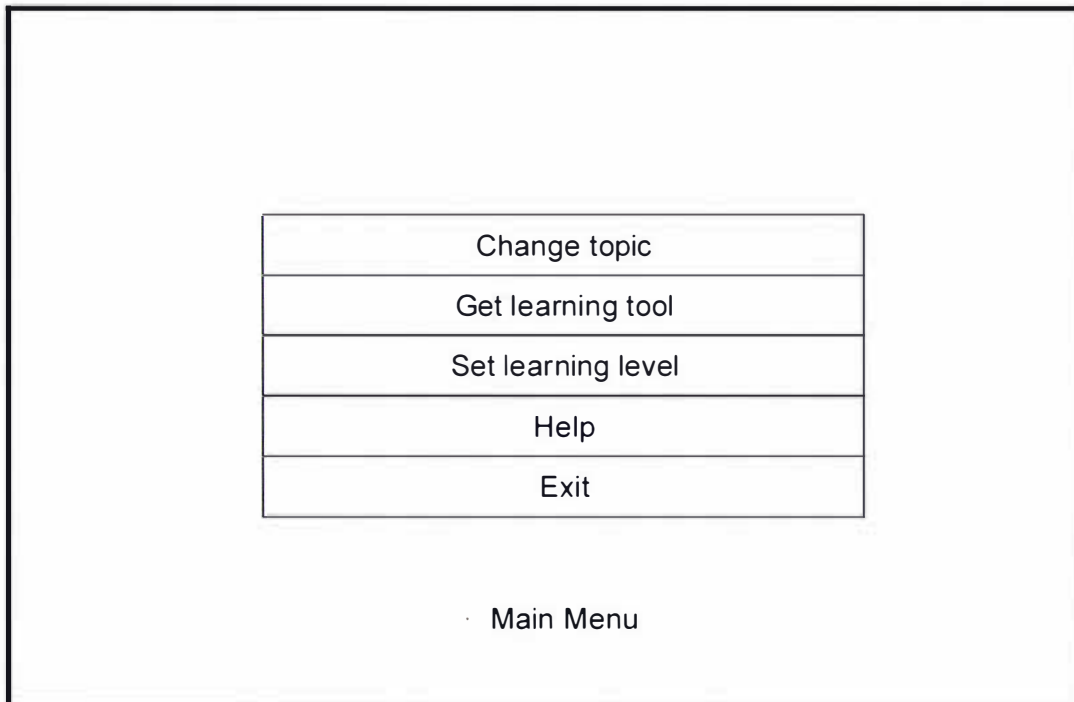
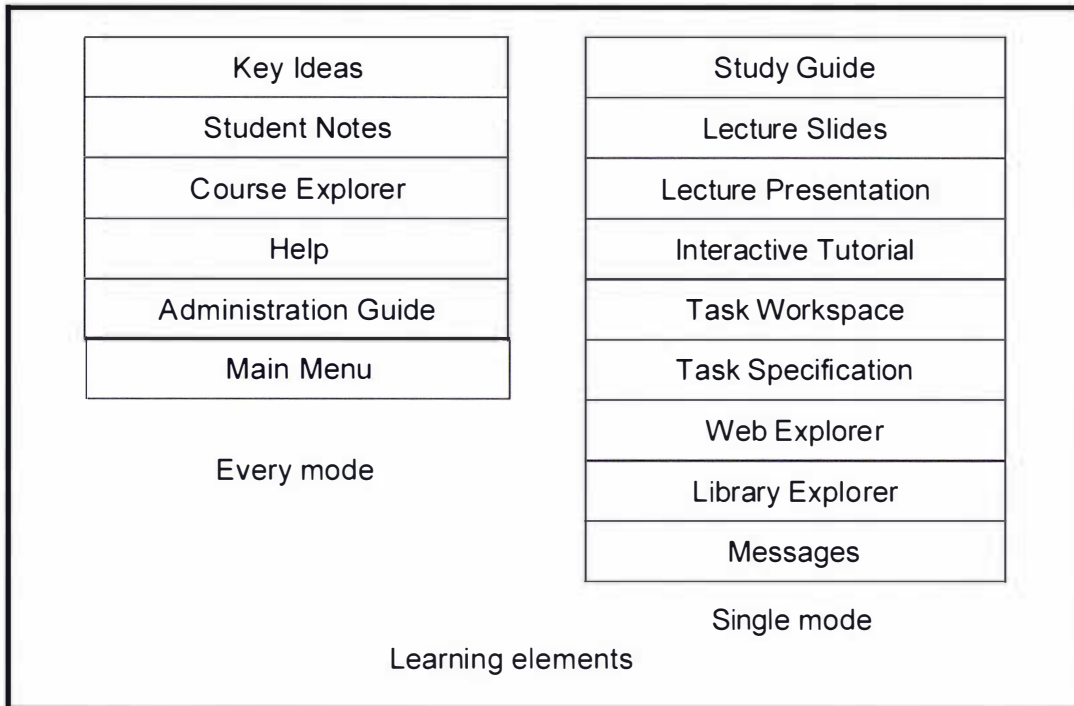
Hyperwave eLearning Suite	http://www.hyperwave.de/e/news_events/news_pr_29.html ; http://www.hyperwave.de/e/products/els.html
Highly Interactive Tutorials	http://www1.ics.uci.edu/~bork
I-HELP	http://julita.usask.ca
IMT/PICAT/PIMECH	http://www.bpta.co.uk/
INDIE	http://www.il.s.nwu.edu/~riesbeck/indie ; http://www.cs.nwu.edu/~wolff/chi98/chi98.html ; http://www.cs.nwu.edu/~wolff/thesis
INTERACTIVE DICTIONARY	http://www.dlab.kiev.ua/hci99.htm ; http://www.dlab.kiev.ua/wis_m.htm
INTERBOOK	http://www.contrib.andrew.cmu.edu/~plb/InterBook.html ; http://ausweb.scu.edu.au/proceedings/eklund/paper.html ; http://www7.scu.edu.au/programme/fullpapers/1893/com1893.htm
LISP TUTOR	http://www.cs.nwu.edu/~wolff/thesis/node113.html ; http://act.psy.cmu.edu/ACT/papers/Lessons_Learned.html
MANIC	http://none.cs.umass.edu/manic/documents/manicdb-saint01.html http://www-aml.cs.umass.edu/~stern/bib.html http://www-aml.cs.umass.edu/~stern/manic.html
MDC	http://helios.hampshire.edu/~tjmCCS/papers/DisCurJIME98/JIMEMurray.html
METALINKS	http://helios.hampshire.edu/~tjmCCS/papers/ITS2000/ITS2000subMurray.html
PAT	http://act.psy.cmu.edu/ACT/awpt/awpt-home.html ; http://www.cyberedinc.com
PLATO	http://www.december.com/cmc/mag/1994/jul/plato.html http://www.coe.uh.edu/courses/cuin6373/idhistory/plato.html http://www.wired.com/news/topstories/0,1287,2614,00.html
RIDES	http://btl.usc.edu/rides/documentn/refMan/index.html ; http://btl.usc.edu/rides/index.html ; http://btl.usc.edu/rides/shortPapers/bldsim.html

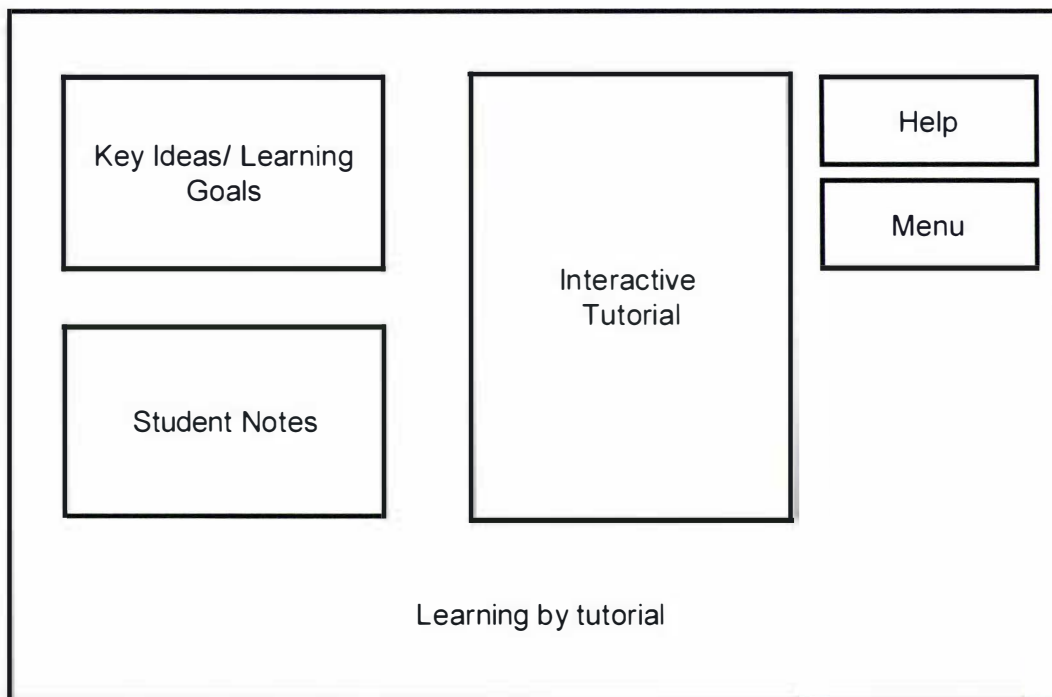
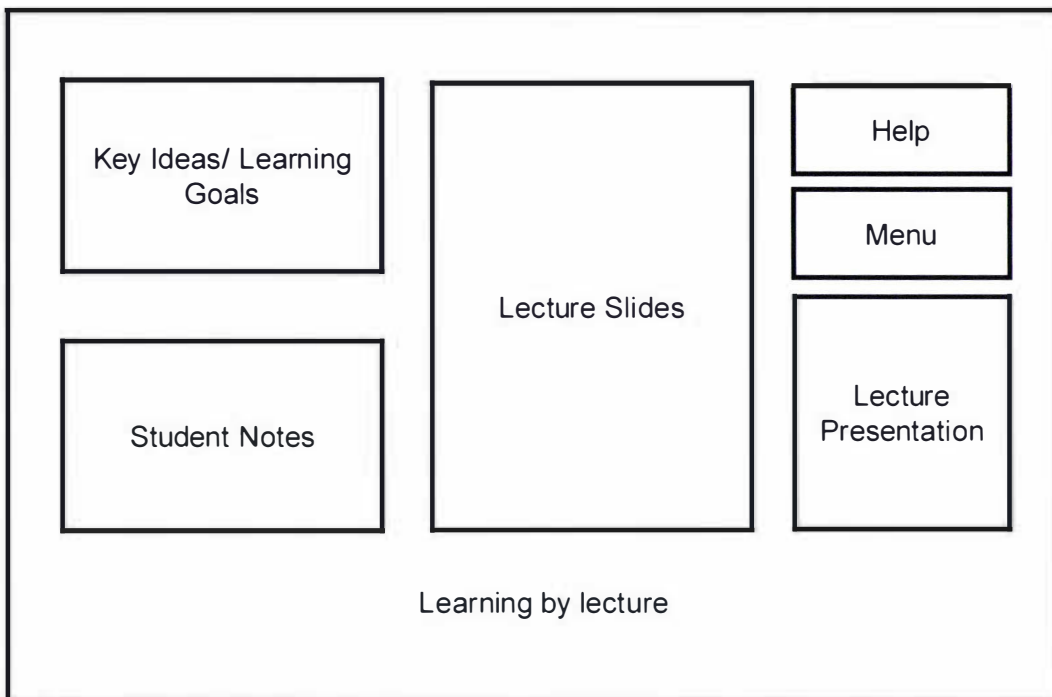
SQL-Tutor; SQT-Web	http://www.cosc.canterbury.ac.nz/~tanja/sql-tut.html
TILE	http://www.nzedsoft.com
TOP-CLASS	http://www.wbtsystems.com http://www.c2t2.ca/landonline/shownote.asp?appRow=13
UNIVERSAL	http://www.ist-universal.org/sommaire.htm
VINCENT	http://gaiva.inesc.pt/amp
WEBCT	http://www.webct.com ; http://www.c2t2.ca/landonline/shownote.asp?appRow=10 ; http://www.marshall.edu/it/cit/webct/compare/whyusewebct.html http://www.webct.ulpgc.es/papers/enable/paper.html http://www.cc.ubc.ca/ccandc/may-june96/webct.html
WITS	http://www.sis.port.ac.uk/conference/abstracts/94/dcallea.html
XAIDA	http://www.accd.edu/accd/workdev/ondemand.htm http://www.accd.edu/accd/workdev/download.htm http://www.ijoa.org/imta96/paper80.html
Comparisons / Overviews	
Brusilovsky, Peter. Adaptive and Intelligent Technologies for Web- based Education.	http://www.contrib.andrew.cmu.edu/~plb/papers/KI-review.html
Comparison of WebCT and CourseInfo	http://software2.bu.edu/webcentral/research/courseware
Landon, Bruce. Online educational delivery applications: a web tool for comparative analysis.	http://www.c2t2.ca/landonline/index.html

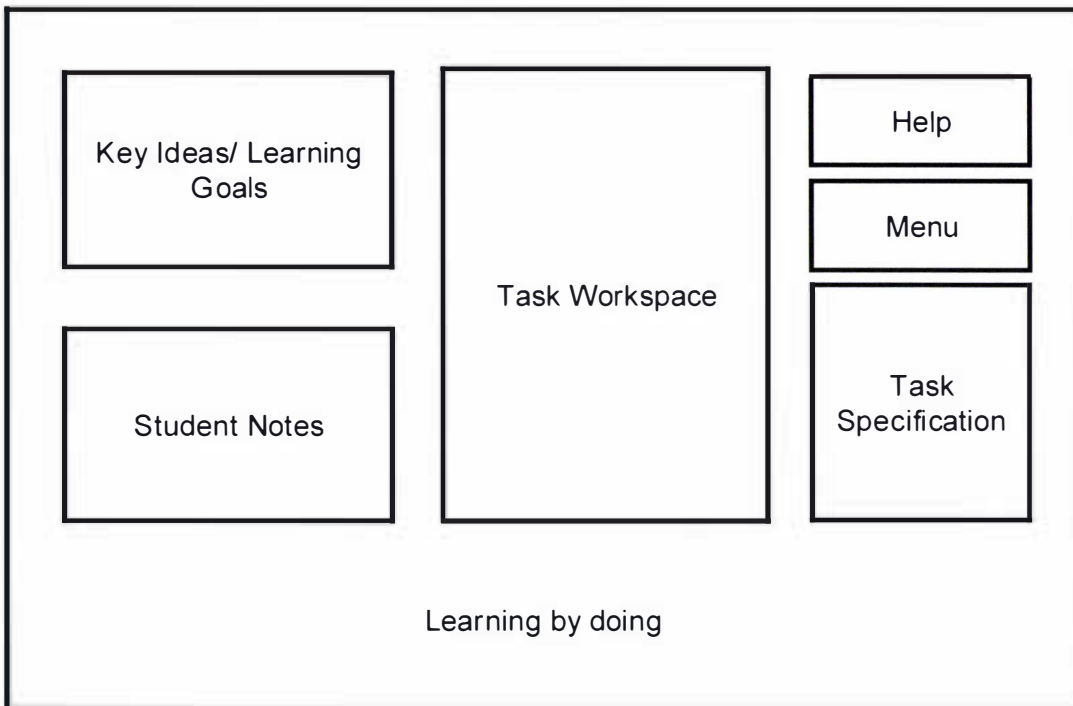
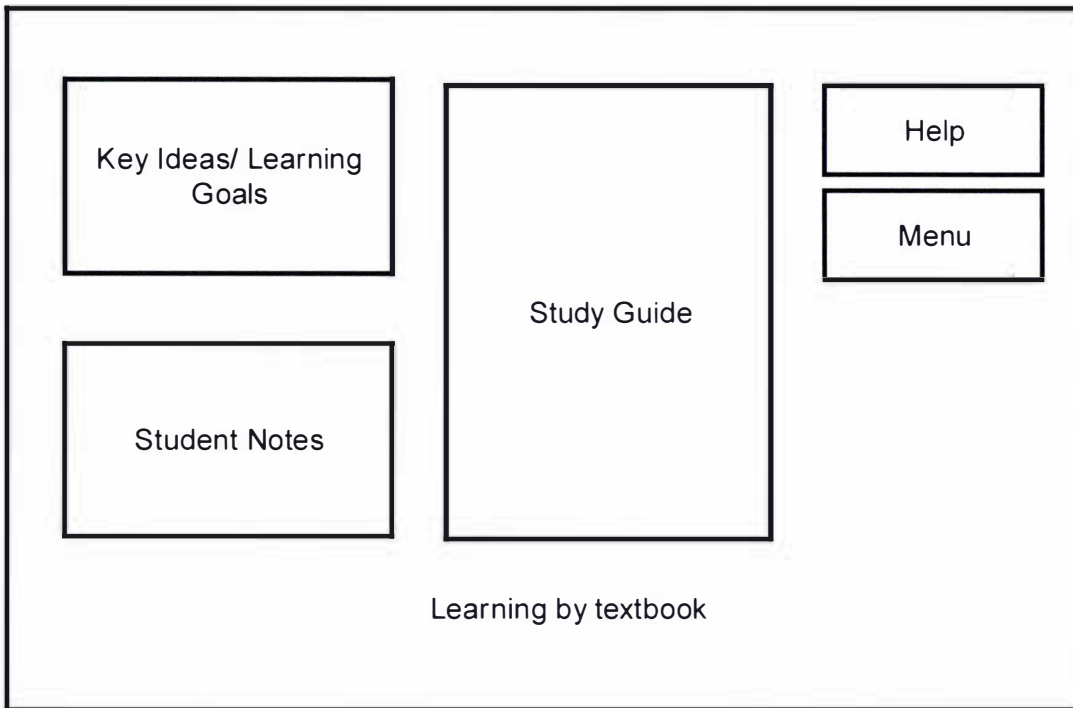
Marshall University: Comparison of Online Course Delivery Software Products	http://www.marshall.edu/it/cit/webct/compare
Murray, Tom. Authoring Intelligent Tutoring Systems: An analysis of the state of the art.	http://helios.hampshire.edu/~tjmCCS/papers/ATSummary/AuthTools.html
Prestamo, Anne. Putting your course online: A Comparison of Courseware Options	http://www.library.okstate.edu/dept/dls/prestamo/nom
Universities online	http://www.terra.com/specials/universities/index.htm

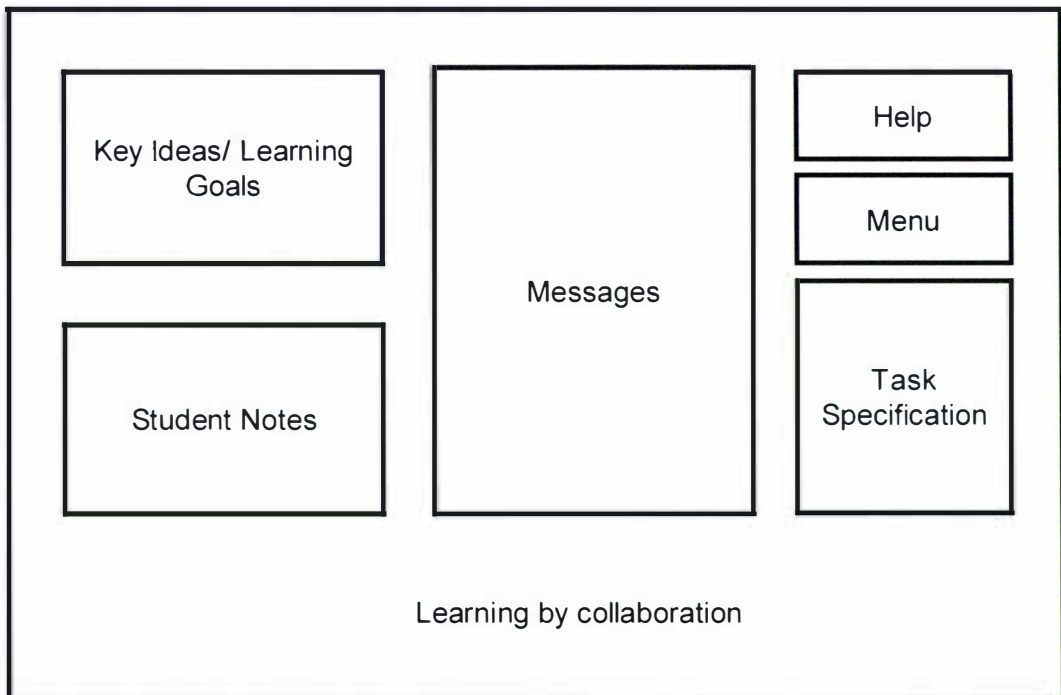
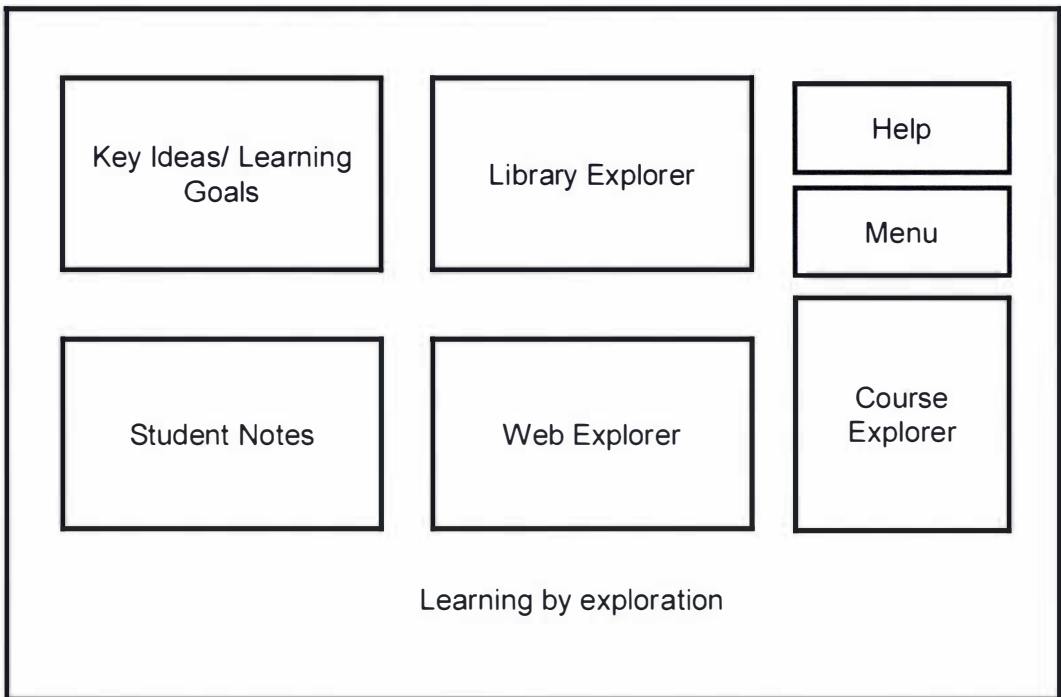
Appendix B

Conceptual view of learning elements and study modes



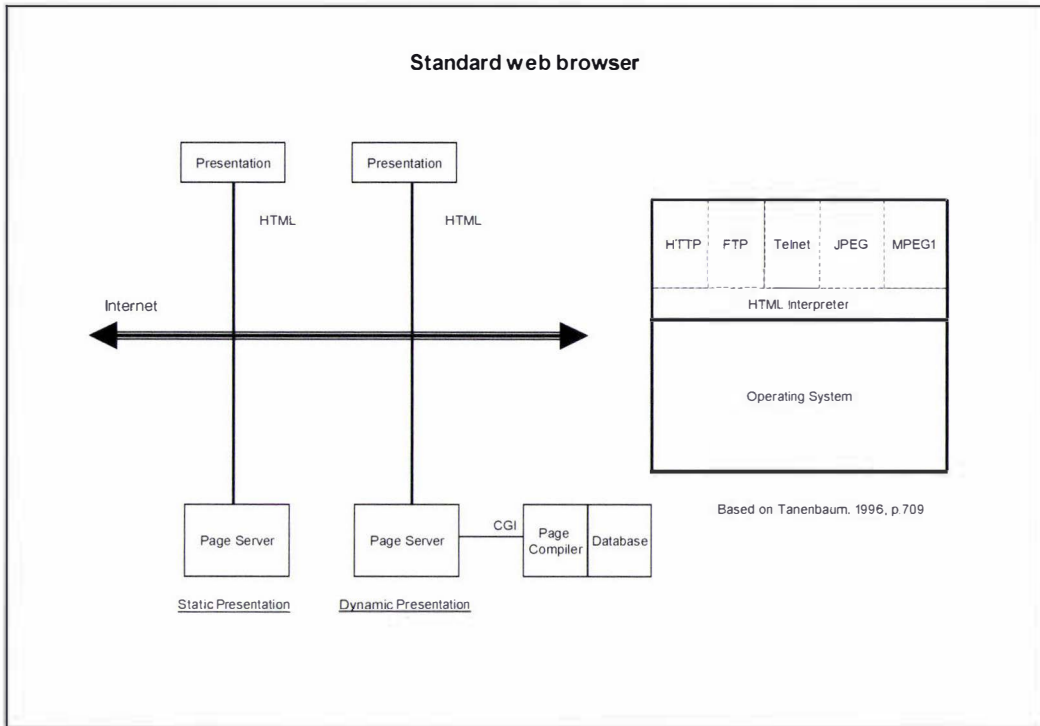




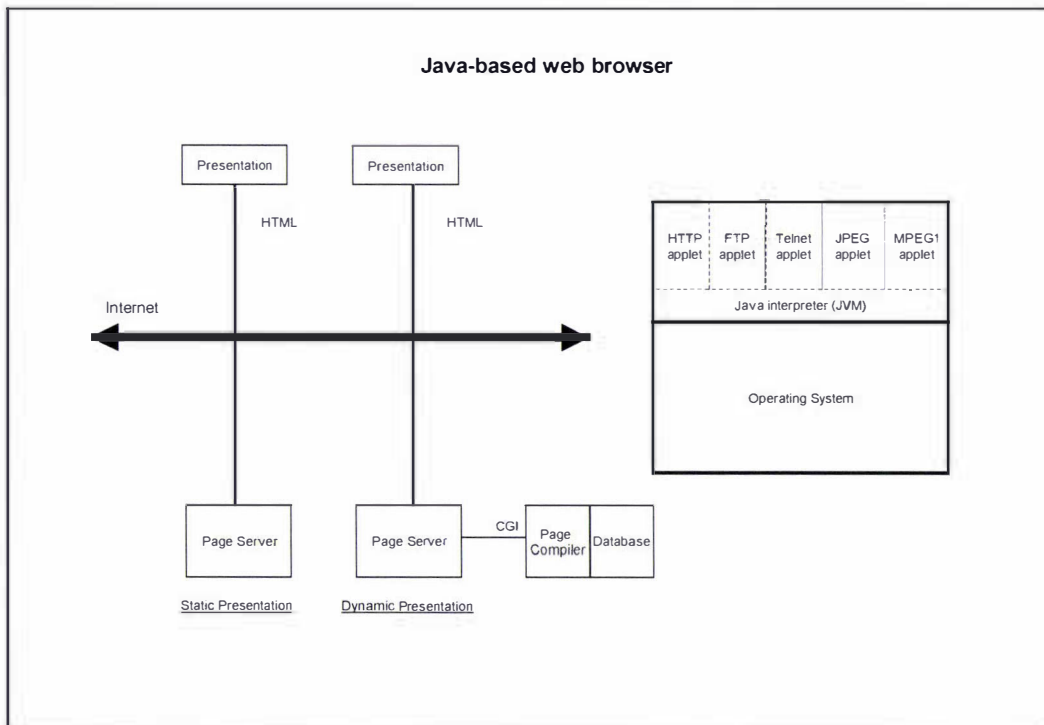


Appendix C

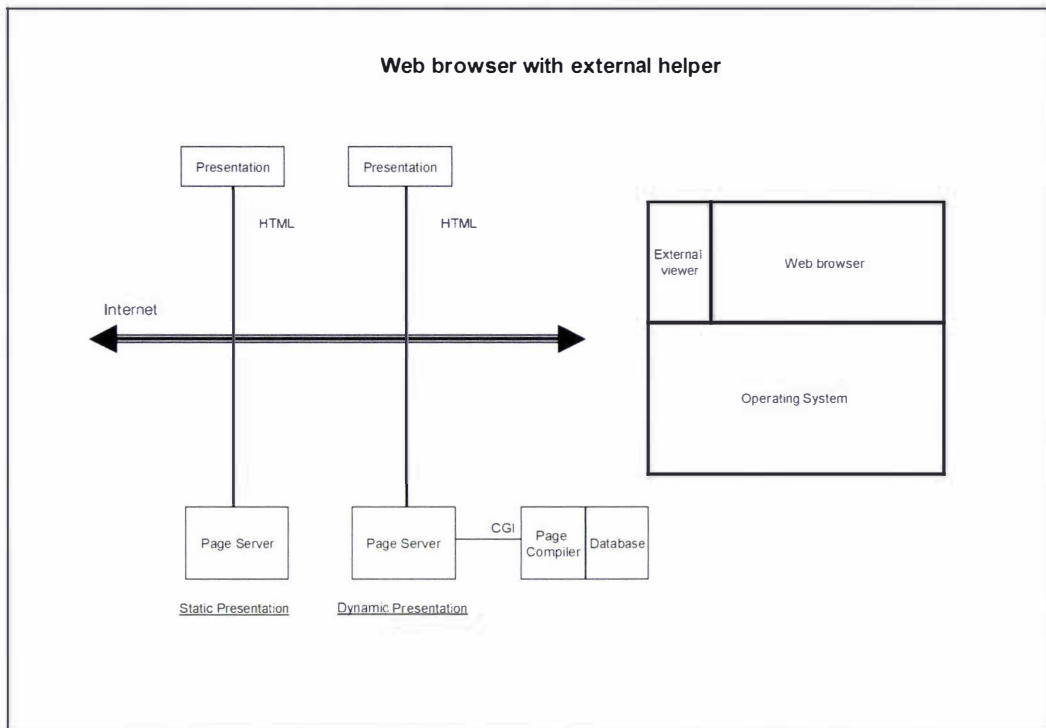
User Interfaces to Internet-based systems



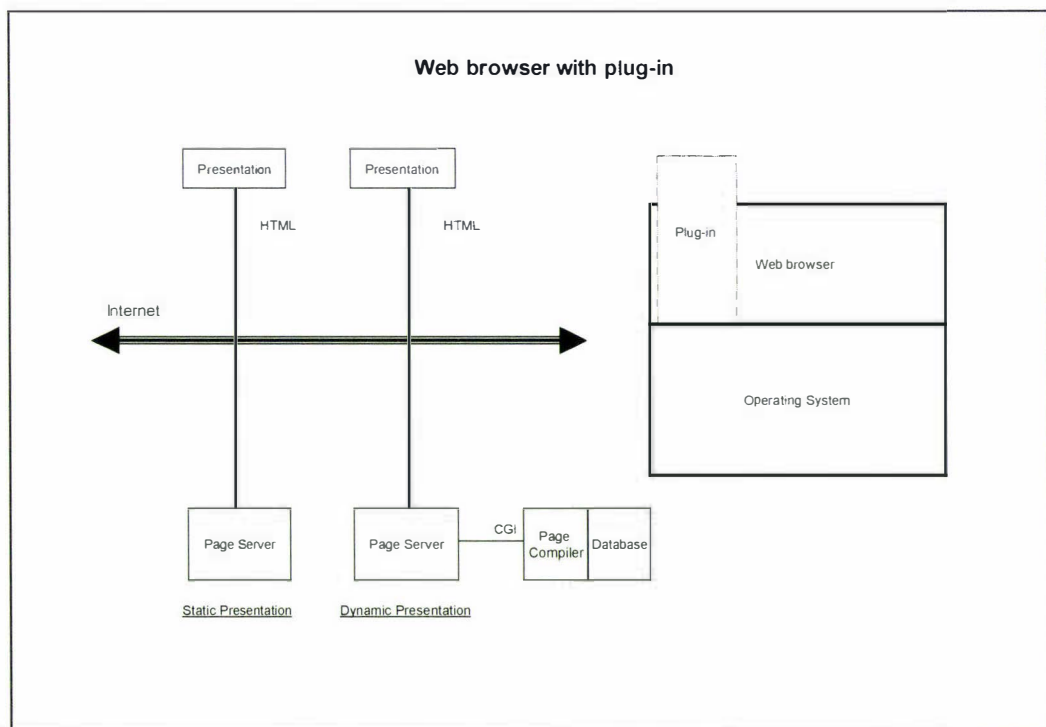
Standard web browser. Must be written for each platform. Presents static or dynamic web pages. Interactive functionality via CGI interface with application at server end. "Conceptually, a browser consists of a set of clients, a set of interpreters, and a controller that manages them" (Comer, 1999, p.427)



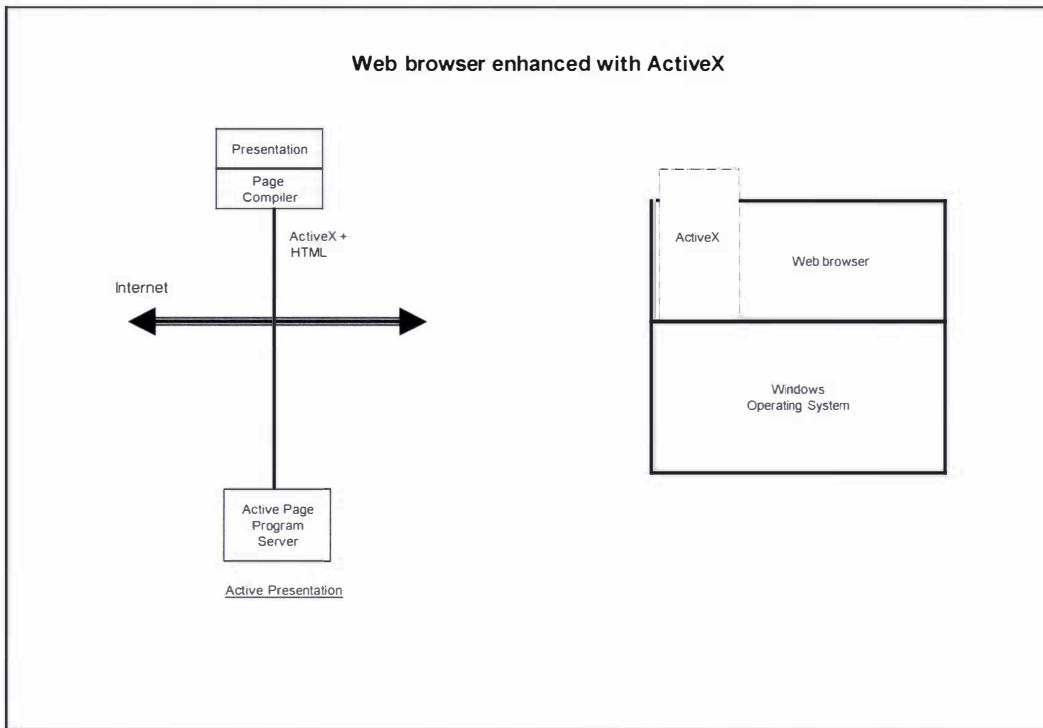
Java-based web browser. Multi-platform. "At startup, the browser is effectively an empty Java virtual machine...By loading HTML and HTTP applets, it becomes able to read standard Web pages. However, as new protocols and decoders are required, their classes are loaded dynamically, possibly over the network from sites specified in Web pages" (Tanenbaum, 1996, p.708)



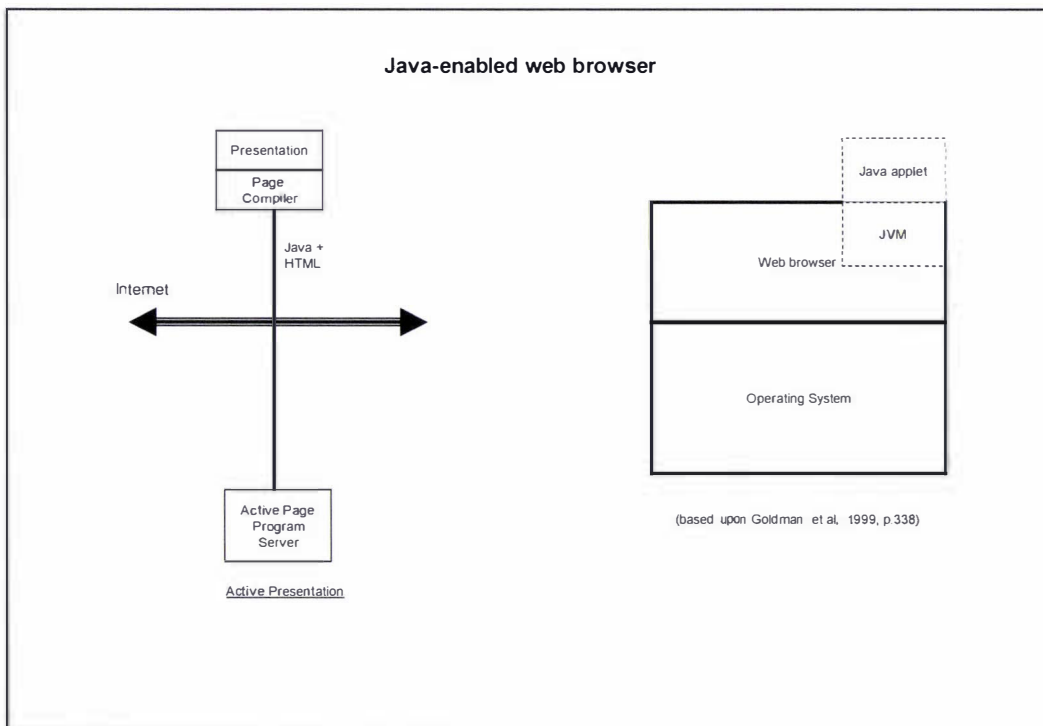
Helper application. Platform-specific. Standard browser passes new formats to external presentation programs installed on machine (Ellsworth et al, 1997, p.663).



Plug-in. Must be written for each platform. Standard browser "plugs-in" external modules for presenting new formats within browser. Plug-ins are downloaded from server as needed and are installed on machine and interface directly with operating system (Dewire, 1998, p.124).



ActiveX. Windows-specific. Multi-application. In contrast to Java applets, ActiveX controls directly interface with operating system enabling tasks like printing. (Goldman et al, 199, pp. 337-38). "Think of Active X controls as self-installing plug-ins for Windows-based systems" (Dewire, 1998, pp. 206-09).



Java applets. Multi-platform. Application-specific. "Each applet is embedded within a web page but, unlike ActiveX controls, can't access local files. Applets are not persistent - the applet is downloaded each time the Web page is... A JavaBean is analogous to an ActiveX control" (Dewire, 1998, pp.210-11).

Appendix D

Learning Shell Requirements Specifications

D1: Distance learner scenarios

1 Initialisation Scenario

1.1 Background

Mary is a young woman who has left a responsible job in town to live on a large sheep and cattle station near Pongaroa, where her partner has been hired as a shepherd. Unable to find full-time employment in the surrounding district, Mary decides to study for a business degree through Massey University's extramural programme.

Massey offers Mary the options of studying via traditional paper-based course material or via their new computer based distance learning system. The minimum requirements for the computer-based option is

- a personal computer with a CD-ROM drive, a modem and a printer, running Windows 98, including Internet Explorer,
- an Internet connection; and
- a DVD or VHS player linked to a TV set.

Students enrolling for the course are also required to submit a passport photo and a short biography

Mary, whose previous computer experience has been limited to some word processing, data entry and email correspondence in her last job, decides to purchase a computer and nominates the on-line option. She buys a second-hand computer off a neighbour. It is a PC about three years old running Windows 98. The neighbour deletes his personal files from "My Documents", but otherwise leaves things untouched. He warns her that, while he can send and receive short email messages with little problem, he often has to wait a long time, and even loses his connection, when sending or receiving emails with large attachments, or when trying to download materials off the Internet. He has heard that this is because of hot-wires running along farmers boundary fences close to the phone cable.

1.2 Installation

Shortly, Mary receives her course material in the mail from Massey. It consists of a course booklet, a CD-ROM and a video tape. She will also be required to obtain the prescribed text book for the course, which is available through Massey.

1.2.1 Booklet

The course booklet contains

- An introduction to the Virtual Learning Machine, outlining its various functions and features
- System requirements and installation instructions
- A hard-copy version of the course administration and study guide

1.2.2 CD-ROM

The CD-ROM contains files from which Mary can install the system software, the system file structure and course learning materials and resources.

1.2.3 Video

The video tape/disk steps through the installation process including

- Installing the system from the CD-ROM
- Optimising hardware and operating system settings for the Learning Computer
- Personalising the Learning Computer for a particular learner

1.2.4 Installing the Learning Computer

Mary places the video in her player and starts the machine. The video begins with an overview of the Learning Computer, demonstrating its various functions and features. Mary can refer to this again later if she so wishes. She then follows step-by-step the instructions for installing the system, including:

- Activating the executable on the CD-ROM which creates the Learning Computer file structure and decompresses and loads the learning resources into the correct locations
- Activating the set-up program on the CD-ROM which installs the Learning Computer software into the Learning Computer file structure

1.2.5 Optimising Windows

Next, Mary follows step-by-step the instructions for optimising hardware and operating system settings through the Control Panel, including:

- Setting up a user profile with the name that she will use when logging onto the Learning Computer
- Setting up an internet connection for this user profile which will start up automatically whenever required.
- Adjusting her monitor settings for optimal viewing of the Learning Computer

1.2.6 Personalising the Learning Computer

After rebooting her PC, Mary logs in to Windows using her learner profile, and double-clicks the Learning Computer icon to enter the learning system for the first time. The Windows desktop is replaced by the Learning Computer desktop. Mary is now presented with a login screen which welcomes her to Communication 101 and prompts her to enter a username and password. She enters the name and password created in her Windows profile, confirms her password by typing it again in a second box, and clicks the OK button (or hits enter).

Mary is now presented with a screen which requests additional information needed to personalise the Learning Computer to suit her preferences, including the drives that she will use to update course material, backup course material, and run lectures, presenting default settings that will work for the most common situation. The training video guides Mary through the process and she chooses her CD-ROM (D) drive for each task.

1.2.7 Starting the course

Mary clicks OK and she is presented with the Desktop Help screen providing an overview of the Learning Computer. After reading the overview Mary closes the screen revealing the Course Explorer component which displays

- The course title
- The course outline (sections and subsections (topics)) in the form of an expandable tree. Each node has a red, amber or green status icon representing not attempted, attempted, or completed respectively
- A box displaying the currently selected section and topic, its title and its status
- A list box displaying the modes of study available for that particular topic
- A Learning Computer Help icon

The Course Explorer is initialised to the start of the course. All nodes are set to the default status (not attempted).The Desktop also displays the course title; the current section, topic and topic title; and the default study mode (Text Book).

Mary clicks on the Help icon. A page opens outlining the features of the Course Explorer.

After perusing the features of the Course Explorer, Mary exits the Help and clicks the OK button on the Course Explorer. The Explorer component closes and the components of the Text Book Mode open in Section 1, Topic 1.

<End of initialisation scenario>

2 Start-up Scenario

2.1 Logon

Mary starts up her PC, logs on to Windows using her learner profile, and double-clicks the Learning Computer icon to enter the learning system. The Windows desktop is replaced by the Learning Computer desktop. Mary is now presented with a login screen which welcomes her to Communication 101, displays her username and prompts her to enter her password. She enters her password and clicks the OK button (or hits enter).

Mary has mistyped her password. A message notifies her of this and she is given the option of selecting CANCEL and quitting the Learning Computer, or OK, after which she can re-enter her password. Mary selects OK, enters her password correctly, and a screen appears offering Mary a number of options. She clicks on the Help icon and a page is displayed that explains each of these options thus:

- Return to previous topic The course will open in the study mode and topic that you were in when you last used the system
- Select new topic The Course Explorer will open from where you may choose to explore any available topic and study mode

- Update course material. You should select this option if you wish to update your course resources. You will be presented with the options of updating from a CD-ROM that has been mailed to you, or directly from the University via the Internet
- Restore from backup. You should select this option if your course materials have become corrupted. You will be presented with the options of updating your course material from a CD-ROM that has been mailed to you, or directly from the University via the Internet
- Update system settings. Select this option if you wish to change your personalised system settings, such as the drives you use for updating, backing up or accessing course materials.
- Exit. Exit the course.

Mary selects "Update Course Material". She is asked to nominate whether she will update from a disk or from the Internet. Mary chooses the update from disk option and she is asked to place the disk in the D drive. She does so clicks OK and she returns to the user options screen. (If she had chosen the update from the Internet option, she would only have to click OK and wait for the Learning Computer to connect to the course repository and download any updates to her machine. If no updates are available the system will notify Mary of this.)

If Mary had chosen the "Restore from backup" option a similar process would be followed.

Now Mary chooses the "Update System Settings" option. A screen opens presenting Mary with her current personalised system settings and options for changing them. She opts to change her backup drive to the floppy drive (A), clicks OK and is returned to the User Options screen again. This time she hits ENTER and the course opens at Section 1, Topic 2 in Text Book Mode, which is where Mary was working when she last exited the course.

<End of start-up scenario>

3 Exiting the Course Scenario

Mary has done enough for tonight. She right-clicks the Learning Computer desktop and a menu pops up. She selects the exit option. She is asked to confirm that she wishes to exit the system now and is given the options of YES or NO. She selects YES and is prompted whether she wishes to backup her work, with the same options. Again she selects YES, and she is asked whether she will backup to a disk or to the Repository. She selects the Repository. The system automatically attempts a modem connection to the University.

Shortly a message appears advising Mary that the connection has failed and the backup was not able to be completed. Mary selects the backup to disk option and she is prompted to place a disk in the A drive. Mary complies, the system saves any notes and other items edited by Mary to the floppy disk, the Learning Computer closes and she is returned to the Windows desktop.

Alternatively, Mary simply switches off her machine, and goes off to watch television.

<End of exit course scenario>

4 Exploring the course Scenario

While working in Text Book Mode Mary right clicks the desktop and a menu pops up. She selects the EXPLORER option and the Course Explorer object opens displaying:

- Communication 101
- The course outline in the form of an expandable tree. Communication 101 has an amber icon next to it indicating the course has been previously attempted.
- Section 1: Modern Poets; Topic 2: Pete Brown; Status: Attempted.
- A list of study options: Text Book, Lectures, Exploration, Group Work.
- AVL M Help icon.

Mary clicks on Communication 101 in the contents tree and the section titles appear below it. Modern Poets has an amber icon by it. All the others have red icons indicating that they have not yet been attempted. She now clicks on Modern Poets and section one's subtopics now appear. Bob Dylan has a green icon beside it indicating that Topic 1 has been successfully completed. Pete Brown has an amber one.

Mary decides to explore the course contents. She clicks on 2. Investigative Reporters (which is then displayed in the current topic box). And then on "2.1 John Pilger" which appears below it with a red icon. The current topic box displays " Section 2.1; John Pilger; Not attempted." The list of study options now reads: Text Book, Lectures, Tutorial, Exploration, Group Work, Do It Yourself.

4.1 **Lecture Mode**

Mary decides to explore the various study options. She double clicks **Lectures**. An image of the course lecturer appears and a window displaying his/her lecture notes in the form of slides (or as a Word or web document). By clicking on the lecturer's image the lecture is loaded and a set of controls are activated. Mary may use the controls to listen to the lecture at her leisure, and following through the notes. (Alternatively, a video presentation may be loaded which Mary can play in conjunction with the notes.)

4.2 **Tutorial**

Mary selects **Tutorial** and then OK. A screen opens presenting excerpts and analysis of Pilger's East Timor documentary.

4.3 **Exploration**

When Mary selects this learning option she is presented with two options:

- Explore the Web; or
- Explore the Library.

Mary selects "Explore the Web" and a screen opens for viewing web pages, containing a web references list, including the additional references she located through the Help System, and a workspace for Mary to enter notes. She chooses "John Pilger's home page" and clicks GO. Mary finds the page unhelpful so she clicks DELETE and the link is deleted from the web references list. She also had the options of

printing the page, adding a new link, or saving her list. On exiting Mary is given the opportunity of saving her list. Any notes she has made are appended to her Student Notes.

Mary then selects "Explore the Library" and a screen opens with a link to the Massey Library system. The recommended reading list for this topic is displayed, along with any titles she located through the Help System. Mary has the options of adding or deleting titles from the book list, printing or saving it.

4.4 Do It Yourself

The current assignment is displayed, plus a space in which Mary can work on it

4.5 Group Work

Mary was assigned to a work group when she enrolled for the course. Here she can see details about each member of her group. She can read all their contributions to the discussion on this topic, plus that of the tutor and general points and queries made by students from other groups; and reply, forward or add contributions of her own. She can access notes on her current assignment if she wishes to discuss these with others in her group.

4.6 Text Book

Mary is able to browse the study guide for Section 2.1 which guides her through relevant sections of the course text book. She also has in front of her a list of the key concepts (learning goals) for this topic. Each of the concepts is pre-fixed by a red 'x', indicating that Mary has not yet demonstrated that she has mastered these concepts.

5 Support Tools Scenario

With the right mouse button Mary clicks the desktop. A menu appears offering the following options:

- Topic => Next
- Topic => Previous
- Course Explorer
- Learning Support
- Course administration
- Help
- Exit

Course Administration opens a document which provides the administrative overview of the course. Help opens the Desktop Help page.

Mary selects the Learning Support option and she is presented with a list of all the learning components which are available in any study mode. The include:

- Student Notes
- Key Concepts
- Self-Evaluation

Mary selects Student Notes and an editable document opens containing all the notes she has made in any mode on the current topic. Mary then selects Self-Evaluation and she is invited to perform a short self evaluation. This may be either:

- A multiple-choice quiz on the concepts covered in the topic; or
- A questionnaire on the concepts covered in the topic.

The results of this self evaluation is used by the system to determine what sections and topic in the course Mary has completed, or is working on, and what concepts/learning goals she has mastered, or still needs to work on.

<End of exploring course scenario>

6 Help System Scenario

While using the Learning Computer, Mary is able to access to kinds of help from any where in the system:

- Help with using Learning Computer components
- Help with mastering course concepts and learning goals

6.1 *Component Help*

Every Learning Computer component has a Learning Computer icon. By clicking on this icon, (or selecting **F1**) Mary is able to open a help page specific to that component which:

- Describes the main functions and features of the component
- Provides tips on using the component including any shortcuts

6.2 *Concept Help*

A Key Ideas component may be accessed from any learning mode via the desktop menu. By double-clicking a concept listed in this component (or selecting it and pressing **F1**) the Help System opens displaying the selected concept and offering Mary several alternative queries she may select, including:

- Explain <SELECTED CONCEPT>? The system will search its database for its best explanation of the query.
- Where do I find more on <SELECTED CONCEPT>? The system will provide its best web reference.
- Who wrote more on <SELECTED CONCEPT>? The system will provide its best book, journal or paper reference.

Mary then selects OK and she is offered some or all of the following options as appropriate:

- OK - Mary's query has been answered to her satisfaction
- Elaborate further (find further answers)
- No - no helpful answer received (The course tutor is automatically and anonymously notified when this option is selected so he/she may review the Help database)

- View concept map - Mary can view a map of related concepts and choose one that may help clarify her query. The Help system will then be reinitialised to the new <SELECTED CONCEPT>.
- Ask - a message form is opened with which Mary can send her query directly to the course tutor
- Copy - Mary may copy the Help System's responses to her query to the appropriate learning component - the Student Notes, Web Explorer or Library Explorer. [Alternatively this could be done automatically on ending a query.

<End of Help scenario>

D2: Use Cases

1 Startup Use Case

1.1 Brief description

This use case is initiated by the system, and provides for the student to log onto the system and begin studying.

1.2 Flow of events for use case

1.2.1 Preconditions

The student has logged onto the computer system and opened the learning application.

1.2.2 Main Flow

1. The system displays the user's name and requests a password (A). The password is entered. The system validates the password (B).
2. User is presented with startup options of returning to previous position in course, selecting new topic (C), providing additional information (I), updating course material (II), restoring course resources from backup(III) or exiting the system. Student selects the default option (previous position). The system opens in the learning mode and course position that the student was in when they previously exited the system. The Desktop displays the current course title, section, topic and study mode.

1.2.3 Subflows

- I Additional Information. The system requests the user provides any additional information necessary to configure the system for that particular user. Once the information has been provided, the system returns to (2).
- II Updating course material. The user is presented with the options of updating via the Internet or updating via removable disk. Once the the update has been completed, the system returns to (2).
- III Restoring course resources. The user is presented with the options of restoring the resources via the Internet or via removable disk. Once the the restoration has been completed, the system returns to (2).

1.2.4 Alternative Flows

- A The user is entering the system for the first time. The system requests that the user enters a name and password. The user enters their name and password. The system requests the user confirms their details (B). The Select New Topic Use case is initiated and the current use case closes.
- B The incorrect password is entered. (1) is repeated until the correct password is entered or the student selects close option and the system closes and the use case ends.
- C The student selects the new topic option. This initiates the Select New Topic use case.

<end of start-up use case>

2 Shutdown Use Case

2.1 Brief description

The system initiates this use case to complete housekeeping tasks before the user closes the system.

2.2 Flow of events for use case

2.2.1 Preconditions

The user has logged onto and begun using the system.

2.2.2 Main Flow

1. The user selects the EXIT option from the course menu.
2. The system requests that the user confirm they wish to quit the system. The user confirms.(A) If the course resources have been modified since the last backup, the user is notified of this and asked whether they wish to backup course material. The user confirms (I). (B).

2.2.3 Subflows

- I. Backup. The user is presented with the options of backing up via the Internet or to another local drive. The backup is completed and the system closes, ending the use case

2.2.4 Alternative Flows

- A. The user cancels the shutdown and the system resumes, ending the use case.
- B. The user declines to back up and the system closes, ending the use case.

<end of shutdown use case>

3 Change Mode Use Case

3.1 Brief description

This use case provides the student with the capability of changing their mode of study from anywhere within the course.

3.2 Flow of events for use case

3.2.1 Preconditions

The student is logged onto the system. The use case begins when the Student selects a learning resource in the Course Explorer.

3.2.2 Main Flow

1. The System Model is updated.
2. All components unique to the current mode are closed.
3. The appropriate resources are attached to all available components unique to the new mode.
4. Each unique component is opened.
5. The Student Model is updated.
6. The Desktop displays the new mode, section and topic.
7. The Explorer closes.

3.2.3 Subflows

3.2.4 Alternative Flows

<end of change mode use topic>

4 Change Topic Use Case

4.1 Brief description

This use case provides the capability for changing a topic within the same study mode

4.2 Flow of events for use case

4.2.1 Preconditions

The user has successfully logged onto the system, and has selected a new topic from within the Course Explorer.

4.2.2 Main Flow

8. The appropriate resources are attached to all open components.
9. The Student Model is updated.
10. The Desktop displays the current mode, section and topic.
11. The Explorer closes and the use case ends.

4.2.3 Subflows

4.2.4 Alternative Flows

<End of change topic use case>

5 Next Topic Use Case

5.1 *Brief description*

Enables the use to move to the next topic from anywhere within the course.

5.2 *Flow of events for use case*

5.2.1 Preconditions

The user has selected the "Next Topic" option from within the system menu.

5.2.2 Main Flow

12. The appropriate resources are attached to all open components.

13. The Student Model is updated.

14. The Desktop displays the current mode, section and topic.

5.2.3 Subflows

5.2.4 Alternative Flows

<End of next topic use case>

6 Select New Topic Use Case

6.1 *Brief description*

Provides for the user to select a new topic from anywhere within the course.

6.2 *Flow of events for use case*

6.2.1 Preconditions

The user has selected the "Select New Topic" option on logging on or has selected the "Course Explorer" option in the system menu.

6.2.2 Main Flow

1. The Course Explorer opens, displaying an outline of the course content in the form of a hierarchical tree (Section, Topic). The current position (topic and mode) in the course is highlighted and the learning resources available for that topic are displayed. Each tree node (Section, Topic) is colour-coded to indicate whether the student has not commenced studying, commenced studying, or completed a particular section or topic

2. The user selects a new topic and the learning resources available for that topic are displayed in the Explorer.

3. The system presents the student with several options: OPEN, CANCEL, or REVISE. (A).
 - If the OPEN option is selected then (i)
 - If CANCEL then the Course Explorer closes.
 - If REVISE then (ii).

6.2.3 Subflows

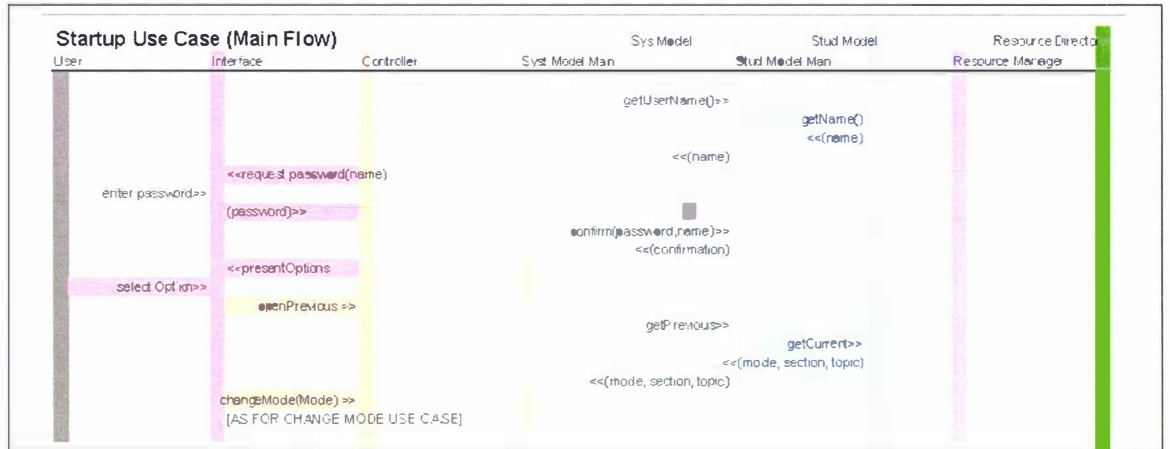
- (i) If the user has selected a learning resource then the Change Mode use case begins. Otherwise the Change Topic use case begins. The current use case closes.
- (ii) The system asks the user to confirm they wish to study in revision mode. The student confirms. The system tree is re-initialised. All nodes display the "Not commenced studying" code. (B).

6.2.4 Alternative Flows

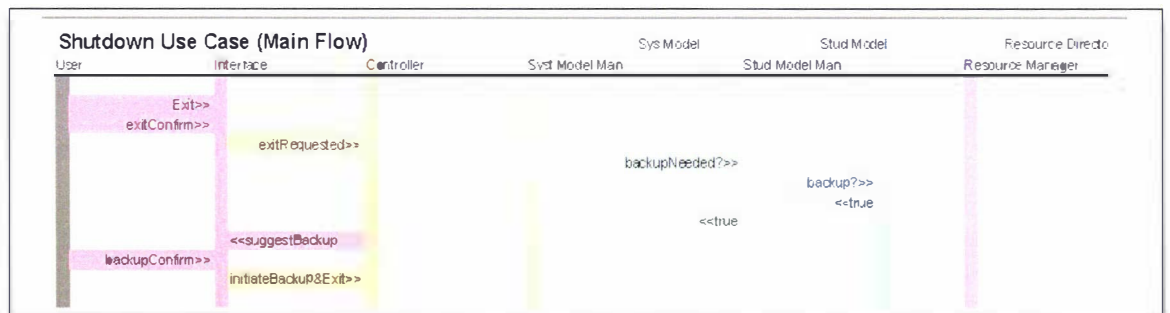
- A The user selects a new topic and (3) recommences.
- B The student cancels, and (3) recommences.

<end of select new topic use case>

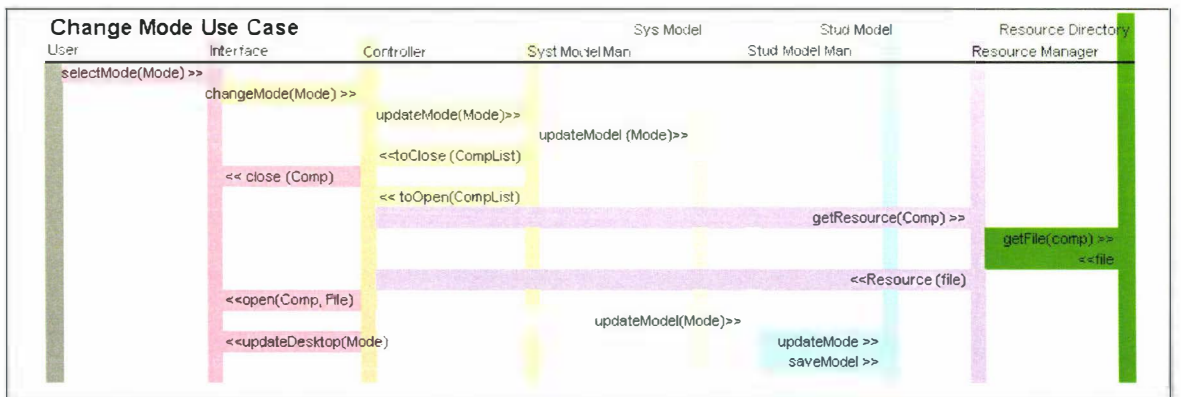
D3: Sequence Diagrams



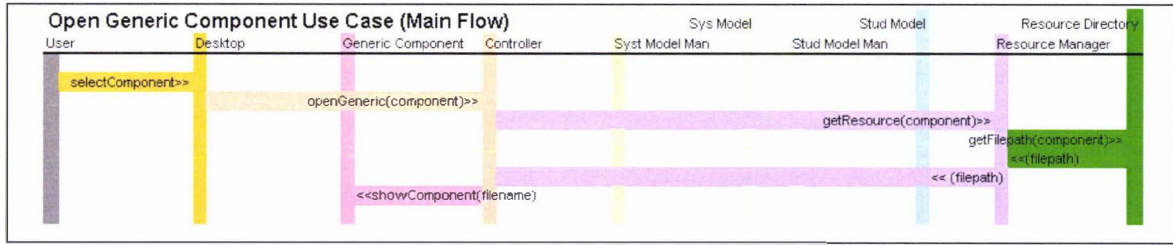
Start-up Use Case



Shutdown Use Case



Change Mode Use Case



Open Generic Component Use Case

D4: Extramural Support: Query Specification

Query Type	Precondition	User Action	System Response	Implementation	+/-
What	The Extramural Support screen is displayed initialised to a concept selected by the user.	Selects 'Explain' from a combo box	Displays an explanation of the selected concept in a memo box, or an appropriate message if none is available.	An elaboration of every key concept/learning goal in any topic is stored in a database by section & topic. This is located and returned using an SQL query. Provision is made for storing additional elaboration on the concept in this or other topics, and for adding other concepts in addition to the key ones.	Locates information in the database most relevant to the current topic.
Who	The Extramural Support screen is displayed initialised to a concept selected by the user.	Selects 'Who wrote more on' from a combo box.	Displays a book or paper reference on the selected concept in a memo box, or an appropriate message if none is available.	Book and paper references are stored in a database by concept, section and topic. The first of the result set from an SQL query by the current concept, section and topic is returned.	
Where	The Extramural Support screen is displayed initialised to a concept selected by the user.	Selects 'Where do I find more on' from a combo box.	Displays a web site relevant to the selected concept in a memo box, or an appropriate message if none is available.	Web references are stored in a database by concept, section and topic. The first of the result set from an SQL query by the current concept, section and topic is returned.	

Query Type	Precondition	User Action	System Response	Implementation	+/-
Why	The Extramural Support screen is displayed and the user has already executed a What, Who or Where query.	Clicks the 'Retry' button on the help screen.	Displays further information on the selected concept, i.e. a further elaboration of the concept, or an additional book or web reference, or an appropriate message if none is available.	Returns the next result in the result set from the previous query, or if there is no more, returns the next result from an SQL query where the topic precedes the current topic, or if there is no more, returns the next result from an SQL query where the topic comes after the current topic.	

D5: Extramural Support: Dialogue Specification

User selects:	System responds:	Next button options
OK (to action a new query)	Presents user with the best result in the query result set.	OK, NO, RETRY
OK (to accept result of query)	None	EXIT, MAP, COPY, NEW
NO	The tutor is notified that the help offered on this concept was found unsatisfactory.	If nor more results found in database: ASK, EXIT, MAP, COPY, NEW; RETRY, ASK, EXIT, COPY, NEW
RETRY	The next best result in the query result set is returned	If nor more results found in database: NO, ASK, EXIT, MAP, COPY, NEW else: OK, NO, RETRY
ASK	A new message window appears, initialised to the tutor.	EXIT, MAP, COPY, NEW
MAP	A window opens displaying a concept map showing all related concepts in the System Database. Any concept may be selected to launch a new query.	ASK, EXIT, MAP, COPY, NEW
COPY	The results of the query, which may be explanatory text, book references or URLs, are copied to the appropriate component's data files, via the controller.	EXIT, NEW
NEW	The Extramural Support screen is cleared and reinitialised to the current concept	OK, EXIT
EXIT	The Extramural Support window closes	---

Appendix E

Learning Shell prototyping – classes and components¹

¹ This appendix includes documentation from the *process* of prototyping the Learning Shell's classes and components. Only the Object Hierarchy (E5) represents the final, complete, refined design.

E1: Prototyping basic component types

Method	Description
ActiveX(OLE) 1	Use a non-visual "drag-and-drop" component to launch an application, e.g. WORD, in background and utilise it to provide services (e.g. printing of a learning resource) not provided by the interface component.
ActiveX (OLE) 2	Embed a Delphi-independent control directly in the interface to provide ready-made, customisable functionality (e.g. an HTML browser without all the buttons and menus).
.EXE	Use a non-visual "drag-and-drop" component to launch an application in the foreground to provide additional functionality not directly provided by the system, e.g. a locally-installed or CD_ROM-based tutorial or dictionary.
Custom-built web application launcher	Build a non-visual "drag-and-drop" component to launch a web-based application in the foreground to provide additional functionality not directly provided by the system, e.g. a web-based tutorial or search engine.
Custom-built systems utility	Build a non-visual "drag-and-drop" component to provide interface components with services that would otherwise require complex direct operating system calls, e.g. file or folder manipulation routines.
Customised Delphi control	Customise IDE-provided controls , e.g. Rich Text editor, to provide just-enough functionality
Form inheritance	Build a generic form (e.g. HTML viewer) and install it in the Delphi Repository as a component which can be inherited and customised for different interface components (e.g. study guide or lecture notes).
Frame component	Design a form layout and install it as a "drag-and-drop" component which can be used to give different forms (screens) the same look and feel, but different functionality (e.g. Desktop with or without pop-up menus.)
Multi-form component	Build a form that provides the interface for functionality that is implemented by multiple forms, which the master form creates and destroys as required. The rest of the system has no knowledge of these child forms.

E2: Component dependencies

Object	Creates	Launches	Via Controller	HelpBtn
Application	Desktop, database module	Desktop		
Desktop	Controller, CourseExplorer, StartUpDesktop, UserOptions, ResourceMenu, SystemLog, ExtramuralSupport, All learning components	StartupDesktop, ResourceMenu, Course Explorer, SystemLog		√
Controller	systemModelManager, studentModelManager, resourceManager, systemTreeManager, UpdateExtramuralSupport	All learning components, UserOptions, Desktop, ExtramuralSupport, CourseExplorer (viaDesktop)		
StartUpDesktop			UserOptions	
CourseExplorer			All mode-based learning components	√
ResourceMenu			All generic learning components	
SystemLog				
UserOptions	Logon, UpdateResources, RestoreResources, BackupResources, MoreInfo	Logon, UpdateResources, RestoreResources, BackupResources, MoreInfo	Explorer, Desktop	√
Logon				√
UpdateResources				√
RestoreResources				√
BackupResources				√
MoreInfo				√
UpdateExtramuralSupport				
SystemModelManager	SystemModel			
SystemModel				
ResourceManager	RootDirectory (ResourceModel)			
RootDirectory (ResourceModel)				
SystemTreeManager	SystemTree			
SystemTree				
StudentModelManager	StudentModel			

Object	Creates	Launches	Via Controller	HelpBtn
StudentModel				
Tutorial	WebTutor	WebTutor; [locally-installed tutorial executable]		√
Dolt	WebTutor	WebTutor; [locally-installed executable]		√
WebTutor				√
Lecture				√
LectureNotes				√
StudentNotes				√
SelfAssess	Quiz, questionnaire	Quiz, questionnaire		√
Quiz				
Questionnaire				
StudyGuide				√
Assignment				√
AssignmentWorkSpace				√
WebExplorer	ExploreTheWeb	ExploreTheWeb		√
ExploreTheWeb				√
LibraryExplorer	ExploreTheLibrary	ExploreTheLibrary		√
ExploreTheLibrary				√
KeyIdeas			ExtramuralSupport	√
ExtramuralSupport	ConceptMap, AskForHelpMessage	ConceptMap, AskForHelpMessage		√
ConceptMap				√
AskForHelpMessage				√
DrawIt				√
Feedback				√
AdministrationGuide				√
PracticeAssignment				√
MessageList	MessageSystem, NewMessage, ReplyMessage, ForwardMessage, AllNewMessages,	MessageSystem, NewMessage, ReplyMessage, ForwardMessage, AllNewMessages,		√
MessageSystem	NewMessage, ReplyMessage, ForwardMessage	NewMessage, ReplyMessage, ForwardMessage		√
ForwardMessage				√
NewMessage				√
ReplyMessage				√

Object	Creates	Launches	Via Controller	HelpBtn
AllNewMessages	ReplyMessage, ForwardMessage	ReplyMessage, ForwardMessage		√

E3: Inheritance hierarchy

Class	Inherits from	Owned By	Uses	Properties	Methods
IvImObject	Interface				
TvImClass	TObject		vlaSystemDictionary, vlaSystemUtilities	Debug:boolean	SetDebugging(Abstract)
TModelManager	TvImClass	TheController	VlaController, vlaSystemDictionary, vlaSystemUtilities		Create (create model); SetDebugging (to Controller.debug)
TModel	TvImClass	AModelManager	<vlaAModelManager>, vlaSystemDictionary, vlaSystemUtilities		SetDebugging (to Amodelmanager.debug)
TComponent	TObject				
TvlaComponent	TComponent		vlaSystemDictionary, vlaSystemUtilities		CreateLearning Component (abstract)
TvlaBaseComponent	TvlaComponent		vlaSystemDictionary, vlaSystemUtilities	Component:string	
TForm	TWinControl(TComponent)				
TvImForm	TForm		vlaSystemDictionary, vlaSystemUtilities	Debug:boolean	Setcolour(<i>Abstract</i>); FormShow(parse caption, FormPaint (setcolour); SetDebugging(<i>Abstract</i>);
TSystemForm	TvImForm	TheController	VlaController, vlaSystemDictionary, vlaSystemUtilities		Setcolour (to systemcolour);
TComponentForm	TvImForm		vlaSystemDictionary, vlaSystemUtilities	HelpBtn:TButton	HelpBtnClick (show help);

Class	Inherits from	Owned By	Uses	Properties	Methods
TMainComponentForm	TComponentForm	TheController	VlaController, vlaSystemDictionary, vlaSystemUtilities		SetDebug (to Controller.debug)
TSecondaryComponentForm	TComponentForm	<AMainComponent>	<vlaAMainComponent> vlaSystemDictionary, vlaSystemUtilities		Setcolour (to owner's); SetDebugging (to AmainComponent.debug)
TMainLearningComponentForm	TMainComponentForm	TheController	VlaController vlaSystemDictionary, vlaSystemUtilities		Setcolour (to mode); LoadResource; FormShow(+loadResource);
TMainSystemComponentForm	TMainComponentForm	TheController	VlaController, vlaSystemDictionary, vlaSystemUtilities		Setcolour (to system colour); SetDebugging (to Controller.debug)

E4: Class Hierarchy

Class	Is_a	Has_a	Uses	Properties	Methods
IvImObject	Interface				
TvImClass	TInterfacedObject, IvImObject			Debug:boolean	SetDebugging(Abstract)
TModelManager	TvImClass		TController		Create (create model); SetDebugging (to Controller.debug)
TModel	TvImClass		(TModelManager)?		SetDebugging (to Amodelmanager.debug)
TComponent	TObject				
TvlaComponent	TComponent				CreateLearningComponent (abstract)
TvlaBaseComponent	TvlaComponent			Component:string	
TForm	TWinControl(Tcomponent)				

Class	Is_a	Has_a	Uses	Properties	Methods
TvImForm	TForm			Debug:boolean	Setcolour(<i>Abstract</i>); FormShow(parse caption, FormPaint(setcolour); SetDebugging(<i>Abstract</i>);
TSystemForm	TvImForm		TController		Setcolour (to systemcolour);
TComponentForm	TvImForm			HelpBtn:TButton	HelpBtnClick (show help);
TMainComponentForm	TComponentForm		TController		SetDebug (to Controller.debug)
TSecondaryComponentForm	TComponentForm	TMainComponentForm			Setcolour (to owner's); SetDebugging (to AmainComponent.debug)
TMainLearningComponentForm	TMainComponentForm		TController		Setcolour (to mode); LoadResource; FormShow(+loadResource);
TMainSystemComponentForm	TMainComponentForm		TController		Setcolour (to system colour); SetDebugging (to Controller.debug)
TDesktop	StartupDesktop	CourseExplorer, StartupDesktop, UserOptions, ResourceMenu, SystemLog, ExtramuralSupport, All learning components			
TController	TvImClass	TDesktop SystemMode IManager, StudentModelManager, ResourceManager, SystemTree Manager, UpdateExtramuralSupport	TCompList		All learning components, UserOptions, Desktop, ExtramuralSupport, CourseExplorer (viaDesktop)

Class	Is_a	Has_a	Uses	Properties	Methods
TStartUpDesktop	TvImForm		TController		
TCourseExplorer	TMainSystemComponentForm		TController		
TResourceMenu	TSystemForm		TController		
TSystemLog	TSystemForm		TController		
TUserOptions	TMainSystemComponentForm	TLogon, TUpdateResources, TRestoreResources, TBackupResources, TMoreInfo	TController		
TLogon	TSecondaryComponentForm				
TResourceTransfer	TSecondaryComponentForm	TvIaFolderManager, TvIaFileTransfer, TnmFTP, TImdStarter			
TUpdateResources	TResourceTransfer				
TRestoreResources	TResourceTransfer				
TBackupResources	TResourceTransfer				
TMoreInfo	TSecondaryComponentForm				
TSystemModelManager	TModelManager	TSystemModel			
TSystemModel	TModel				
TResourceManager	TModelManager	TRootDirectory			
TRootDirectory (ResourceModel)	TModel				
TSystemTreeManager	TModelManager	TSystemTree			
TSystemTree	TModel				
TStudentModelManager	TModelManager	TStudentModel			
TStudentModel	TModel				
TLauncher	TMainLearningComponentForm	TvIaTutor TMLDStarter			
TvIaTutor	TComponent				
TPlayer	TMainLearningComponentForm				

Class	Is_a	Has_a	Uses	Properties	Methods
TViewer	TMainLearningComponentForm				
TEditor	TMainLearningComponentForm				
TSelfAssess	TMainLearningComponentForm	TQuiz, TQuestionnaire			
TQuiz	TvImClass				
TQuestionnaire	TvImClass				
TAssignmentWorkSpace	TEditor	TvIaFileManager			
TWebLauncher	TMainLearningComponentForm	TWebResourceBrowser			
TWebExplorer	TWebLauncher	TExploreTheWeb			
TWebResourceBrowser	TSecondaryComponentForm				
TExploreTheWeb	TWebResourceBrowser				
TLibraryExplorer	TWebLauncher	TExploreTheLibrary			
TExploreTheLibrary	TWebResourceBrowser				
TKeyIdeas	TMainLearningComponentForm		Extramural Support		
TExtramuralSupport	TMainSystemComponentForm	TConceptMap, TAskForHelpMessage			
TConceptMap	TSecondaryComponentForm				
TAskForHelpMessage	TMessageEditor				
TDrawIt	TMainLearningComponentForm				
TFeedback	TMainLearningComponentForm				
TMessageList	TMainLearningComponentForm	TMessageSystem, TNewMessage, TReplyMessage, TForwardMessage, TAllNewMessages,			

Class	Is_a	Has_a	Uses	Properties	Methods
TMessageSystem	TSecondaryComponentForm	TNewMessage, TReplyMessage, TForwardMessage			
TMessageEditor	TSecondaryComponentForm				
TForwardMessage	TMessageEditor				
TNewMessage	TMessageEditor				
TReplyMessage	TMessageEditor				
TAllNewMessages	TSecondaryComponentForm	TReplyMessage, TForwardMessage			
TvlaHelpBtn	TBitBtn	TComponentHelp			
TvlaFolderManager	TComponent				
TvlaFileManager	TComponent				
TComponentHelp	TvImForm	TvlaHelpBtn			
TvlaComponent	TComponent	[TMainComponentForm?]			CreateLearningComponent;
TvlaBaseComponent	TvlaComponent			Component: string;	
TvlaLauncher	TvlaBaseComponent	TLauncher			
TvlaPlayer	TvlaBaseComponent	TPlayer			
TvlaViewer	TvlaBaseComponent	TViewer			
TvlaEditor	TvlaBaseComponent	TEditor			
TvlaResourceBrowser	TvlaBaseComponent	TWebLauncher			
TvImStudyGuide	TvlaComponent	TViewer			
TvImLectureNotes	TvlaComponent	TViewer			
TvImAdministrationGuide	TvlaComponent	TViewer			
TvImAssignment	TvlaComponent	TViewer			
TvImPracticeAssignment	TvlaComponent	TViewer			

Class	Is_a	Has_a	Uses	Properties	Methods
TvImSelfTest	TvIaComponent	TSelfTest			
TvImDiscussion	TvIaComponent	TMessageList			
TvImTutorial	TvIaComponent	TTutorial			
TvImDoItYourself	TvIaComponent	TDoItYourself			
TvImWebExplorer	TvIaComponent	TWebExplorer			
TvImLibraryExplorer	TvIaComponent	TLibraryExplorer			
TvImDrawIt	TvIaComponent	TDrawIt			
TvImFeedback	TvIaComponent	TFeedback			
TvImKeyIdeas	TvIaComponent	TKeyIdeas			
TvImLecture	TvIaComponent	TPlayer			
TvImStudentNotes	TvIaComponent	TEditor			
TvImAssignmentWorkspace	TvIaComponent	TAssignmentWorkspace			

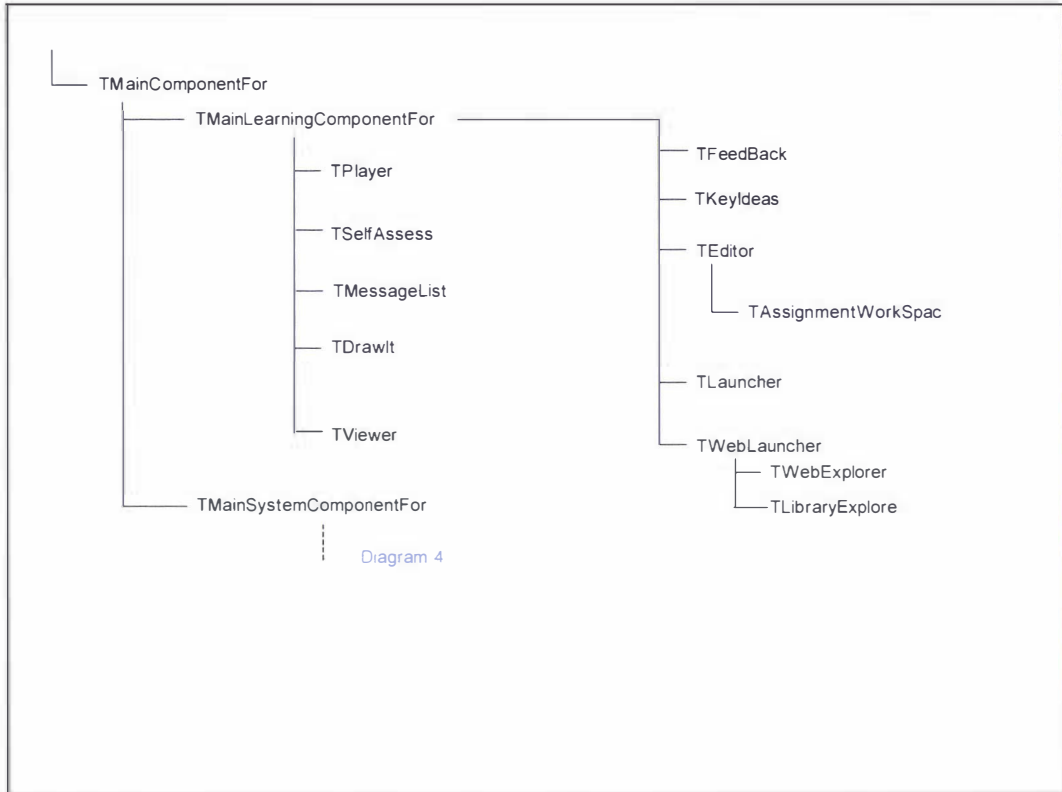


Diagram 2: Delphi Class Hierarchy, Learning Shell. Main Forms for Learning Components.

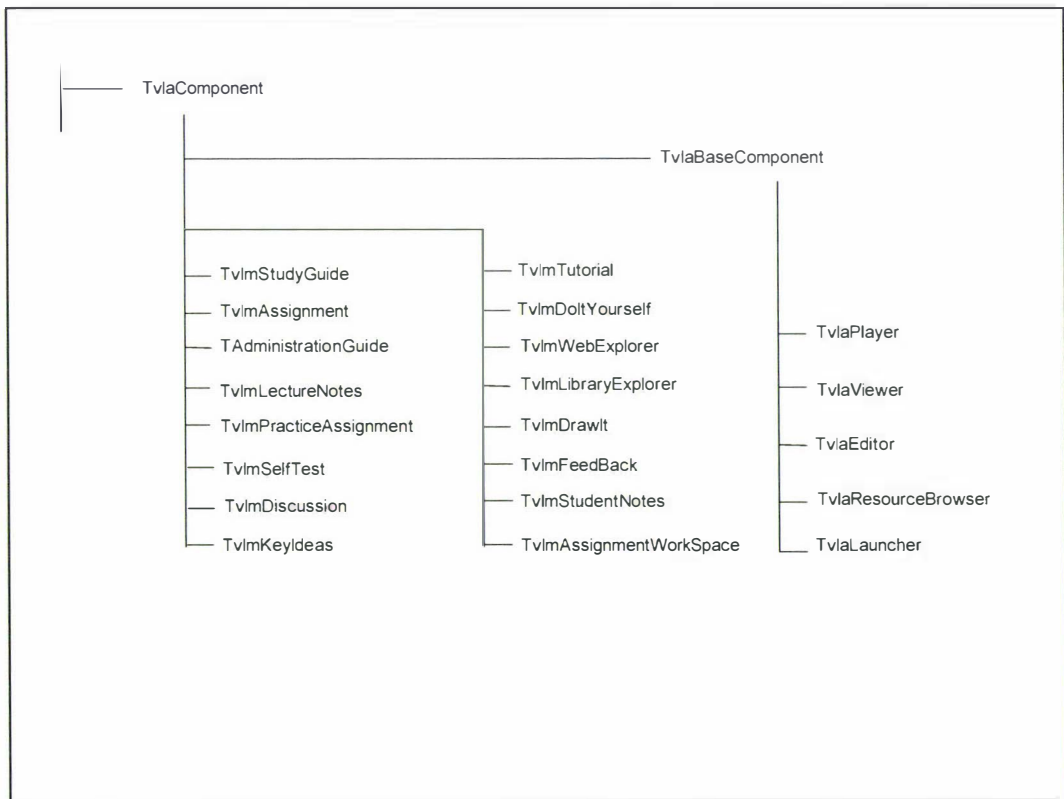


Diagram 3: Delphi Class Hierarchy, Learning Shell. "Drag & Drop" Learning Components.

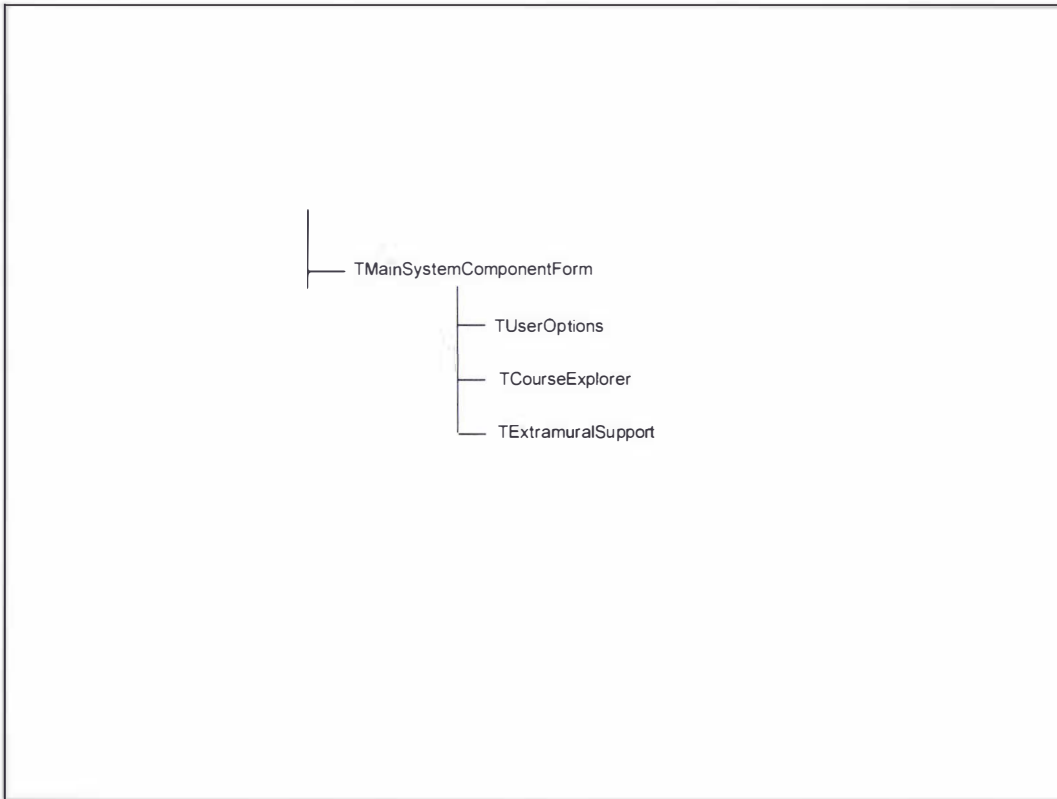


Diagram 4: Delphi Class Hierarchy, Learning Shell. Main Forms for System Components.

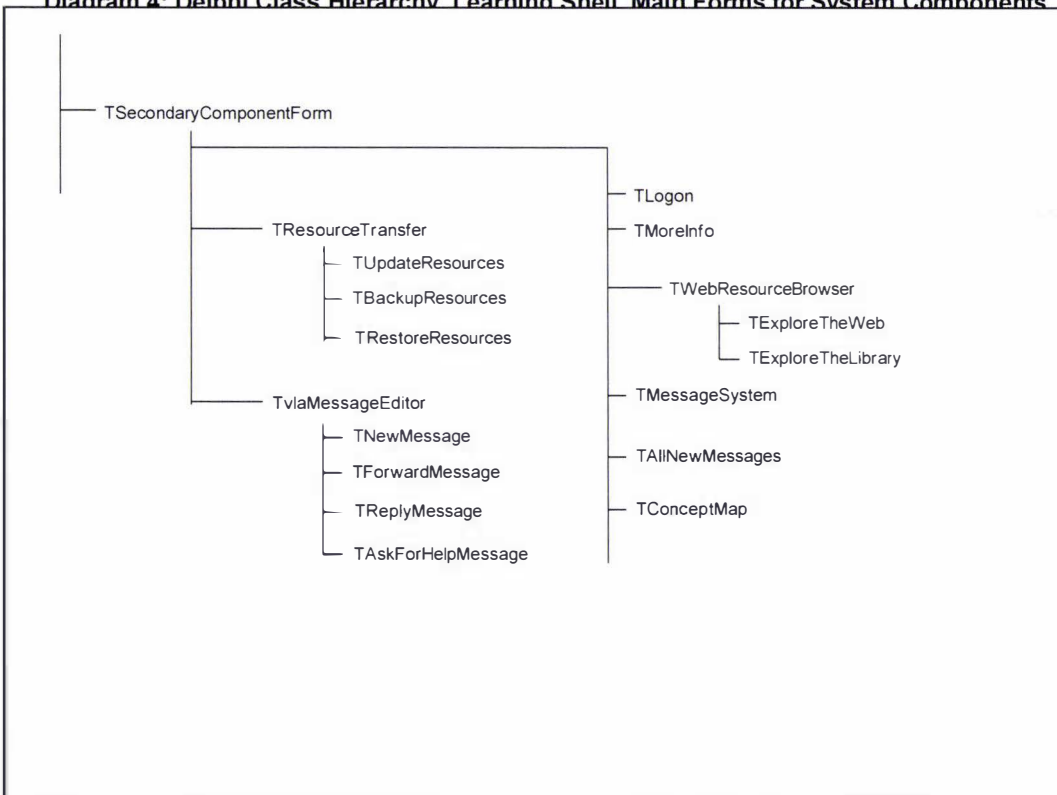


Diagram 5: Delphi Class Hierarchy, Learning Shell. Main Forms for System Components.

Appendix F

Learning Shell – selected class interfaces and source code

F1: Controller object – interface and selected source code¹

```

unit vlaController;
{   This unit defines a Controller class that manages interactions between
{   Shell components. typically, between interface and system components
{   These interactions may involve use of constants defined in the System
{   Dictionary
}

interface
uses
  vlaSystemModelManager, vlaResourceManager, vlaStudentModelManager,
  vlaComponentList, vlaPrintForm, vlaSelfTest, Dialogs, Messages, Controls,
  vlaSystemTreeManager, classes, comctrls, Forms, vlaUserOptions, vlaDesktop,
  vlaClipboard, vlaUpdateExtramuralSupport;

type
  path =record
    component, filepath: string;
  end;
  TController = class (TObject)
  private
    sysModelMan: TSystemModelManager;
    resMan: TResourceManager;
    studModelMan: TStudentModelManager;
    sysTreeMan: TSystemTreeManager;
    pathTable: array[1..20] of path;
    tableLength: integer;
    openGenericList: TCompList;
    visibleComponents: integer;
    PrintForm: TPrintForm ;
    UpdateExtramuralSupport: TUpdateExtramuralSupport;
    Desktop: TDesktop;
    changingMode: boolean;
    //   These operations link Learning Shell learning components with the
    //   corresponding Windows forms
    function findForm (component: string): integer;
    procedure processForm(component: string; action: char);
    procedure initialisePathTable;
    //stores component's current filepath in Pathtable
    procedure setFilePath(component, filepath: string);

  public
    {   These methods are provided to manage all interactions between Learning
    {   Shell components. In general programmers writing new interface learning
    {   components. should only need to call methods relating to file access or
    {   current course settings
    }

    // Setting controller debug to true changes default behaviour of
    // interface components, and extends information collected in log
    // file to facilitate debugging
    debug: boolean;

    {   Creates the system component managers and links controller
    {   with interface via desktop component
    }
    constructor Create(appDesktop: TDesktop);

    {   Information on current course settings
    }
    function getCourseTitle: string;
    function getTopic: integer;
    function getSection: integer;
    function getMode: string;
  end;
end;

```

¹ Early versions of the Learning Shell prototype were called the Virtual Learning Machine (VLM) and Virtual Learning Appliance (VLA). This is reflected in the naming conventions for components, files and other elements of the Shell structure.

```

{   Routines for updating & displaying course status to user   }
    //update status of current topic with result of self-assessment
    //Returns title of current topic
    procedure recordTestResult(status: integer);
    //Brings status of each section, and the course as a whole, in line
    // with status of its topics : completed, attempted, not attempted
    procedure updateSectionStatus;
    function getStatus: string; //of current topic
    // Loads course outline into a TreeView component,
    // showing current status of all nodes
    procedure makeTreeView(treeView: TTreeView);
    function getTopicTitle: string;
    procedure showExplorer;

    Routines for changing topic or study mode in course
    procedure setSection (section: integer);
    procedure setTopic (topic: integer);
    procedure changeMode (newMode: string);
    procedure changeTopic(section,topic: integer);
    procedure changeTopicAndMode(section, topic: integer; mode: string);
    procedure getNextTopic( var VSection, VTopic: integer);
    procedure getPreviousTopic( var VSection, VTopic: integer);
    //Returns true if component should be displayed even if no resource
    //available for current topic
    function alwaysAvailable(component: string): boolean;
    function modeChangeUnderway: boolean;
    //Returns true if at least one learning component is
    // currently displayed on desktop
    function ComponentsShowing: boolean;
    function getColourScheme(mode: string): integer;
    //returns true if component belongs to mode
    function includes(mode, component: string): boolean;
    // returns list of study modes provided for current topic
    function getAvailableModes( section, topic: integer): TStringList;

{   File access management   }
    //Returns component's current filepath. Use this function in
    // a component's show method
    function getFilePath(component: string): string;
    //Returns component's filepath for current topic. Use when copying
    // data to a component that may not be open.
    function copyTo (component: string): string;
    //Returns filepath where the key concepts for the current topic are stored
    function getIdeasFilepath: string;

    //Initialise Extramural Support to concept and display to user
    procedure getSupportOn(concept: string);

{   Start-up   }
    //Logging on routines
    procedure startUp; //the logon process
    //Returns true if logon and startup procedures are not completed
    function isStartingUp: boolean;
    function getUsername: string;
    function getUserPassword: string;
    function confirmPassword(password: string): boolean;
    procedure setNewLogon( name, password: string);
    //display help overview for the Learning Shell to user. Normally accessed through
    //main desktop menu
    procedure showHelp;
    //starting course
    procedure openPrevious; //Mode and topic
    procedure openCourse; // show main desktop

{   Shutdown   }

```



```

procedure performExitRoutine;

{ Messaging management information }
function getLastReceived: TDateTime;
procedure setLastReceived(lastReceived: TDateTime);
function getMsgCount: integer;
procedure setMsgCount(msgCount: integer);
function getMyGroup: integer;
procedure setMyGroup(myGroup: integer); //preset in prototype

{ Routines for managing components available in all modes
and opened and closed by user directly }
procedure showGenericComp(component: string);
procedure closeGenericComp(component: string);
function openInGeneric(component: string): boolean;
// Returns list of all components that can be opened
// and closed by user directly}
function getGenericList: TCompList;

{ Managing components for which the user can create edit resources }
function userCanEdit(component: string): boolean;
function getUserEditList: TCompList;

{ Information on user customisable settings }
procedure setDrive (drive: char; BackupOrUpdate: integer);
function getDrive(BackupOrUpdate: integer): char;

{ Information on system determined settings }
//returns URL of university library
function locateLibrary: string;
//returns IP address of repository FTP server
function getRepositoryIP: string;

{ Routines for updating course resources from repository }
procedure setUpdateFileNo(fileNo: integer);
function getUpdateFileNo: integer;
//update Extramural Support database
procedure UpdateExtramuralSupportFiles;

end;

var
controller: TController;

implementation
uses Sysutils, viaLecture, viaLogon, viaSystemDictionary,
viaStartUpDesktop, viaStudentNotes, viaSystemUtilities,
viaMoreInfo, LMDMSG, viaStudyGuide, viaHelpSystem;

{ TController }

{*****
* Initialisation routines
*****}
constructor TController.Create(appDesktop: TDesktop);
begin
DEBUG :=false; // true, //
initialiseLog;
sysTreeMan:=TSystemTreeManager.Create(vSTUDENT);
resMan := TResourceManager.Create(vSTUDENT);
studModelMan:=TStudentModelManager.Create;
sysModelMan:= TSystemModelManager.Create(studModelMan.getMode);
openGenericList :=TCompList.Create;
UpdateExtramuralSupport := TUpdateExtramuralSupport.create(Application);
initialisePathTable;
visibleComponents:=0;

```

```

Desktop:= appDesktop;
end;

...

{ end of initialisation routines }

{-----}
*
*                               Mode management routines
*                               -----
*
}

procedure TController.changeMode(newMode: string);
var
    toClose, toOpen: TCompList;
begin
    tellUser('Controller', 'Change mode: '+newMode, LOG_IT, false);
    changingMode := true;
    try
        //update system model with new mode and obtain lists of components
        //to open and components to close
        toClose := TCompList.Create;
        toOpen := TCompList.Create;
        sysModelMan.updateMode(newMode, toClose, toOpen);

        //update student mode
        StudModelMan.updateModel(newMode);

        //update Desktop with new mode
        Desktop.FormShow (nil);
        Desktop.synchroniseMenu(newMode);

        //close components not in new mode
        if not toClose.isEmpty then
            processForm(toClose.getFirstItem, 'c')
        else
            tellUser('Controller', 'Nothing to close', LOG_IT, false);
        while toClose.isMore do
            processForm(toClose.getNextItem, 'c');

        //get resources associated with new components and open them
        if not toOpen.isEmpty then
            processForm(toOpen.getFirstItem, 's');
        while toOpen.isMore do
            processForm(toOpen.getNextItem, 's');

        //close Explorer
        Desktop.ExplorerClose;

    finally
        toOpen.Free;
        toClose.Free;
        changingMode:=false;
    end;
end;

{-----}
*
*                               This routine processes the form associated with each
*                               component passed to it If the action required is 'c'
*                               the form is closed if 's' the associated file is
*                               identified and the form is shown with the file opened
*                               -----
*
}

procedure TController.processForm( component: string; action: char);
var
    i: integer;
begin

```

```

try
  //check that component's form exists
  if (findForm(component)<Screen.FormCount) then

    if action = 's' then begin //open component
      // add to open component count
      inc(visibleComponents);

      //get Resource
      setFilePath(component, resMan.getResource(getSection, getTopic,
        component, sysModelMan.userCanEdit(component),
        sysModelMan.isAlwaysTheSame(component),
        sysModelMan.isSectionWide(component)));

      i:= findForm(component) ; //no of screen forms may have changed
      parseCaption( Screen.Forms[i]);
      tellUser (Screen.Forms[i].Name,
        'Virtual File: '+getFilePath(component), debug, debug);

      //Only open component if resource available
      //or component is user modifiable
      if (getFilePath(component)<> NO_FILE) or
        (sysModelMan.isAlwaysAvailable(component)) then
        if sysModelMan.DisplayAsModal(component) then begin
          i:= findForm(component) ;
          tellUser (Screen.Forms[i].Name, 'Show Modal',
            LOG_IT, debug);
          Screen.Forms[i].ShowModal ;
        end
        else begin
          i:= findForm(component) ;
          tellUser (Screen.Forms[i].Name, 'Show',
            LOG_IT, debug);
          Screen.Forms[i].Show;
        end
      else
        // so if we want something to happen when no resource //available
        tellUser(Screen.Forms[i].Name, 'Not shown: '+NO_FILE,
          LOG_IT, debug)
      end //of opening component

    else begin //action is 'c' so close component
      i := findForm(component);
      //subtract from open components count
      dec(visibleComponents);

      if sysModelMan.DisplayAsModal(component) then begin
        tellUser (Screen.Forms[i].Name, 'Close Modal',
          LOG_IT, debug);
        Screen.Forms[i].ModalResult:=1 ;
      end
      else begin
        tellUser (Screen.Forms[findForm(component)].Name, 'Hide',
          LOG_IT, debug);
        Screen.Forms[i].hide;
      end;
    end; //of close component
  except
    on Exception do
      raise // for application to log
  end;
end; { of processForm }

```

```

{                                     SHUTDOWN ROUTINES                                     }
{
*   Perform any housekeeping tasks before the
*   systems terminates
*
}
procedure TController.performExitRoutine;
begin
  if LMDMessageDlg('Exit course now?',
    mtConfirmation, [mbYes, mbNo], 0)=mrYes then begin
    //for each open generic component ensure that their OnClose event
    // is called before the system terminates
  if not openGenericList.isEmpty then
    Screen.Forms[findForm( openGenericList.getFirstItem)].close ;
  while openGenericList.isMore do
    Screen.Forms[findForm( openGenericList.getNextItem)].close ;

    // Backup user modifiable resource
  if LMDMessageDlg('Backup data before closing?',
    mtConfirmation, [mbYes, mbNo], 0) = mrYes
  then UserOptions.startBackupAndTerminate //application
  else
    application.Terminate;

  end;
end;

{                                     END OF SHUTDOWN ROUTINES                                     }
{
*   Locate the form for an open component. Returns position of
*   component's main form* in Screen.Forms array.
*
}
function TController.findForm (component: string): integer;
  var i: integer;
begin
  i:=0;
  while ((i < Screen.FormCount-1)
    and (component <> Screen.Forms[i].Name) do
    inc(i);
  Result := i;
end;
...

procedure TController.changeTopic(section, topic: integer);
var
  modeComps.genComps:TCompList;
  component: string;
begin
  tellUser('Controller',
    'Change topic: '+ parseSection(section, topic), LOG_IT, false);
  // update student model
  studModelMan.updateModel(section, topic);

  // get list of all generic components
  genComps := sysModelMan.getGenericComponents;
  // close all open generic components
  if not genComps.isEmpty then begin
    component := genComps.getFirstItem;
    if openGenericList.contains(component) then
      closeGenericComp(component);
  end;
  while genComps.isMore do begin
    component := genComps.getNextItem;
    if openGenericList.contains(component) then
      closeGenericComp(component);
  end;
end;

```

```

// get list of all components associated with current mode
modeComps:=sysModelMan.getComponents(getMode);
//and close them
if not modeComps.isEmpty then
    processForm(modeComps.getFirstItem, 'c');
while modeComps.isMore do
    processForm(modeComps.getNextItem, 'c');

//reopen all components in current mode
if not modeComps.isEmpty then
    processForm(modeComps.getFirstItem, 's');
while modeComps.isMore do
    processForm(modeComps.getNextItem, 's');

//close Explorer
Desktop.ExplorerClose;
//update Desktop
Desktop.FormShow(nil);

end;

procedure TController.showExplorer;
begin
    Desktop.ExplorerLaunch;
end;

procedure TController.changeTopicAndMode(section, topic: integer;
mode: string);
var
    modeComps,genComps:TCompList;
    component: string;
begin
    {      close all open components      }

    // get list of all generic components
    genComps := sysModelMan.getGenericComponents;
    // close all open generic components
    if not genComps.isEmpty then begin
        component := genComps.getFirstItem;
        if openGenericList.contains(component) then
            closeGenericComp(component);
    end;
    while genComps.isMore do begin
        component := genComps.getNextItem;
        if openGenericList.contains(component) then
            closeGenericComp(component);
    end;

    // get list of all components associated with current mode
    modeComps:=sysModelMan.getComponents(getMode);
    //and close them
    if not modeComps.isEmpty then
        processForm(modeComps.getFirstItem, 'c');
    while modeComps.isMore do
        processForm(modeComps.getNextItem, 'c');

    {      all components closed      }

    // set the new section and topic
    studModelMan.updateModel(section, topic);
    tellUser('Controller',
        'Change topic: '+ parseSection(section, topic), LOG_IT, false);

```

```

    //change the mode with the new section and topic
    changeMode(mode);
end;

procedure TController.initialisePathTable;
var
    allComps:TCompList;
    i:integer;
begin
    //get list of all learning components
    allComps:= sysModelMan.getAllComponents;
    //insert them into pathtable
    i:=1;
    pathTable[i].component :=allComps.getFirstItem;

    while allComps.isMore do begin
        inc(i);
        pathTable[i].component:= allComps.getNextItem;
    end;
    tableLength:=i;
end;

procedure TController.getPreviousTopic(var VSection, VTopic: integer);
begin
    sysTreeMan.getPreviousTopic( VSection, VTopic);
end;

function TController.getIdeasFilepath: string;
begin
    Result := resMan.getResource(getSection, getTopic, vIDEAS,
        sysModelMan.userCanEdit(vIDEAS),
        sysModelMan.isAlwaysTheSame(vIDEAS),
        sysModelMan.isSectionWide(vIDEAS));
end;

// returns filepath of component resource to be copied into
function TController.copyTo (component: string): string;
begin
    Result:= resMan.getResource(getSection, getTopic, component,
        sysModelMan.userCanEdit(component),
        sysModelMan.isAlwaysTheSame(component),
        sysModelMan.isSectionWide(component));
end;

procedure TController.UpdateExtramuralSupportFiles;
begin
    UpdateExtramuralSupport.UpdateExtramuralSupport;
end;

procedure TController.getSupportOn(concept: string);
begin
    ExtramuralSupport.selectConcept(concept);
    ExtramuralSupport.show;
end;

end. //end of viaController

```

F2: Course Explorer object – class interface and selected source code

```

unit viaCourseExplorer;
{   This unit provides the functionality for navigating anywhere within the course   }
{   The user accesses this functionality through the CourseExplorer object or     }
{   from the Desktop Menu                                                         }
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ComCtrls, FileCtrl, ImgList, ExtCtrls, Buttons, viaHelpBtn;

type
  TCourseExplorer = class(TForm)
    lbResources: TListBox;
    ...
    viaHelpBtn1: TviaHelpBtn;
    procedure btCancelClick(Sender: TObject);
    procedure btOKClick(Sender: TObject);
    procedure viaHelpBtn1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure lbResourcesDbClick(Sender: TObject);
    procedure TreeView1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure TreeView1DbClick(Sender: TObject);
    procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
  private
    procedure selectMode;
    procedure clearResources;
    procedure openTopic;
    function findTopic(title: string): TTreeNode;
    procedure expandCurrentSection; //in Tree View
  public
    procedure getNext( currentSection, currentTopic: integer );
    procedure getPrevious( currentSection, currentTopic: integer );
  end;

var
  CourseExplorer: TCourseExplorer;

Implementation

uses viaController, viaSystemDictionary, viaDesktop, viaSystemUtilities,
  LMDMSG;
{$R *.DFM}

...

procedure TCourseExplorer.FormShow( Sender: TObject );
begin
  tellUser(self.name, 'Show', LOG_IT, false);
  lbCourseTitle.Caption:= controller.getCourseTitle;
  lbSection.Caption:= intToStr(controller.getSection);
  lbTopic.Caption := intToStr(controller.getTopic);
  lbTopicTitle.caption:=controller.getTopicTitle;
  lbStatus.Caption := controller.getStatus;

  //get list of available resources and display in list box
  lbResources.Items:= controller.getAvailableModes(controller.getSection, controller.getTopic);
  controller.updateSectionStatus;
  controller.makeTreeView(TreeView1);

```

```

        expandCurrentSection;
    end;

procedure TCourseExplorer.lbxResourcesDbClick(Sender: TObject);
begin
    selectMode;
end;

procedure TCourseExplorer.selectMode;
var
    i:integer;
    newSection, newTopic: integer;
    oldSection, oldTopic: integer;
    newMode, oldMode : string;
    sameTopic, sameMode, somethingShowing: boolean;
begin
    //initialise variables
    sameTopic:=false;
    sameMode := false;
    newSection := strToInt(lbSection.caption);
    newTopic := strToInt(lbTopic.Caption);
    oldSection := controller.getSection;
    oldTopic := controller.getTopic;
    oldMode := controller.getMode;
    somethingShowing := controller.ComponentsShowing;

    //check whether same topic has been selected
    if ( (newSection = oldSection) and (newTopic = oldTopic)) then sameTopic := true;

    // identify the new mode selected
    i:=0;
    while ((i <= lbxResources.Items.Count-1) and not lbxResources.Selected[i]) do
        inc(i);
    if i<> lbxResources.Items.Count then
        newMode:= lbxResources.Items.Strings[i]
    else newMode :=oldMode;

    //check whether same mode has been selected
    if newMode =oldMode then sameMode:=true;

    //now determine what to do
    if sameTopic and sameMode and somethingShowing then
        self.Hide //do nothing
    else if sameTopic then
        controller.changeMode(newMode)
    else if sameMode then
        controller.changeTopic(newSection, newTopic)
    else //different topic and mode
        controller.changeTopicAndMode( newSection, newTopic, newMode );
end;

procedure TCourseExplorer.TreeView1MouseDown(Sender: TObject;
    Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var
    AnItem: TTreeNode;
    str, st: string;
    i: integer;
    newSection, newTopic: integer;
begin
    newSection:=-1; //initialisation
    newTopic :=-1;
    if TreeView1.Selected = nil then
        Exit;

    AnItem := TreeView1.GetNodeAt(X, Y);
    TreeView1.Selected := AnItem ; //synchronise with cursor position
    lbTitle.caption:= AnItem.Text;

```



```

if AnItem.text = controller.getCourseTitle then begin
    lbSection.caption:="";
    lbTopic.caption:="";
    lbTopicTitle.caption:="";
    lbStatus.caption := 'Whole course';
    clearResources;
    Exit;
end;

i:=1;
str:="";
st:=lbTitle.caption;
while st[i]<>'.' do begin
    str:= str + st[i];
    inc(i);
end;
lbSection.caption:= str;
if str <> " then
    newSection:= strToInt(str);

str:="";
st:=lbTitle.caption;
inc(i);
while st[i] in ['0'..'9'] do begin
    str:= str + st[i];
    inc(i);
end;
lbTopic.caption:= str;
if str<> " then
    newTopic := strToInt(str);

str:="";
st:=lbTitle.caption;
inc(i);
while i<= length(st) do begin
    str:= str + st[i];
    inc(i);
end;
lbTopicTitle.caption:= str;

case( AnItem.StateIndex ) of
    -1:    lbStatus.caption:= 'Unknown';
    1:    lbStatus.caption := NOT_ATTEMPTED;
    2:    lbStatus.caption := ATTEMPTED;
    3:    lbStatus.caption:= COMPLETED;
end;

//update Resources box if topic selected
if newTopic >=0 then
    lbResources.Items:= controller.getAvailableModes( newSection, newTopic )
else clearResources;
end;

procedure TCourseExplorer.getNext( currentSection, currentTopic :integer);
var
    VSection, VTopic: integer;
begin
    VSection:= currentSection;
    VTopic:= currentTopic;
    controller.getNextTopic( VSection, VTopic ); // passing var parameters
    if VSection = END_OF_COURSE then begin
        showVLAMessage( 'End of course!' );
    end
    else begin
        controller.changeTopic( VSection, VTopic );
    end;
end;
end;

```

```

procedure TCourseExplorer.clearResources;
begin
    lbxResources.Items.Clear;
end;

procedure TCourseExplorer.openTopic;
var
    newSection, newTopic: integer;
begin
    if lbTopic.Caption <> "" then begin
        newSection := strToInt( lbSection.caption );
        newTopic := strToInt ( lbTopic.Caption );
        controller.changeTopic( newSection, newTopic );
    end
    else showVLAMessage( 'Please select topic.' );
end;

procedure TCourseExplorer.TreeView1DbClick( Sender: TObject );
begin
    openTopic;
end;

procedure TCourseExplorer.btOKClick(Sender: TObject);
begin
    selectMode
end;

procedure TCourseExplorer.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    action:=caHide;
end;

procedure TCourseExplorer.getPrevious( currentSection, currentTopic: integer );
var
    VSection, VTopic: integer;
begin
    VSection:= currentSection;
    VTopic:= currentTopic;
    controller.getPreviousTopic( VSection, VTopic ); // passing var parameters
    if VSection=START_OF_COURSE then begin
        showVLAMessage('Start of course!');
    end
    else
        controller.changeTopic( VSection, VTopic );
end; //of getPrevious

procedure TCourseExplorer.vlaHelpBtn1Click( Sender: TObject );
begin
    vlaHelpBtn1.Execute(self );
end;

procedure TCourseExplorer.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState );
begin
    if Key = VK_F1 then
        vlaHelpBtn1.Execute (self );
end;

procedure TCourseExplorer.FormCreate(Sender: TObject);
begin
    if controller.debug then
        Self.FormStyle:= fsNormal
    else Self.FormStyle:= fsStayOnTop;
end;

function TCourseExplorer.findTopic( title: string ): TTreeNode;
var

```

```
    curlItem: TTreeNode;
begin
    CurlItem := TreeView1.Items.GetFirstNode;
    while CurlItem.Text <> title do begin
        CurlItem := CurlItem.GetNext;
    end;
    Result := curlItem;
end;

procedure TCourseExplorer.expandCurrentSection;
var
    str: string;
    curTopic: TTreeNode;
begin
    TreeView1.Items[0].Expand(false);
    str:=lbSection.Caption+' '+lbTopic.Caption+' '+lbTopicTitle.Caption;
    curTopic:=findTopic(str);
    curTopic.Parent.Expand( true );
end;

end. //of CourseExplorer unit
```

F3: System Dictionary – constants

```
unit vIaSystemDictionary;
```

```
{      This unit provides central place to define constants and code fragments      }
{      used throughout the prototype, including the Learning Shell, the authoring    }
{      application, and the Repository Manager.                                     }
}
```

```
interface
const
```

```
{      Directory Paths      }

ROOTPATH = 'c:\Program Files\VLM\';
RESOURCE_DIR = ROOTPATH+ 'Resources\';
PR_RESOURCE_DIR = RESOURCE_DIR + 'Print_Resources\';
LECTURE_DIR = RESOURCE_DIR+ 'Lecture\';
IMAGE_DIR = LECTURE_DIR;// ROOTPATH+ 'Lecture\';
SYSTEM_DIR = ROOTPATH + 'System\';
SYSTEM_MODELS_DIR = SYSTEM_DIR + 'Models\';
UPDATE_DIR = 'Update\';
UPDATES_TEMP = 'C:\VLM\';
UPDATES_ARCHIVE = 'UpdatesArchive\';
BACKUP_RESOURCE_DIR = 'Backup\Resources\';
BACKUP_PR_RESOURCE_DIR = BACKUP_RESOURCE_DIR + 'Print_Resources\';
IMPORT_DIR = 'C:\WINDOWS\Desktop\Imports\';
EXPORT_DIR = 'C:\WINDOWS\Desktop\Exports\';
TUTORIAL_DIR = RESOURCE_DIR + 'Tutorial\';
DB_UPDATES = SYSTEM_DIR + 'DB_Updates\';
SYSTEM_LOG_DIR = SYSTEM_DIR + 'Log\';
TEMP_DIR = SYSTEM_DIR + 'Temp\';
HELP_DIR = SYSTEM_DIR + 'Help\';
HELP_UPDATES= ROOTPATH+ 'HelpUpdates\';
HELP_UPDATE_DIR = ROOTPATH+ 'HelpUpdates\';

{      end of file path constants      }

{      System files      }

VISITED = 'visited.txt';

{      end of System Files      }

{      Student Model      }

DEFAULT_NAME = 'STUDENTNAME';
vBACKUP = 1;
vUPDATE = 2;
vLECTURE = 3;
NO_DEFAULTS = -1;

{      end of student model constants      }

{      RESTORING BACKUPS      }

vPR_RESOURCE = 1;
vRESOURCE =2;

{      end of RESTORING      }
```

```

{
    RESOURCES MANAGEMENT
}

NO_FILE = 'No Resource Available';
SUCCESS = 'Resources successfully transferred.';
SUFFIXES = 'suffixes.txt';

{
    end of RESOURCES
}

{
    USER SUPPORT SYSTEM
}

type
QuType = (qHow, qWhat, qWhere, qWho, qWhy);
const
    HP_ENDINGS = 'endings.txt' ;
    MAX_CONCEPTS = 10;
    MAX_THEMES = 10;

{
    end of SUPPORT SYSTEM
}

{
    RTF editors
}

HEAD = 16;
SUBHEAD = 14;
NORMAL = 12;

{
    end of RTF editors
}

{
    QUIZ
}

ALL_QUESTIONS = 10;
SELECTIONS = 6;
OPTIONS = 4;
REPEATS = 4;
type
    TQuizResult= record
        concept: string;
        count: integer;
        responses: array [1..REPEATS] of integer;
    end;
    TQuizResults = array [1..SELECTIONS] of TQuizResult;

{
    end of QUIZ
}

{
    QUESTIONNAIRE
}

    TQuestionnaireResult= record
        concept: string;
        count: integer;
        responses: array [1..REPEATS] of integer;
    end;
    TQuestionnaireResults = array [1..MAX_CONCEPTS] of TQuestionnaireResult;

{
    end of QUESTIONNAIRE
}

{
    LECTURE
}

const
DEFAULT_IMAGE = 'bmpLecture.bmp';

{
    end of LECTURE
}

```

```

{                               COMPONENTS                               }

vLINKS = 'WebExplorer';
vBOOKS = 'LibraryExplorer';
vNOTES = 'StudentNotes';
vIDEAS = 'KeyIdeas';

{                               end of COMPONENTS                       }

{                               SYSTEM MODEL, MODES, COMPONENTS        }

MAX_MODES = 10;
MAX_COMP = 20;
htGENERIC = 220 ;

{                               end of SYSTEM MODEL                     }

{                               SYSTEM FONTS                           }

vtHandWriting ='Connecticut';

{                               end of SYSTEM FONTS                     }

{                               SYSTEM TREE                             }

vRED = 1;
vAMBER = 2;
vGREEN= 3;
MAX_TOPICS = 12;
MAX_SECTIONS = 12;
END_OF_COURSE = -1;
START_OF_COURSE = -2;
NONE_FOUND = -3;
NOT_ATTEMPTED ='Not attempted';
ATTEMPTED = 'Attempted';
COMPLETED = 'Completed';
vSTUDENT =1;
vTEACHER = 2;

{                               end of SYSTEM TREE section             }

{                               AUTHORIZING INTERFACE section          }

AUTHOR_DIR = 'C:\VLM_Repository\Teachers\';
AU_WORK_DIR = AUTHOR_DIR + 'Working_Files\';
AU_DATABASE_DIR = AUTHOR_DIR + 'Database\';
AU_DB_UPDATES = AUTHOR_DIR + 'Updates\';
AU_SYSTEM_DIR = AUTHOR_DIR + 'System\';
AU_MODELS_DIR = AU_SYSTEM_DIR + 'Models\';
AU_SYSTEM_LOG_DIR = AU_SYSTEM_DIR + 'Log\';
AU_TEMP_DIR = AU_SYSTEM_DIR + 'Temp\';
BASIC = 'basicmodel.txt';
OUTLINE = 'initialoutline.txt';
REPOSITORY_DIR = 'C:\VLM_Repository\Repository\';
COURSE_OUTLINE = 'contents.txt';
AU_RESOURCE_DIR = REPOSITORY_DIR+ 'VLM\Resources\';
AU_PR_RESOURCE_DIR = AU_RESOURCE_DIR + 'Print_Resources\';
AU_LECTURE_DIR = AU_RESOURCE_DIR+'Lecture\';
AU_TUTORIAL_DIR = AU_RESOURCE_DIR+'Tutorial\';
AU_ARCHIVE_DIR = REPOSITORY_DIR +'Archive\';
AU_ARCHIVE = REPOSITORY_DIR +'Archive\';
AU_CD_UPDATE_ONLY_DIR = REPOSITORY_DIR+ 'CD_Only\';
IMAGE_SELECTED =1;

```

```

NO_IMAGE = 2;

{           end of AUTHORIZING INTERFACE      section           }

{           REPOSITORY MANAGER section       }

RP_MESSAGE_DIR = REPOSITORY_DIR + 'MessageQueue\';
RP_DATABASE_DIR = REPOSITORY_DIR + 'Database\';
RP_TEMP_DIR = REPOSITORY_DIR + 'Temp\';
STUDENT_DIR = 'C:\VLM_Repository\Students\';
RP_SYSTEM_DIR = REPOSITORY_DIR + 'System\';
RP_SYSTEM_LOG_DIR = RP_SYSTEM_DIR + 'Log\';
RP_HELP_FILE = REPOSITORY_DIR + 'Help.txt';
RP_UPDATE_DIR = REPOSITORY_DIR + 'VLM\';
RP_HELP_UPDATE_DIR = RP_UPDATE_DIR + 'HelpUpdates\';
RP_UPDATED_TABLES_LIST = RP_SYSTEM_DIR + 'updatedtables.txt';

{*****
                                     TRANSMISSION HEADER FORMAT
*****}

<TO RCVR>
<FROM SNDR>
<MESSAGE TYPE>
<GROUP NUMBER>
<DATETIME STAMP OF LAST TRANSMISSION RECEIVED FROM RCVR> default = 0 (send all)
}

{           Transmission types           }

tsLAST_RCVD = 1 ;
tsMSG_LST = 2 ;
tsTEST = 3;

ANSWER = 'answer.txt';
MESSAGE_LIST = 'messages.txt';
REPOSITORY = 'System';
TUTOR = 'Tutor';
DEFAULT_GROUP = 0;

{           end of REPOSITORY MANAGER section           }

implementation

end. // of System Dictionary

```

F4: System Utilities – class interface

```

unit vlaSystemUtilities;
{   This unit provides a collection of commonly used functions and   }
{   procedures, which can be accessed from anywhere within the prototype }

interface
uses vlaSystemMessage, Forms, Windows;

// Returns maximum of two real numbers. May be used with dates and times
function max ( x,y: double): double;

#disable/enable Windows system keys
procedure DisableSystemButtons;
procedure EnableSystemButtons;

/debugging & communicating with user
const
    LOG_IT = true;
    TELL_IT = true;
procedure initialiseLog;
procedure closeLog;
procedure tellUser( component, msg: string; debug: boolean ); overload;
procedure tellUser( component, msg: string; logIt, tellIt: boolean ); overload;
procedure tellUser( component, msg: string; logIt, tellIt, isTeacher: boolean ); overload;
procedure showVLAMessage ( msg: string); overload;
procedure showVLAMessage ( component, msg: string); overload;

//parsing strings to provide white space before every upper case character
procedure parseCaption( vlaForm: TForm );
function parseLabel( caption: string ): string;

//returns file suffix
function getSuffix( filename: string ): string;

//Returns input section and topic in string form as 'ss tt'
function parseSection( section, topic: integer ): string;

//for Win 3.1 controls which list directory names as '[foldername]
function stripBrackets( str: string): string;

#Takes filename in 'form name_no' and returns 'no' as an integer
function getFileNumber( filename: string ): integer;

{.....
    Error checking
.....}

// check for valid section , topic range
function isValid( sectionOrTopic: string): boolean;

implementation

uses Dialogs, vlaSystemDictionary, SysUtils, vlaController;

...
end.

```


F5: Resource Model Manager – interface and selected source code

```

unit vlaResourceManager;
{
  This class encapsulates and provides an interface to the resources model,
  enabling the implementation of the resources model to be hidden from the
  rest of the Learning Shell. Other components of the Learning Shell interact
  via the Controller object
}

interface
uses vlaResources, classes, vlaComponentList;

type
  TResourceManager = class
  private
    RootDirectory: TRootDirectory; //the Resources model
    userType: integer;
  public
    {
      Creates a resources object that encapsulates the directory structure
      of the current course for either a student or a teacher/author
    }
    constructor Create(studentOrTeacher:integer);
    {
      Returns the full filepath for the resource associated with the
      specified section, topic and component, or a NO_FILE message
      if no resource found
    }
    function getResource(section, topic: integer; component: string;
      canEdit, alwaysTheSame, sectionWide:boolean): string;
    {
      Returns the list of components for which resources available in the
      specified section and topic
    }
    function getResourceList(section,topic:integer): TStringList; overload;
    function getResourceList(section : integer;
      sectionWideList: TCompList): TStringList; overload;
    {
      Returns the component associated with a given Learning Shell filename
    }
    function GetComponent(filename:string):string;
    {
      Returns the Learning Shell file code associated with the input section, topic
      and component.
    }
    function getFileCode(section, topic: integer; component: string): string;
    {
      Returns true if component should be displayed even if no resource is
      available. Used by RootDirectory to access system model information
      via the Controller
    }
    function alwaysAvailable(component:string): boolean;
  end;

implementation

uses sysutils, vlaSystemDictionary, vlaController;

{ TResourceManager }

{...}

function TResourceManager.getResource (section, topic: integer; component: string;
  canEdit, alwaysTheSame, sectionWide:boolean): string;
var
  str:string;
begin
  str:= RootDirectory.getFile(section, topic, component, false,
    canEdit,alwaysTheSame, sectionWide);
  if NO_FILE = str then
    Result:= str
  else
    if userType = vSTUDENT then
      Result:= RESOURCE_DIR + str
    else Result :=AU_RESOURCE_DIR + str;
end;

end.

```

F6: Student Model Manager – class interface

```

unit vlaStudentModelManager;
{
  This class encapsulates and provides an interface to the student model,
  enabling the implementation of the student model to be hidden from the
  rest of the Learning Shell. Other components of the Learning Shell interact
  via the Controller object.
}

interface
  uses vlaStudentModel;
  type
    logon = record
      name, password:string;
    end;
    TStudentModelManager = class
      private
        studentModel: TStudentModel;
      public
        {
          Creates a student model initialised to the current student
          constructor Create;
        }
        {Procedures for updating the student model with the user's current
        {
          position in the course section, topic, study mode. Student
          model is saved whenever it is updated
        }
        procedure updateModel (mode:string); overload;
        procedure updateModel(section, topic:integer); overload;
        procedure updateModel(section, topic:integer; mode:string); overload;

        {
          Procedures for accessing and updating information held by
          the student model
        }
        function getMode: string; //current study mode
        function getSection:integer; //current section
        procedure setSection(section:integer);
        function getTopic: integer; //current topic
        procedure setTopic(topic:integer);
        procedure setLogon(name, password:string);
        function getStudName:string;
        function getStudPassword:string;
        function confirmPassword(password:string):boolean;

        {
          file transfer settings
        }
        procedure setDrive (drive:char;BackupOrUpdate:integer);
        function getDrive(BackupOrUpdate:integer):char;
        procedure setUpdateFileNo(fileNo:integer);
        function getUpdateFileNo:integer;

        {
          message transfer settings
        }
        function getLastReceived: TDateTime;
        procedure setLastReceived(lastReceived:TDateTime);
        function getMyGroup:integer;
        procedure setMyGroup(myGroup: integer);
        function getMsgCount: integer;
        procedure setMsgCount(msgCount: integer);
      end;
  implementation
  uses SysUtils, vlaSystemDictionary;

  { TStudentModelManager }

  {...}
end.

```

F7: System Model Manager – class interface and selected code

```

unit vlaSystemModelManager;
{ Provides the functionality needed to maintain the system model i.e. study modes and their
{ learning components. Other components of the Shell interact via the Controller object. }

interface
uses vlaComponentList, vlaSystemModel, dialogs, classes, Messages;

Type
  TSystemModelManager = class
  private
    sysModel: TSystemModel;
    startingUp: boolean;
  public
    { Creates a system model initialised to 'initialMode' study mode }
    constructor Create(initialMode: string);
    { Updates system model with the new Mode and returns a list of }
    { components to close and a list to open to complete the change }
    { of study mode }
    procedure updateMode( newMode: string;
                          toCloseList, toOpenList: TCompList );
    { Returns true if resources associated with 'component' may be }
    { edited by the student }
    function userCanEdit(component: string): boolean;
    { Returns list of all components whose resources can be edited }
    { by the student }
    function getUserEditList: TCompList;
    { Returns true if 'component' is always displayed to the user }
    { even where no resource is available }
    function isAlwaysAvailable(component: string): boolean;
    { Returns true if 'component' displays the same resource }
    { at any position in the course }
    function isAlwaysTheSame(component: string): boolean;
    { Returns true if the same 'component' resource should be }
    { displayed at any topic within a section }
    function isSectionWide(component: string): boolean;
    { Returns list of all sectionwide components }
    function getSectionWideList: TCompList;
    function getLibraryURL: string;
    { Returns the IP address of the System FTP server }
    function getServerIP: string;
    { Returns true if 'component' should be displayed as a modal }
    { form }
    function displayAsModal(component: string): boolean;
    { Takes a list of components and returns a list of all the }
    { modes that can be displayed with those components }
    function getAvailableModeList(resourceList: TStringList): TStringList;
    { Return list of all components associated with 'mode' }
    function getComponents(mode: string): TCompList;
    { Returns a list all components the user can open from }
    { within any mode }
    function getGenericComponents: TCompList;
    function getAllComponents: TCompList;
    {Returns true if 'mode' contains 'component' }
    function includes(mode, component: string): boolean;
    function getColourScheme(mode: string): integer;
  end;

implementation

{ TSystemModelManager }

uses vlaSystemUtilities;

```

```

{...}
procedure TSystemModelManager.updateMode( newMode: string; toCloseList, toOpenList: TCompList);
var
    item: string;
    list:TCompList;
begin
    {
        For each open component,
        {
            if not in new mode then
            {
                tell controller to close it
            }
            delete from open list
        }
    }
    if not sysModel.openComponents.isEmpty then
    begin
        item := sysModel.openComponents.getFirstItem;
        if not sysModel.contains(newMode,item) then
        begin
            toCloseList.insertComponent(item);
        end;

        while sysModel.openComponents.isMore do
        begin
            item := sysModel.openComponents.getNextItem;
            if not sysModel.contains(newMode,item) then
            begin
                toCloseList.insertComponent(item);
            end
        end
    end
    end
    else tellUser('System Model Manager',
        'Nothing in Open Components list', LOG_IT,false);

    {
        update open components list by deleting items in toClose list
    }
    if not toCloseList.isEmpty then
    sysModel.openComponents.deleteComponent(toCloseList.getFirstItem );
    while toCloseList.isMore do
    sysModel.openComponents.deleteComponent(toCloseList.getNextItem );

    {
        for each component in new Mode
        {
            if not generic component
            {
                tell controller to open it
            }
            insert into open list
        }
    }
    list := sysModel.getCompList(newMode);
    item:= list.getFirstItem;

        toOpenList.insertComponent(item);
        if startingUp then
            sysModel.openComponents.insertComponent(item);
    while list.isMore do
    begin
        item:= list.getNextItem;
        toOpenList.insertComponent(item);
        if startingUp then
            sysModel.openComponents.insertComponent(item);
        end;
    if startingUp then
        startingUp:=false;

end;

end;

end. // of System Model Manager

```

F8: System Tree Manager – class interface

```

unit vlaSystemTreeManager;
{
  This class encapsulates and provides an interface to the system tree.
  which represents the course table of contents, enabling the
  implementation of the system tree to be hidden from the rest of the Learning Shell
  Other components of the Learning Shell interact via the Controller object
}

interface

uses comctrls, vlaSystemTree;

type
  TSystemTreeManager = class(TObject)
  private
    sysTree: TSystemTree;
  public
    {
      Creates a system tree initialised with the current course contents
      for either a student or a teacher author
    }
    constructor Create(studentOrTeacher: integer);
    function getCourseTitle: string;
    {
      Displays system tree in a TreeView component
    }
    procedure createTreeView(treeView: TTreeView);
    {
      Updates the status of each section (and the course tree as a
      whole) in line with the status of its component topics
    }
    procedure updateSectionStatus;
    procedure getNextTopic( var VSection, VTopic: integer);
    procedure getPreviousTopic( var VSection, VTopic: integer);
    {
      Returns the string form of the current status of the input topic
    }
    function getStatusString(section, topic: integer): string;
    procedure setStatus(section, topic, status: integer);
    {
      Returns position of the next topic offering the 'mode' study
      option
    }
    function getClosestNext( var VSection, VTopic: integer; mode: string): integer;
    {
      Returns position of the closest previous topic offering the
      'mode' study option
    }
    function getClosestPrevious( var VSection, VTopic: integer; mode: string): integer;
    {
      Returns position of the closest topic offering the 'mode' study
      option
    }
    procedure getClosest( var VSection, VTopic: integer; mode: string);
    function getTopicTitle( section, topic: integer): string;
  end;

implementation
  uses vlaController, vlaSystemDictionary, Classes;

{ TSystemTreeManager }

{...}

end. //of System Tree Manager

```

F9: Sample reference model implementation – System Tree

```

unit vlaSystemTree;
{   Data structure to store and navigate the course structure   }

interface
uses comctrls, vlaSystemDictionary;

type
  TTopicNode= record
      title: string;
      status: integer;
  end;
  TSectionNode = record
      title: string;
      status: integer;
      topics: array [1..MAX_TOPICS] of TTopicNode;
      topicCount: integer;
  end;
  TSystemTree = class(TObject)
      private
          title:string;
          sections: array [1..MAX_SECTIONS] of TSectionNode;
          sectionCount: integer;
          status: integer;
          SYSTEM_TREE_DIR: string;

      public
          constructor Create ( studentOrTeacher: integer );
          function getCourseTitle: string;
          procedure nextTopic ( var section, topic: integer );
          procedure previousTopic ( var section, topic: integer );
          function getSectionTitle( section: integer ): string;
          function getTopicTitle( section, topic: integer ): string;
          function getStatus( section, topic: integer ): integer;
          procedure setStatus( section, topic, status: integer);
          procedure createTreeView( TreeView: TTreeView );
          procedure saveTree;
          procedure updateSectionStatus;
          procedure updateStatus;

  end;

implementation
uses sysutils, Dialogs, LMDMSG, vlaSystemUtilities;

{ TSystemTree }

constructor TSystemTree.Create( studentOrTeacher: integer );
var
  infile: text;
  i, j: integer;
  line: string ;
begin
  if studentOrTeacher = vSTUDENT then
      SYSTEM_TREE_DIR := SYSTEM_MODELS_DIR
  else SYSTEM_TREE_DIR := AU_MODELS_DIR;
  sectionCount:= 0;
  for i:= 1 to MAX_SECTIONS do
      sections[i].topicCount :=0;
  try
      AssignFile(infile, SYSTEM_TREE_DIR + 'contents.txt');
      Reset (infile);
      readln(infile,title);
      readln(infile,status);
      readln(infile, line);           // throw away SECTION marker
  
```

```

i:=1;
while not eof(infile) do begin
  inc(sectionCount);
  readln(infile,sections[i].title);
  readln(infile, sections[i].status);
  j:=1;
  readln(infile,line);
  repeat
    inc(sections[i].topicCount);
    sections[i].topics[j].title :=line;
    readln(infile, sections[i].topics[j].status);
    readln(infile,line);
    inc(j);
  until ((line='SECTION') or(eof(infile)));
  inc(i);
end;
close(infile);

except
if studentOrTeacher = vSTUDENT then
  tellUser( 'System Tree', 'Read error. Could not create!', true, true )
else
  tellUser( 'System Tree', 'Read error. Could not create!', true, true, true );

end;

end;

{ Reads System Tree into Delphi TreeView object }
{ including current status of each node }
procedure TSystemTree.createTreeView( TreeView:TTreeView );
var
  i,j: integer;
  MyTreeNode1, MyTreeNode2: TTreeNode;
  //items: TTreeNodes;
begin
  with TreeView.items do
  begin
    { remove any existirig nodes }
    Clear;

    { Add course as root node }
    MyTreeNode1 := Add(nil, title);
    MyTreeNode1.StateIndex:= status;    //-1

    for i:=1 to sectionCount do begin
      {add section to course}
      MyTreeNode1 :=TreeView.items[0];
      MyTreeNode2:=AddChild(MyTreeNode1,intToStr(i)+' '+sections[i].title);
      MyTreeNode2.StateIndex:=sections[i].status;
      {Add topics to section}
      for j:= 1 to sections[i].topicCount do begin
        MyTreeNode1 := AddChild(MyTreeNode2,intToStr(i)+
          '+intToStr(j)+' '+sections[i].topics[j].title);
        MyTreeNode1.StateIndex:=sections[i].topics[j].status;
      end
    end;
  end;
end;

...

{ Return the section topic values of the next topic or else vEND_OF_COURSE }
procedure TSystemTree.nextTopic(var section, topic: integer);
begin
  if topic< sections[section].topicCount
  then inc(topic)

```

```

        else if section < sectionCount
            then begin
                inc(section);
                topic:= 1;
            end
        else section:= END_OF_COURSE;
end;

procedure TSystemTree.previousTopic( var section, topic: integer );
begin
    if topic >1
        then dec(topic)
    else if section >1
        then begin
            dec(section);
            topic:= sections[section].topicCount;
        end
    else section:= START_OF_COURSE;
end;

procedure TSystemTree.saveTree;
var
    outfile:text;
    i, j: integer;
begin
    try
        AssignFile(outfile, SYSTEM_TREE_DIR + 'contents.txt');
        Rewrite (outfile);
        writeln(outfile,title);
        writeln(outfile, status);
        for i:=1 to sectionCount do begin
            writeln(outfile, 'SECTION');// insert SECTION marker
            writeln(outfile,sections[i].title);
            writeln(outfile, sections[i].status);
            for j:= 1 to sections[i].topicCount do begin
                writeln(outfile,sections[i].topics[j].title);
                writeln(outfile, sections[i].topics[j].status);
            end
        end;
        close(outfile);

    except
        LMDMessageDlg("Write error!", mtConfirmation, [mbYes], 0);
    end;

end;

procedure TSystemTree.setStatus(section, topic, status: integer);
begin
    sections[section].topics[topic].Status:= status;
end;

{ Update section status to bring in line with the status }
{ of its subtopics }
procedure TSystemTree.updateSectionStatus;
var
    i,j, sumStatus:integer;
begin
    for i:= 1 to sectionCount do begin
        sumStatus :=0;
        for j:=1 to sections[i].topicCount do
            sumStatus := sumStatus+sections[i].topics[j].status;
        if (sumStatus*10)/sections[i].topicCount = vRED*10
            then sections[i].status:=vRED
        else if (sumStatus*10)/sections[i].topicCount = vGREEN*10
            then sections[i].status:= vGREEN
        else sections[i].status:= vAMBER;
    end;
end;

```



```
end;

end; //of update section status

procedure TSystemTree.updateStatus; // of entire course
var
  i,j, sumStatus:integer;
begin
  sumStatus :=0;
  for i:= 1 to sectionCount do begin
    sumStatus := sumStatus+sections[i].status;
  if (sumStatus*10)/sectionCount = vRED*10
    then status:=vRED
  else if (sumStatus*10)/sectionCount = vGREEN*10
    then status:= vGREEN
    else status:= vAMBER;
  end;
end;

end; //of update course status

end.
```

F10: Sample learning component implementation

```

unit vlaMainComponentForm:
{   Template for the base form of all learning components. This is copied from
{   the repository to allow free editing of default properties and methods
}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, vlaHelpBtn, ComCtrls;

type
  TComponentForm = class(TForm)
    vlaHelpBtn1: TvlaHelpBtn;
    StatusBar1: TStatusBar;
    procedure vlaHelpBtn1Click(Sender: TObject);
    procedure FormPaint(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    private
      resourcePath: string;
    public
      debug:boolean;
      procedure DisplayHint(Sender:TObject);
  end;

var
  ComponentForm: TComponentForm;

implementation

uses vlaController, vlaSystemUtilities, vlaSystemDictionary;

{$R *.DFM}

-----
EVENT HANDLERS
-----

procedure TComponentForm.vlaHelpBtn1Click(Sender: TObject);
begin
  vlaHelpBtn1.Execute(self);
end;

procedure TComponentForm.FormPaint(Sender: TObject);
begin
  if controller.openInGeneric(self.name) then
    color := controller.getColourScheme(mdGENERIC)
  else Color:= controller.getColourScheme(controller.getMode);
end;

procedure TComponentForm.FormShow(Sender: TObject);
var
  output : textFile;
  filepath: string;
  Save_Cursor:TCursor;
begin
  // Show hourglass cursor
  Save_Cursor := Screen.Cursor;
  Screen.Cursor := crHourglass;

  //get resource filepath
  filepath := controller.getFilePath(self.name);

```

```
//load resource location
try try
    AssignFile(output, filepath);
    reset(output);
    readln(output,resourcePath);
    tellUser(self.name,'Resource: '+ resourcePath, controller.debug, true);
except
    tellUser(self.name, 'Could not load resource', controller.debug, true);
end;
finally
    // Always close FILE and restore CURSOR to normal
    closeFile(output);
    Screen.Cursor := Save_Cursor;
end;
end;

procedure TComponentForm.FormCreate(Sender: TObject);
begin
    if controller.debug then
        Self.FormStyle:= fsNormal
    else Self.FormStyle:= fsStayOnTop;
    Application.OnHint := DisplayHint;
    // child forms are created here
end;

procedure TComponentForm.DisplayHint(Sender: TObject);
begin
    StatusBar1.SimpleText := GetLongHint(Application.Hint);
end;

end. //of v1aMainComponentForm
```

F11: Extramural Support – class interface and selected source code

```

unit vlaHelpSystem;

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    Db, DBCtrls, StdCtrls, DBTables, Buttons, ComCtrls, vlaSystemDictionary,
    vlaSystemUtilities, vlaAskForHelp, vlaHelpBtn;

const
    OK = 'OK';
    NOK = 'NOK';

type
    TUserStat = record
        concept: string;
        quType: string;
        rsOK, rsNo, rsRetry, rsAsk, rsGiveUp: integer;
    end;

    TExtramuralSupport = class(TForm)
        edConcept: TEdit;
        dsQuType: TDataSource;
        dsElaborate: TDataSource;
        gbUserResponse: TGroupBox;

        dsMsg: TDataSource;
        dsQuNextBefore: TDataSource;
        dsQuNextAhead: TDataSource;
        cbConcept: TDBLookupComboBox;
        dsConcept: TDataSource;
        cbQuType: TDBLookupComboBox;
        meElaboration: TMemo;
        meLinks: TMemo;
        meBooks: TMemo;
        meClipbrd: TMemo;
        reClipbrd: TRichEdit;
        meMsg: TMemo;

        procedure FormShow(Sender: TObject);
        procedure FormCreate(Sender: TObject);
        procedure FormClose(Sender: TObject; var Action: TCloseAction);
        procedure FormPaint(Sender: TObject);
        procedure FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);

        // dialog buttons
        procedure btOKClick(Sender: TObject);
        procedure btNoClick(Sender: TObject);
        procedure btRetryClick(Sender: TObject);
        procedure btAskClick(Sender: TObject);
        procedure btExitClick(Sender: TObject);
        procedure btMapClick(Sender: TObject);
        procedure btNewClick(Sender: TObject);
        procedure btCopyClick(Sender: TObject);

        procedure edConceptClick(Sender: TObject);
        procedure edConceptExit(Sender: TObject);

        procedure vlaHelpBtn1Click(Sender: TObject);
    end;

private
    qu_Type: QuType;
    userStat: TUserStat;
    userResponse: char;

```

```

underway, waiting : boolean;
checkAhead, checkPrevious: integer;
anotherQuestion, noMoreResults: boolean;
tOK, tNO, tRETRY, tASK, tEXIT, tNEW, tCOPY, tMAP: boolean;
fOK, fNO, fRETRY, fASK, fEXIT, fNEW, fCOPY, fMAP: boolean;
checkThemePrevious, checkThemeAhead: integer;
firstResponse, lastResponse: string;
responseCount: integer;
copyNotesPath, copyBooksPath, copyLinksPath: string;
AskForHelp: TAskForHelp; // message form

procedure HelpUser;
procedure prepareHelp;
procedure parseUserInput;
procedure executeAsSQL;
procedure rankResults;
procedure presentBestResultToUser;
procedure waitForUserResponse;
procedure doHousekeeping;
procedure resetUserResponse( bOK, bNo, bRetry, bAsk, bExit, bMap, bCopy, bNew: boolean);
procedure actionUserResponse;

procedure updateStatisticsDB;
procedure initialiseUserStat( selectedConcept, selectedQueryType: string );
procedure updateUserStat( bOK, bNo, bRetry, bAsk, bGiveup: boolean );
procedure presentNextBestResult;
procedure TryHelpingUserAgain;
procedure askTutor;
procedure notifyTutor;
function getSystemMsgCount: integer;
procedure setSystemMsgCount( msgCount: integer );
function getNextClosestBefore( var concept: string; var section, topic: integer ): string;
function getNextClosestAhead( var concept: string; var section, topic: integer ): string;
function getNextClosestThemeBefore( var concept: string; var section, topic: integer ): string;
function getNextClosestThemeAhead( var concept: string; var section, topic: integer ): string;
procedure copyToNotes;
procedure setupConceptList;
procedure setupQuTypelist;
procedure initialiseBtnValues;
procedure displayResult( elaboration, concept: string; section, topic: integer;
                        isFirst: boolean );

procedure initialiseHelpScreen;
procedure initialiseResultLists;
procedure recordUserResponse( OKorNOK: string );
procedure copyToLinks;
procedure copyToBooks;

public
debug: boolean;
procedure DisplayHint( Sender: TObject );
function getUsername: string;
function getSection: integer;
function getTopic: integer;
function getMyGroup: integer;
function getMsgCount: integer;
procedure setMsgCount( msgCount:integer );
procedure selectAnotherConcept( concept:string );
procedure selectConcept( concept: string );

end;

var
  ExtramuralSupport: TExtramuralSupport;

implementation

uses viaMsgData, viaConceptMap, viaController;

{$R *.DFM}

```

```

{      Initialisation routines      }

procedure TExtramuralSupport.FormCreate(Sender: TObject);
begin
    debug:= false; //true
    if controller.debug then
        Self.FormStyle:= fsNormal
    else Self.FormStyle:= fsStayOnTop;
    Application.OnHint := DisplayHint;
    initialiseBtnValues;
    color:= controller.getColourScheme( mdSYSTEM );
end;

//called to self prompt before showing Extramural Support screen
procedure TExtramuralSupport.selectConcept(concept: string);
begin
    with edConcept do begin
        ReadOnly:=false;
        Text:= concept;
        ReadOnly:=true;
    end;
    tellUser( self.name, 'Concept = '+ concept, LOG_IT, false );
end;

procedure TExtramuralSupport.Form Show(Sender: TObject);
begin
    tellUser( self.name, 'Show' , LOG_IT, false );
    parseCaption( self );
    copyNotesPath:= controller.copyTo( vNOTES );
    copyBookspath:= controller.copyTo ( vBOOKS );
    copyLinksPath:= controller.copyTo( vLINKS );
    tellUser( self.name, copyNotesPath + ' ; '+copyBooksPath + ' ; '+copyLinksPath, LOG_IT, false );
    setupQuTypeList;
    initialiseHelpScreen;
    initialiseResultLists;
end;

procedure TExtramuralSupport.setupConceptList;
begin
    with dmMsgData.quConcept do begin
        close;
        Params[0].value:= getSection;
        params[1].value:= getTopic;
        open;
    end;
    with cbConcept do begin
        ListSource:= dsConcept;
        KeyField := 'Concept';
        ListField:= 'Concept';
        enabled:= true;
        KeyValue:= edConcept.Text;
        enabled:= false;
    end;
end;

procedure TExtramuralSupport.setupQuTypeList;
begin
    cbQuType.Enabled := true;
    cbQuType.KeyValue:= 'qWhat';
    cbQuType.DropDown;
end;

procedure TExtramuralSupport.initialiseBtnValues;
begin
    tOK:=true;
    tNO :=true;
...

```

```

    fNEW:= false;
end;

procedure TExtramuralSupport.initialiseResultLists;
begin
    with meElaboration do begin
        Clear;
        modified := false;
    end;
    with MeLinks do begin
        Clear;
        modified := false;
        visible :=false;
    end;
    with meBooks do begin
        Clear;
        modified := false;
        visible :=false;
    end;
end;

procedure TExtramuralSupport.initialiseHelpScreen;
begin
    underway:= false;
    cbQuType.SetFocus;
    cbQuType.DropDown;
    resetUserResponse( tOK, false, false, false, tEXIT, false, false, false);
    btOK.hint:='Submit the question';
end;

{ INTERACTION WITH USER }

procedure TExtramuralSupport.edConceptExit(Sender: TObject);
begin
    cbQuType.SetFocus;
    edConcept.Enabled:=false;
    resetUserResponse(tOK, false,false, false, tEXIT,false, false, false);
end;

procedure TExtramuralSupport.btOKClick(Sender: TObject);
begin
    if not underway then
        HelpUser
    else begin
        userResponse:='O';
        actionUserResponse;
    end
end;

procedure TExtramuralSupport.btNoClick(Sender: TObject);
begin
    userResponse:= 'N';
    actionUserResponse;
end;

...

procedure TExtramuralSupport.btExitClick(Sender: TObject);
begin
    close;
end;

procedure TExtramuralSupport.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    if underway then begin
        userResponse:= 'X';
        actionUserResponse;
    end;
end;

```

```

end;
end;

procedure TExtramuralSupport.actionUserResponse;
var
  btn: string ;
begin
  waiting :=false;
  inc(responseCount);
  meMsg.Clear;

  case userResponse of

    'A': begin
      btn:= 'Ask';
      recordUserResponse(NOK);
      askTutor;
      resetUserResponse( fOK, fNO, fRETRY, fASK, tEXIT, tMAP, tCOPY, tNEW);
    end;

    'C': begin
      btn:= 'Copy';
      if meLinks.Modified then
        copyToLinks;
      if meBooks.Modified then
        copyToBooks;
      if meElaboration.modified then
        copyToNotes;
      initialiseResultLists;
      resetUserResponse( fOK, fNO, fRETRY, fASK, tEXIT, fMAP, fCOPY, tNEW);
      dec(responseCount);
    end;

    'M': begin
      btn:= 'Map';
      ConceptMap := TConceptMap.create(Application);
      with ConceptMap do begin
        setConcept(edConcept.Text);
        showModal;
        if modalResult<> 5 then /no helpful concept found
          resetUserResponse( fOK, fNO, fRETRY, tASK, tEXIT, tMAP,
            tCOPY, tNEW );
      end;
    end;

    'N': begin
      btn:= 'No';
      notifyTutor;
      if noMoreResults then
        resetUserResponse( fOK, fNO, fRETRY, tASK, tEXIT, tMAP, tCOPY, tNEW )
      else
        resetUserResponse( fOK, fNO, tRETRY, tASK, tEXIT, tMAP, tCOPY, tNEW );
      recordUserResponse(NOK);
    end;

    'O': begin
      btn:= 'OK';
      resetUserResponse( fOK, fNO, fRETRY, fASK, tEXIT, tMAP, tCOPY, tNEW );
      recordUserResponse( OK );
    end;

    'R': begin
      btn:= 'RETRY';
      recordUserResponse( NOK );
      TryHelpingUserAgain;
      if noMoreResults then
        resetUserResponse( fNO,tNO, fRETRY, tASK, tEXIT, tMAP, tCOPY, tNEW )

```



```

        else
            resetUserResponse( tOK, tNO, tRETRY, fASK, fEXIT, fMAP, fCOPY, fNEW );
        end;
    'W': begin
        btn:= 'New';
        cbQuType.Enabled:=true;
        anotherQuestion:= true;
        initialiseHelpScreen;
        cbQuType.SetFocus;
        cbQuType.DropDown;
    end;
    'X': begin
        btn:= 'Exit';
        dec( ResponseCount );
        updateStatisticsDB;
        doHousekeeping;
    end;
end;
tellUser( self.name, 'User action: '+ btn, LOG_IT , false );
end;

procedure TExtramuralSupport.resetUserResponse( bOK, bNo, bRetry, bAsk, bExit, bMap,
                                                bCopy, bNew: boolean);
begin
    btOK.enabled := bOK;
    ...
    btNew.Enabled:= bNew;
end;

{   end of INTERACTION WITH USER   }
{   HIGH-LEVEL OPERATIONS           }

procedure TExtramuralSupport.HelpUser;
begin
    prepareHelp;
    parseUserInput;
    executeAsSQL;
    presentBestResultToUser;
    waitForUserResponse;
end;

procedure TExtramuralSupport.TryHelpingUserAgain;
begin
    presentNextBestResult;
    waitForUserResponse;
end;
{   end of HIGH-LEVEL OPERATIONS   }
{   ENGINE ROOM                     }

procedure TExtramuralSupport.prepareHelp;
begin
    underway:=true;
    noMoreResults:= false;
    firstResponse:="";
    lastResponse:="";
    responseCount:=0;
    if not anotherQuestion then
        meElaboration.Clear;
    anotherQuestion := false;
    initialiseUserStat(edConcept.Text,cbQuType.Text);
    cbQuType.Enabled:=false;
    checkPrevious:=0;
    checkAhead:=0;
end;

```

```

    btOK.Hint:= 'Accept the answer';
end;

procedure TExtramuralSupport.parseUserInput;
var
    qtp :string;
begin
    qtp:= cbQuType.KeyValue;
    if qtp = 'qHow' then qu_Type:= qHow
    else if qtp = 'qWhat' then qu_Type:= qWhat
    else if qtp = 'qWhere' then qu_Type:= qWhere
    else if qtp = 'qWho' then qu_Type:= qWho
    else qu_Type:= qWhy;
    tellUser( self.name, qtp, LOG_IT, false );
end;

procedure TExtramuralSupport.executeAsSQL;
begin
    with dsElaborate do begin
        DataSet.Close;
        case ( qu_Type ) of
            qWhat: begin
                DataSet:= dmMsgData.quWhat;
                dmMsgData.quWhat.Params[0].Value:= edConcept.Text;
                dmMsgData.quWhat.Params[1].Value:= getSection;
                dmMsgData.quWhat.Params[2].Value:= getTopic;
            end;
            qWhere: begin
                DataSet:= dmMsgData.quWhere;
                dmMsgData.quWhere.Params[0].Value:= edConcept.Text;
                dmMsgData.quWhere.Params[1].Value:= getSection;
                dmMsgData.quWhere.Params[2].Value:= getTopic;
            end;
            qWho: begin
                DataSet:= dmMsgData.quWho;
                dmMsgData.quWho.Params[0].Value:= edConcept.Text;
                dmMsgData.quWho.Params[1].Value:= getSection;
                dmMsgData.quWho.Params[2].Value:= getTopic;
            end;
        end;
        Dataset.open;
    end;
end;

procedure TExtramuralSupport.presentBestResultToUser;
var
    quResult:string;
begin
    with dsElaborate.DataSet do begin
        First;
        quResult:= Fields[0].AsString;
        displayResult( quResult, edConcept.text,getSection,getTopic, true )
    end;
end;

procedure TExtramuralSupport.presentNextBestResult;
var
    concept, ElabText: string;
    sec, top: integer;
    isCncpt: boolean ;
begin
    concept:= edConcept.Text;
    sec:= getSection;
    top := getTopic;

    //first check that no more results in current topic

```

```

with dsElaborate.DataSet do begin
    Next;
    if not eof then begin
        ElabText:= Fields[0].AsString;
        displayResult( ElabText, concept, sec, top, true );
        Exit; //the procedure
    end;
end ;

//if no more results in current topic then look in
//previous topics
ElabText:= getNextClosestBefore( concept, sec, top );

//if still no more results then look in upcoming topics
if ElabText = " then
    ElabText := getNextClosestAhead( concept, sec, top );
if not noMoreResults then
    displayResult( ElabText concept, sec, top, false );
end;

procedure TExtramuralSupport.displayResult( elaboration, concept: string; section, topic: integer;
                                           isFirst :boolean );

var
    header: string;
begin
    meElaboration.lines.append("");
    if elaboration <> " then begin //ElabText
        meElaboration.modified := true;
        meElaboration.lines.append( "" + concept+""; Section '+ intToStr(section)+'+intToStr(topic));
        meElaboration.lines.Append(elaboration);
        if qu_type = qWhere then begin
            meLinks.modified := true;
            meLinks.Lines.Append(elaboration)// for saving as links
        end
        else if qu_type = qWho then begin
            eBooks.modified := true;
            meBooks.Lines.Append(elaboration)// for saving as books
        end;
    end
    else begin
        if isFirst then begin
            meMsg.lines.Append('- NO USEFUL ANSWER FOUND IN SECTION '+
                               intTo(section)+'+intToStr(topic));
            meMsg.lines.Append( '- SELECT "RETRY" FOR A BROADER SEARCH. ');
        end
        else begin
            noMoreResults:=true;
            meMsg.lines.Append( '- NO FURTHER USEFUL ANSWERS FOUND');
            meMsg.lines.Append( '- FOR FURTHER HELP SELECT A RELATED CONCEPT
                                FROM THE CONCEPT MAP');
            meMsg.lines.Append( '- OR TRY ASKING THE TUTOR. ');
        end;
    end;
end;

function TExtramuralSupport.getNextClosestBefore( var concept: string; var section, topic: integer): string;
begin
    inc(checkPrevious);
    Result:="";
    case (qu_Type) of
        qWhat: with dmMsgData.quWhatBefore do begin
            if checkPrevious = 1 then begin
                close;
                Params[0].Value:= concept;
                Params[1].Value:= section;
                Params[2].Value:= topic;
                Params[3].Value:= section;
            end;
        end;
    end;
end;

```

```

        open;
        if not eof then begin
            First;
            concept:= Fields[0].AsString;
            section:= Fields[1].AsInteger;
            topic := Fields[2].AsInteger;
            Result := Fields[3].AsString;
        end;
    end
else begin
    Next;
    if not Eof then begin
        concept:= Fields[0].AsString;
        section:= Fields[1].AsInteger;
        topic := Fields[2].AsInteger;
        Result := Fields[3].AsString;
    end;
end;
end;
qWhere: with dmMsgData.quWhereBefore do begin
    if checkPrevious = 1 then begin
        close;
        Params[0].Value:= concept;
        Params[1].Value:= section;
        Params[2].Value:= topic;
        Params[3].Value:= section;
        open;
        if not eof then begin
            First;
            concept:= Fields[0].AsString;
            section:= Fields[1].AsInteger;
            topic := Fields[2].AsInteger;
            Result := Fields[3].AsString;
        end;
    end
else begin
    Next;
    if not Eof then begin
        concept:= Fields[0].AsString;
        section:= Fields[1].AsInteger;
        topic := Fields[2].AsInteger;
        Result := Fields[3].AsString;
    end;
end;
end;
qWho: with dmMsgData.quWhoBefore do begin
    if checkPrevious = 1 then begin
        close;
        Params[0].Value:= concept;
        Params[1].Value:= section;
        Params[2].Value:= topic;
        Params[3].Value:= section;
        open;
        if not eof then begin
            First;
            concept:= Fields[0].AsString;
            section:= Fields[1].AsInteger;
            topic := Fields[2].AsInteger;
            Result := Fields[5].AsString +', '+ Fields[4].AsString
                +', '+ Fields[6].AsString +', '+ Fields[3].AsString;
        end;
    end
else begin
    Next;
    if not eof then begin
        concept:= Fields[0].AsString;
        section:= Fields[1].AsInteger;

```

```

        topic := Fields[2].AsInteger;
        Result := Fields[5].AsString +', '+ Fields[4].AsString
                +', '+ Fields[6].AsString +', '+ Fields[3].AsString;
    end;
end;
end;
end;

function TExtramuralSupport.getNextClosestAhead( var concept: string; var section, topic: integer): string;
begin
    ...
end;

procedure TExtramuralSupport.waitForUserResponse;
begin
    waiting :=true; // initialise
    resetUserResponse( tOK, tNO, tRETRY, fASK, fEXIT, fMAP, fCOPY, fNEW );
    btOK.Hint:= 'Accept the answer' ;
end;

procedure TExtramuralSupport.doHousekeeping;
begin
    underway:= false;
    waiting := false;
end;

{ *      end of ENGINE ROOM      }

{      Routines to log user's dialogues with support system This information can then be sent as
{      anonymous statistics to help identify where query responses need improving      }

procedure TExtramuralSupport.recordUserResponse(OKorNOK: string);
begin
    if responseCount = 1 then begin
        firstResponse:= OKorNOK;
        lastResponse := firstResponse;
    end
    else
        lastResponse := OKorNOK;
end;

procedure TExtramuralSupport.updateStatisticsDB;
var
    thisMoment: TDateTime;
begin
    thisMoment:= Now;
    with dmMsgData.tbUserStats do begin
        AppendRecord( [thisMoment, edConcept.text, getSection, getTopic,
                    FirstResponse, LastResponse, ResponseCount ] );
    end;
end;

procedure TExtramuralSupport.initialiseUserStat( selectedConcept, selectedQueryType: string );
begin
    userStat.concept:= selectedConcept;
    userStat.quType:= selectedQueryType;
    userStat.rsOK:= 0;
    userStat.rsNo := 0;
    userStat.rsRetry:= 0;
    userStat.rsAsk:= 0;
    userStat.rsGiveUp :=0;
end;

procedure TExtramuralSupport.updateUserStat( bOK, bNo, bRetry, bAsk, bGiveup: Boolean );

```

```

begin
  if bOK then
    inc(userStat.rsOK);
  if bNo then
    inc(userStat.rsNo);
  if bReTry then
    inc(userStat.rsRetry);
  if bAsk then
    inc(userStat.rsAsk);
  if bGiveup then
    inc(userStat.rsGiveUp);
end;

```

End of statistics updating methods

```

procedure TExtramuralSupport.askTutor;
begin
  AskForHelp:= TAskForHelp.Create(self);
  AskForHelp.edHeader.Text:= 'Section '+ intToStr(getSection)+'.'
    +intToStr(getTopic)+ ' '+ edConcept.Text +". Please help me on this.";
  AskForHelp.ShowModal;
end;

```

```

procedure TExtramuralSupport.notifyTutor;
var
  headerText: string;
  msgCount: integer ;
begin
  try
    dsMsg.DataSet.Last;
    msgCount:= ExtramuralSupport.getMsgCount + 1;
    ExtramuralSupport.setMsgCount(msgCount);
    headerText:='Section '+ intToStr(getSection)+'.'
      +intToStr(getTopic)+ ': '+cbQuType.Text+ ' '+ edConcept.Text +". '
      +'System response was rejected.';
    dsMsg.DataSet.AppendRecord( [msgCount, controller.getUsername, TUTOR , "", 0, getSection,
      getTopic, 'Q', Now, "", "", "", headerText, meElaboration.text] );
    ModalResult:=1;
    tellUser( self.name, 'NotifyTutor system message saved.', LOG_IT, false );
  except
    tellUser( self.name, 'Error saving system message.', LOG_IT, false );
    ModalResult:=2;
  end
end;

```

```

procedure TExtramuralSupport.copyToNotes;
begin
  with meElaboration do begin
    ReadOnly:= false;
    SelectAll;
    CutToClipboard;
    ReadOnly:= true;
  end;
  with reClipbrd do begin
    Clear;
    Lines.LoadFromFile(copyNotesPath);
    PasteFromClipboard;
    Lines.SaveToFile(copyNotesPath);
  end;
end;

```

end.// of Extramural Support unit

F12: Concept Map – class interface and selected source code

```

unit vlaConceptMap;
{
    Takes inputs in the form of ordered pairs (CONCEPT, THEME) and displays them as
    a concept map i.e. a graph in which concepts are nodes and themes are arcs connecting
    related concepts. This enables the user to explore all concepts in the database related to the
    the initial concept
}

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, vlaSystemDictionary,
    vlaSystemUtilities, lmdmsg, Buttons, vlaHelpBtn ;

const
    TPOS = 50; //top of map
    MPOS = 200; //middle
    BPOS = 350; //bottom
    LPOS = 100; // leftmost concept
    LTPOS = 200 ; //leftmost theme
    CON_OFF = 150; //concept offset
    THM_OFF = 200; //theme offset
    X_OFF = 20;
    Y_OFF = 20;

type
    TConceptMap = class( TForm )
        lbVisited: TListBox; //hidden
        vlaHelpBtn1: TvlaHelpBtn;
        procedure FormCreate( Sender: TObject );
        procedure FormShow( Sender: TObject );
        procedure FormMouseDown( Sender: TObject; Button: TMouseButton;
            Shift: TShiftState; X, Y: Integer );
        procedure FormPaint( Sender: TObject );
        procedure FormClose( Sender: TObject; var Action: TCloseAction );
        procedure vlaHelpBtn1Click( Sender: TObject );
    private
        debug: boolean;
        concept: string;
        ctConcept, ctTheme: integer;
        Concepts : array [1..MAX_CONCEPTS] of TLabel;
        Themes : array [1..MAX_THEMES] of TLabel;
        Connected :array [1..MAX_CONCEPTS, 1..MAX_THEMES] of boolean;

        // initialisation
        procedure initialiseTables;
        procedure populateTables;

        // drawing and updating the graph (nodes and arcs)
        function insertConcept( concept: string ): integer;
        function insertTheme( theme: string ): integer;
        function conceptPos( concept: string ): integer;
        function themePos( theme: string ): integer;
        procedure showConnections;
        function isAbove( lab:TLabel ): boolean;
        procedure highlightSelectedConcept( lab1:TLabel );
        procedure highlightVisitedConcepts;

        //finding which concept (node) has been selected
        function SelectedConcept( X, Y: integer ): TLabel;
        procedure setSelectArea( X, Y: integer; var XMin, XMax, YMin, YMax: integer );

        //maintaining the list of previously visited concepts
        procedure populateVisitedList;
        function notVisitedBefore( concept: string ): boolean;
        procedure updateVisitedList( concept: string );
    end;
end;

```

```

        procedure saveVisitedList;

    public
        initialise graph to particular concept
        procedure setConcept( concept: string );
    end;

var
    ConceptMap: TConceptMap;

implementation

uses viaMsgData, viaHelpSystem;

{$R *.DFM}

procedure TConceptMap.FormCreate( Sender: TObject );
var
    i: integer;
begin
    debug:= true; false
    //create tables
    for i := 1 to MAX_CONCEPTS do begin
        concepts[i] := TLabel.Create( self );
        concepts[i].Font.Style:= [];
    end;
    for i := 1 to MAX_THEMES do begin
        themes[i] := TLabel.Create( self );
        themes[i].Font.Style := [];
        themes[i].Font.Color := clNavy;
    end;
end;

// concept is set by Extramural Support before show method is called
procedure TConceptMap.setConcept( concept: string );
begin
    self.concept:=concept;
end;

procedure TConceptMap.FormShow( Sender: TObject );
begin
    tellUser( self.name, 'Show', LOG_IT, false );
    initialiseTables;
    populateTables;
    populateVisitedList;
end;

procedure TConceptMap.initialiseTables; //of themes concepts and connections
var
    i, j : integer;
begin
    ctConcept:=0; // initialise concept count to 0
    ctTheme:= 0; // initialise theme count to 0
    for i := 1 to MAX_CONCEPTS do
        for j := 1 to MAX_THEMES do
            Connected[i,j] := false;
        end;
    end;
end;

procedure TConceptMap.populateTables; //with themes concepts and connections
var
    i, j : integer;
begin
    with dmMsgData.quConnected do begin
        // run query
        close;
        Params[0].Value:= concept;
        open;
    end;
end;

```



```

//load result set into tables
if not eof then begin
    First;
    i:= insertConcept( Fields[0].AsString );
    j:= insertTheme( Fields[1].AsString );
    Connected[i, j] := true;
    Next;
end;
while not eof do begin
    i:= insertConcept( Fields[0].AsString );
    j:= insertTheme( Fields[1].AsString );
    Connected[i, j] := true;
    Next;
end;
end;

end;

procedure TConceptMap.populateVisitedList;
begin
    try
        lbVisited.Items.LoadFromFile( SYSTEM_MODELS_DIR + VISITED );
    except
        tellUser( self.name, 'Could not load VISITED file', LOG_IT, debug);
    end;
end;

end;

//update graph whenever screen is redrawn
procedure TConceptMap.FormPaint( Sender: TObject );
begin
    showConnections;
    highlightVisitedConcepts;
end;

end;

{          DRAWING THE CONCEPT MAP (GRAPH)          }

//insert concept node in map and add to concept table
function TConceptMap.insertConcept( concept: string ): integer;
begin
    Result:= conceptPos( concept );
    if Result > ctConcept then begin //not already in table so
        inc( ctConcept );
        with concepts[ctConcept] do begin
            parent:= self;
            Caption:= concept;
            if ( ctConcept mod 2 ) = 1 then begin
                if (( ctConcept div 2 ) mod 2 ) = 1 then
                    top :=TPOS
                else top :=TPOS - 25;
                if ctConcept =1 then
                    Left:= LPOS
                else Left := concepts[ctConcept-2].Left + CON_OFF;
            end
            else begin
                if (( ctConcept div 2 ) mod 2 )=0 then
                    top :=BPOS
                else top :=BPOS + 25;
                Left:= concepts[ctConcept-1].Left ;
            end;
            visible := true;
        end ;
        Result:= ctConcept;
    end;
end;

end;

function TConceptMap.conceptPos( concept: string ): integer;

```

```

var
  i: integer;
begin
  i:= 1;
  while (( i<= ctConcept ) and ( concept <> concepts[i].Caption )) do
    inc( i );
  Result:= i ;
end;

//insert theme in map and add it to theme table
function TConceptMap.insertTheme( theme: string ): integer;
begin
  Result:= themePos( theme );
  if Result > ctTheme then begin //not already in table so
    inc( ctTheme );
    with themes[ctTheme] do begin
      parent:= self;
      Caption:= theme;
      top :=MPOS;
      if ctTheme = 1 then
        Left:= LTPOS
      else Left := themes[ctTheme-1].Left + THM_OFF;
      visible := true;
    end ;
    Result:= ctTheme;
  end;
end;

function TConceptMap.themePos( theme: string ): integer;
var
  i: integer;
begin
  i:= 1;
  while (( l <= ctTheme ) and ( theme <> themes[i].Caption )) do
    inc( i );
  Result:= i ;
end;

//show connecting themes (arcs) between concept nodes
procedure TConceptMap.showConnections;
var
  i, j: integer;
begin
  for i:= 1 to ctConcept do
    for j:=1 to ctTheme do
      if connected[i,j] then begin
        with self.Canvas do begin
          if isAbove( concepts[i] ) then begin
            MoveTo(concepts[i].Left+X_OFF, concepts[i].Top+Y_OFF);
            LineTo( themes[j].left+ X_OFF, themes[j].Top );
          end
          else begin
            MoveTo( concepts[i].Left+X_OFF, concepts[i].Top );
            LineTo(themes[j].left+X_OFF, themes[j].Top+Y_OFF);
          end;
        end;
      end;
    end;
  end;
end;

function TConceptMap.isAbove( lab: TLabel ): boolean;
begin
  Result := lab.Top < MPOS; // = TPOS;
end;

procedure TConceptMap.highlightSelectedConcept( labl: TLabel );
begin
  //reset label fonts

```

```

highlightVisitedConcepts;

//highlight selected node to red bold
with lab1 do begin
    Font.color := clRed;
    Font.Style:= [fsBold]
end;
end;

procedure TConceptMap.highlightVisitedConcepts;
var
    i: integer;
begin
    //set visited label fonts to Red italic
    for i :=1 to ctConcept do
        with concepts[i]do begin
            if notVisitedBefore( caption ) then
                Font.Color:= clBlack
            else Font.Color:= clRed;
            if concepts[i].Font.Style= [fsBold]
                then concepts[i].Font.Style := [fsItalic];
        end ;
    end;

{           end of map drawing routines           }

{ INTERACTION WITH USER }

procedure TConceptMap.FormMouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
var
    lab:Tlabel;
begin
    lab:= SelectedConcept( X, Y );
    if lab <> nil then begin
        highlightSelectedConcept(lab);
        if LMDMessageDlg( 'Search for help on "'+ lab.Caption +"'?',
            mtConfirmation, [mbYes, mbNo], 0 ) = mrYes
            then begin
                ExtramuralSupport.selectAnotherConcept( lab.caption );
                updateVisitedList( lab.caption );
                modalResult:= 5;
            end
            else
                highlightVisitedConcepts;
        end;
    end;

{           end of INTERACTION WITH USER           }

{           Routines to find concept that has been selected with the mouse by user           }

function TConceptMap.SelectedConcept( X, Y: integer ): TLabel;
var
    i: integer;
    XMin, XMax, YMin, YMax: integer;
begin
    setSelectArea( X, Y, XMin, XMax, YMin, YMax );
    i:= 1;
    while (( i <= ctConcept ) and not (( concepts[i].Left > Xmin ) and (concepts[i].Left <Xmax)
        and (concepts[i].Top>YMin) and ( concepts[i].Top < YMax ))) do
        inc(i);
    if i > ctConcept then
        Result:= nil
    else Result:= concepts[i];
end;

```

```

procedure TConceptMap.setSelectArea( X, Y: integer; var XMin,XMax,YMin, YMax:integer);
begin
    Xmin := X - ( X_OFF * 2 );
    XMax := X + ( X_OFF );
    YMin := Y - Y_OFF;
    YMax := Y + Y_OFF;
end;
{      End of routines to find selected concept      }

{      MAINTAINING LIST OF VISITED NODES USING HIDDEN LIST BOX COMPONENT      }

procedure TConceptMap.FormClose( Sender: TObject; var Action: TCloseAction );
begin
    saveVisitedList;
    Action:=caFree;
end;

procedure TConceptMap.updateVisitedList( concept: string );
begin
    if notVisitedBefore( concept ) then
        lbVisited.Items.Append( concept ); hidden component
end;

function TConceptMap.notVisitedBefore(concept: string): boolean;
begin
    Result := ( lbVisited.Items.IndexOf( concept ) = -1);
end;

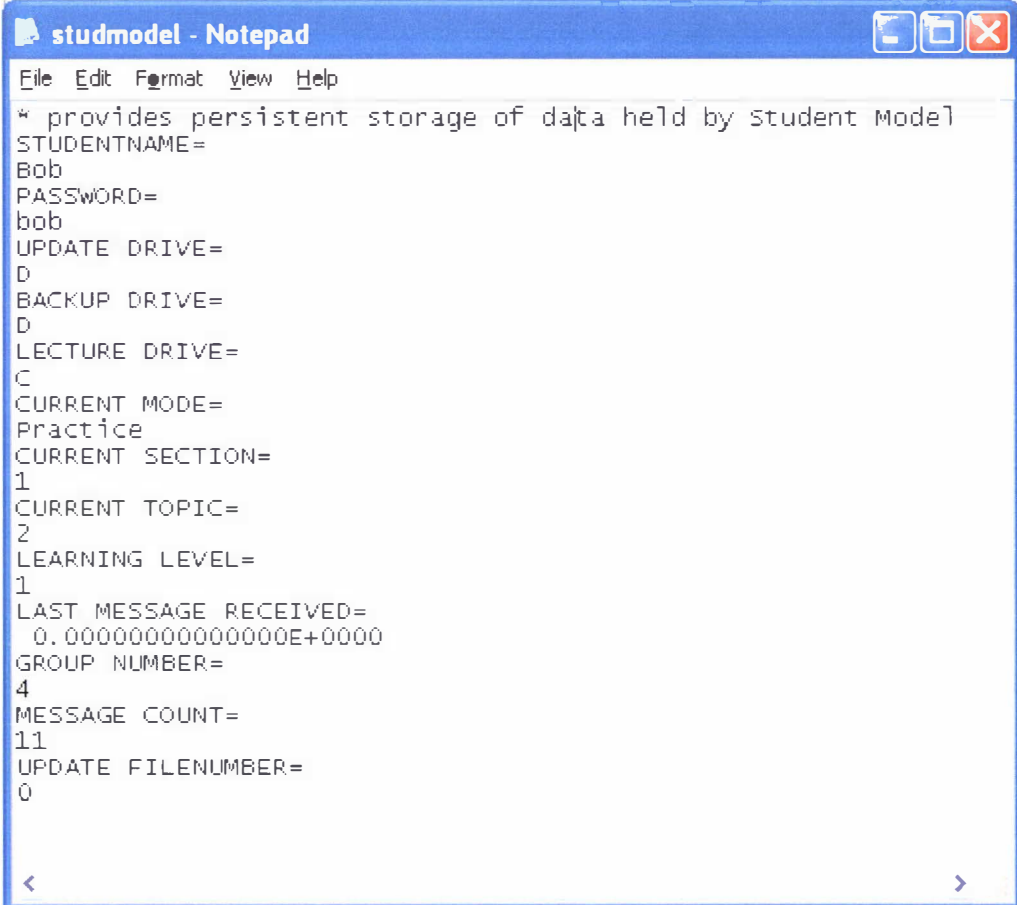
procedure TConceptMap.saveVisitedList;
begin
    try
        lbVisited.Items.SaveToFile( SYSTEM_MODELS_DIR + VISITED );
    except
        tellUser(self.name,'Could not save VISITED file', LOG)_IT, debug);
    end;
end;
{      end of MAINTAINING VISITED NODES      }
...
end. of TConceptMap

```

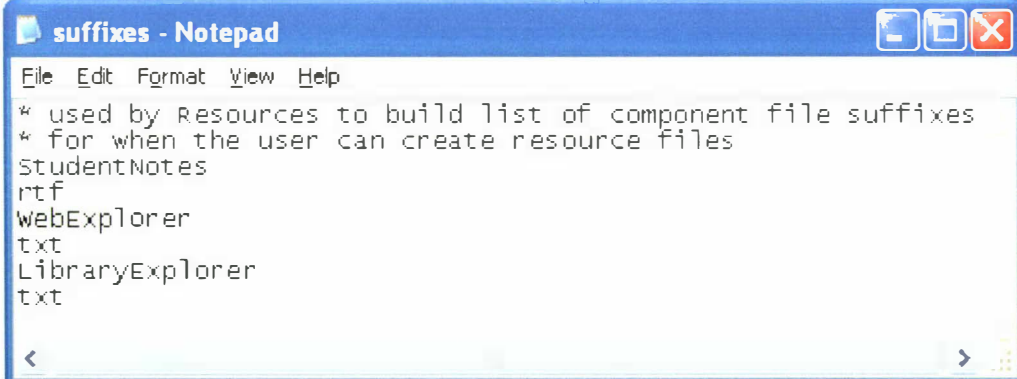
F13: System Models – Models directory and selected model files

Name	Size	Type	Modified
allmodes.txt	1KB	Text Document	09-Oct-03 12:30
alwaysavailable.txt	1KB	Text Document	10-Oct-03 12:38
alwaysthesame.txt	1KB	Text Document	14-Jul-03 14:46
clipboard.txt	1KB	Text Document	25-May-03 18:28
codes.txt	1KB	Text Document	09-Oct-03 12:31
colours.txt	1KB	Text Document	29-Sep-03 14:28
components.txt	1KB	Text Document	09-Oct-03 12:32
contents.txt	1KB	Text Document	01-Dec-03 15:36
contentsDefault.txt	1KB	Text Document	26-Sep-03 14:30
endings.txt	1KB	Text Document	09-Oct-03 12:19
gencomp.txt	1KB	Text Document	09-Oct-03 12:31
modaldisplay.txt	1KB	Text Document	09-Oct-03 12:31
sectionwide.txt	1KB	Text Document	24-Jun-03 15:43
studmodel.txt	1KB	Text Document	01-Dec-03 15:36
studmodelDefault.txt	1KB	Text Document	02-Oct-03 15:39
suffixes.txt	1KB	Text Document	13-Jun-03 18:07
sysmodel.txt	1KB	Text Document	09-Oct-03 12:32
systemsettings.txt	1KB	Text Document	10-Oct-03 13:06
systemsettingsDefault.txt	1KB	Text Document	10-Oct-03 10:01
useredit.txt	1KB	Text Document	26-Jun-03 17:47
visited.txt	1KB	Text Document	10-Oct-03 13:06
visitedDefault.txt	1KB	Text Document	10-Oct-03 12:42

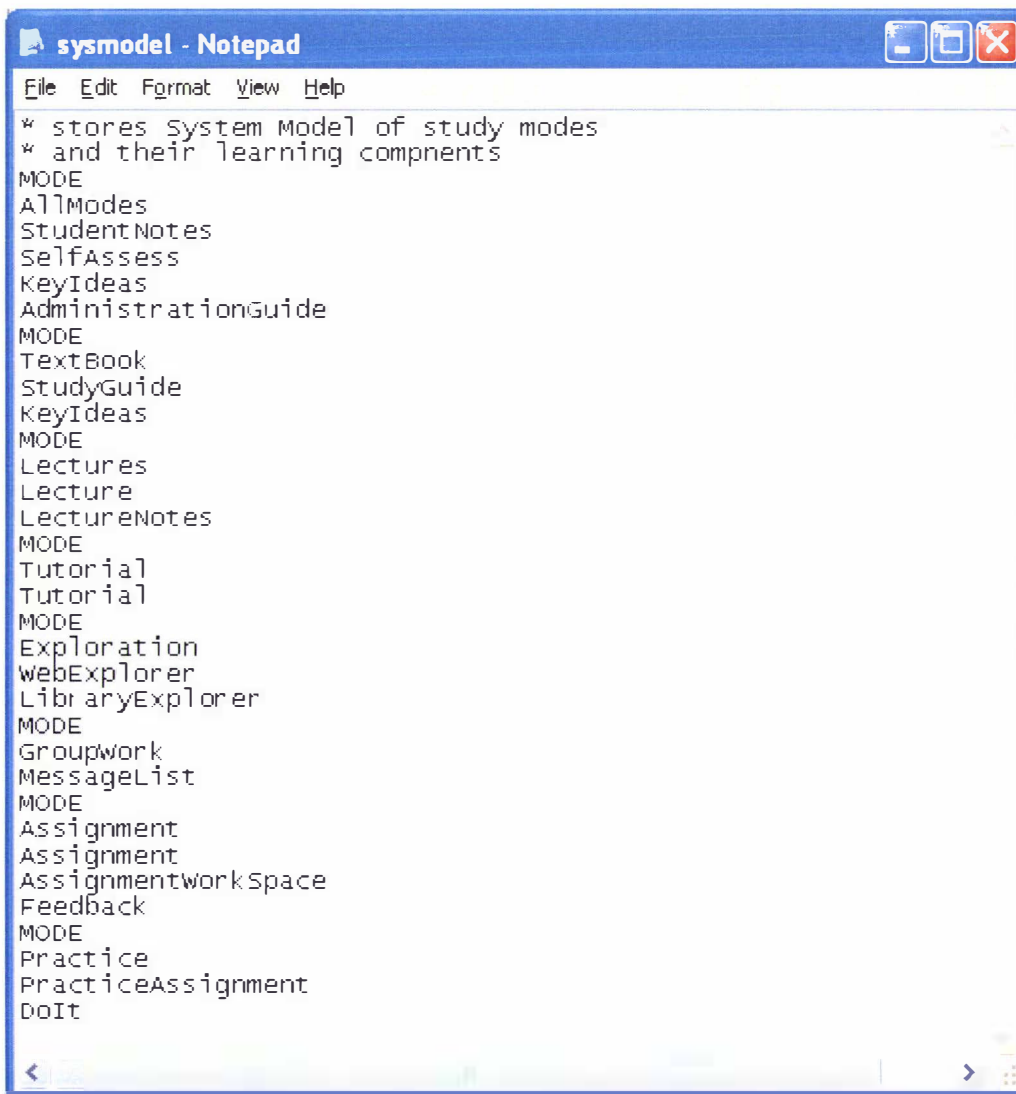
22 object(s) 3.95KB My Computer



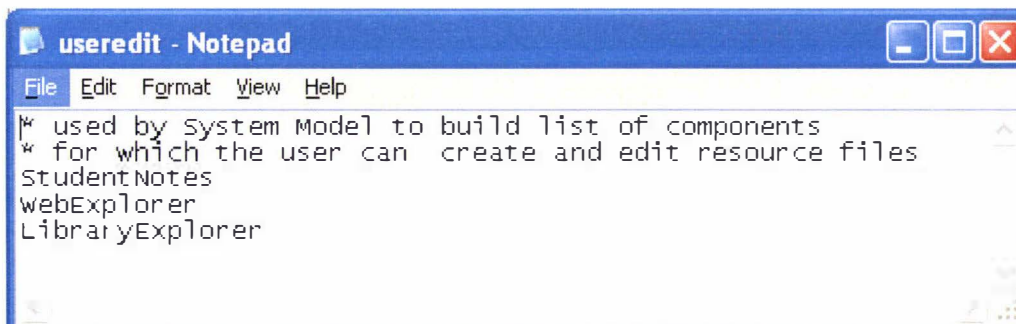
```
studmodel - Notepad
File Edit Format View Help
* provides persistent storage of data held by student Model
STUDENTNAME=
Bob
PASSWORD=
bob
UPDATE DRIVE=
D
BACKUP DRIVE=
D
LECTURE DRIVE=
C
CURRENT MODE=
Practice
CURRENT SECTION=
1
CURRENT TOPIC=
2
LEARNING LEVEL=
1
LAST MESSAGE RECEIVED=
0.0000000000000000E+0000
GROUP NUMBER=
4
MESSAGE COUNT=
11
UPDATE FILENUMBER=
0
```




```
suffixes - Notepad
File Edit Format View Help
* used by Resources to build list of component file suffixes
* for when the user can create resource files
StudentNotes
rtf
WebExplorer
txt
LibraryExplorer
txt
```



```
sysmodel - Notepad
File Edit Format View Help
* stores System Model of study modes
* and their learning compnents
MODE
AllModes
StudentNotes
SelfAssess
KeyIdeas
AdministrationGuide
MODE
TextBook
StudyGuide
KeyIdeas
MODE
Lectures
Lecture
LectureNotes
MODE
Tutorial
Tutorial
MODE
Exploration
WebExplorer
LibraryExplorer
MODE
Groupwork
MessageList
MODE
Assignment
Assignment
AssignmentWorkSpace
Feedback
MODE
Practice
PracticeAssignment
DoIt
```

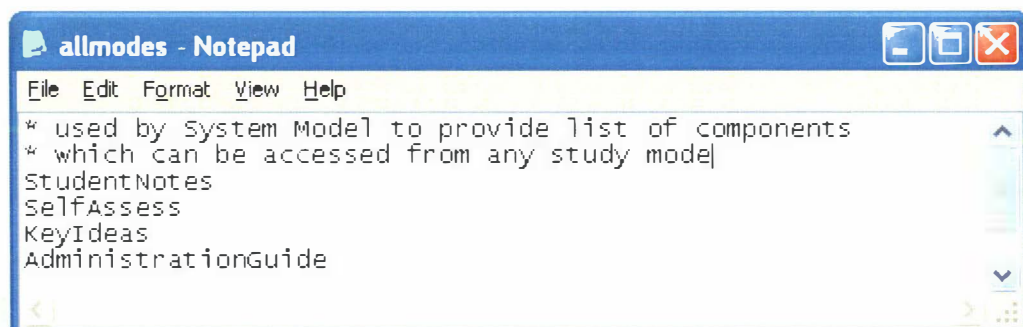


```
useredit - Notepad
File Edit Format View Help
* used by system Model to build list of components
* for which the user can create and edit resource files
StudentNotes
WebExplorer
LibraryExplorer
```



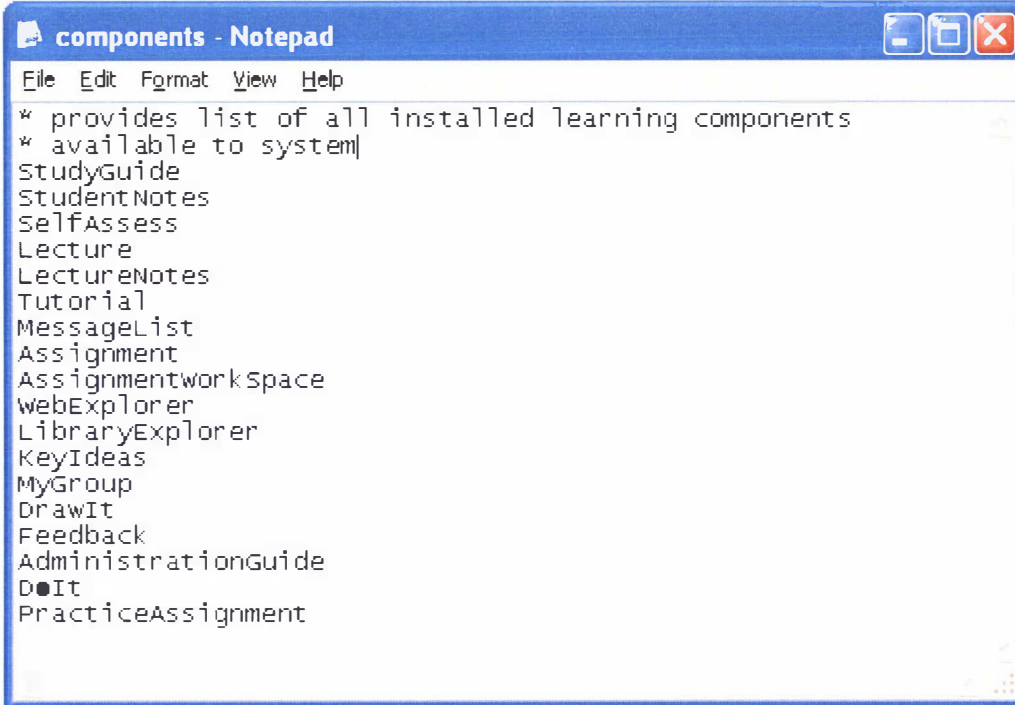
The screenshot shows a Notepad window with a blue title bar and a menu bar containing 'File', 'Edit', 'Format', 'View', and 'Help'. The text content is as follows:

```
* used by Resources Model to build table of learning
* components and their matching file naming codes
sg
StudyGuide
ce
CourseExplorer
sn
StudentNotes
st
SelfAssess
he
Help
le
Lecture
ln
LectureNotes
tu
Tutorial
me
MessageList
as
Assignment
aw
AssignmentworkSpace
we
webExplorer
lb
LibraryExplorer
ki
KeyIdeas
mg
MyGroup
dr
DrawIt
fe
Feedback
ad
AdministrationGuide
do
DoIt
pr
PracticeAssignment
```

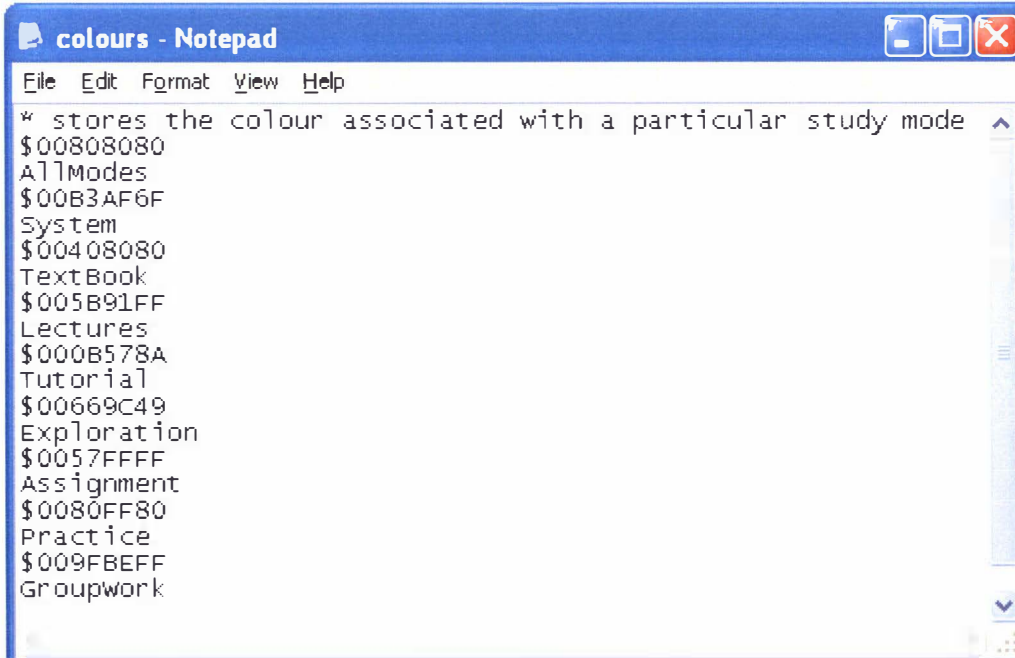


The screenshot shows a Notepad window with a blue title bar and a menu bar containing 'File', 'Edit', 'Format', 'View', and 'Help'. The text content is as follows:

```
* used by system Model to provide list of components
* which can be accessed from any study mode]
StudentNotes
SelfAssess
KeyIdeas
AdministrationGuide
```

```
components - Notepad
File Edit Format View Help
* provides list of all installed learning components
* available to system
StudyGuide
StudentNotes
SelfAssess
Lecture
LectureNotes
Tutorial
MessageList
Assignment
Assignmentwork space
webExplorer
LibraryExplorer
KeyIdeas
MyGroup
DrawIt
Feedback
AdministrationGuide
D●It
PracticeAssignment
```



```
colours - Notepad
File Edit Format View Help
* stores the colour associated with a particular study mode
$00808080
AllModes
$00B3AF6F
System
$00408080
TextBook
$005B91FF
Lectures
$000B578A
Tutorial
$00669C49
Exploration
$0057FFFF
Assignment
$0080FF80
Practice
$009FBEFF
Groupwork
```

contents - Notepad

```
File Edit Format View Help
* stores course table of contents
* number indicates completion status of each node
159.353 Human-Computer Interaction
2
SECTION
HCI Fundamentals
2
Designing for Usability
1
Modelling The User Interface
3
The User: Cognitive Issues
1
Modelling The Interaction
1
SECTION
Analysis and Conceptual Design
1
Design Issues
1
Life-cycle; Requirements analysis
1
Task description; Scenario-based modelling
1
Task Analysis, HTA
1
Lo-fi and Hi-fi Prototyping; PowerPoint
1
Conceptual Design: Review
1
SECTION
Physical Design and Implementation
1
NEW TOPIC1
```

visited - Notepad

```
File Edit Format View Help
* records every node that the user has visited
* within the Concept Map
Conceptual models
metaphor
```

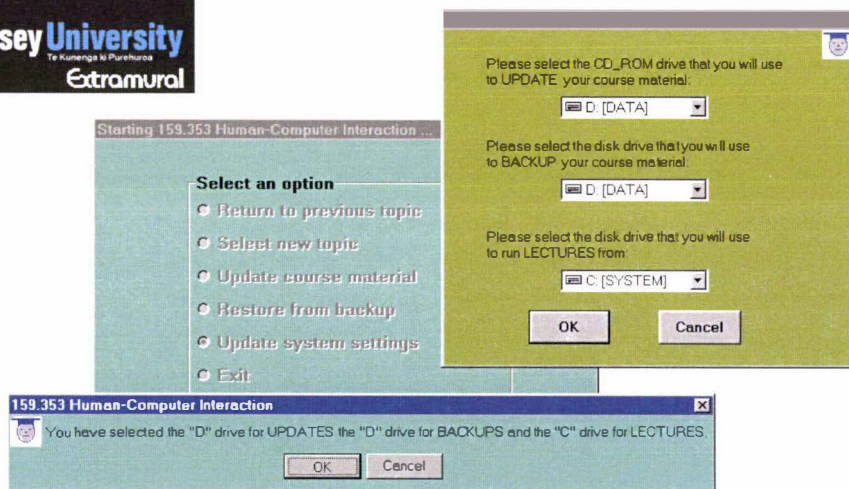
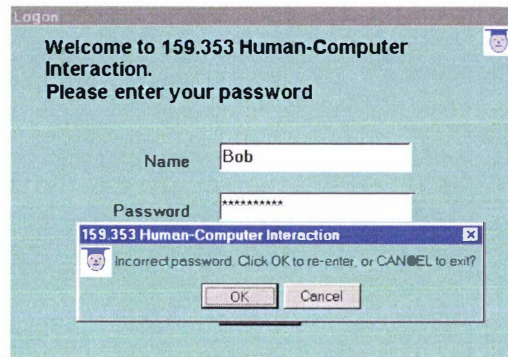
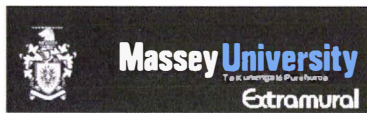
systemsettings - Notepad

```
File Edit Format View Help
http://library.massey.ac.nz/
192.168.0.1
```


Appendix G

Learning Shell – screen shots¹

¹ The screens illustrated here are in addition to those included in Chapter 7 (Figures 7.2 – 7.13).



User Options – incorrect password & update system settings



159.353 Human-Computer Interaction

Topic 1.02



Tutorial Mode

Web Tutor

[Introduction](#) [Background](#) [Links](#) [Heuristics home](#) [Close window](#)

Interactive Heuristic Evaluation Toolkit

A 'heuristic' is a guideline which is used when conducting a usability evaluation (or 'heuristic evaluation')

This interactive toolkit enables you to do either of the following:


- view a list of the suggested heuristics for a particular type of electronic device, or
- select your own heuristics for a type of device and then compare your selections against a list of suggested heuristics

Please follow the three steps below:

- Select a type of device for which to study evaluation heuristics (roll over the options with the mouse to see a description)

Device type

- [Website](#)
- [Mobile phone](#)
- [Pocket PC](#)
- [Tablet PC](#)
- [Interactive TV](#)
- [Interactive TV](#)
- [Personal organiser](#)
- [Interactive toy](#)



Interactive toys, eg. Barney, Aclimate and Aibo dog. Non-standard interaction. May respond to audio, light and touch. Gives audio and tactile output. Can show...

- If you need to consider any specific type of user or specific application area, please select the appropriate options below

Users: Application:

- Either:
 - [look at the suggested heuristics](#) for this device, or
 - [try selecting your own heuristics](#) for this device

Topic 1.02

Tutorial mode



159.353 Human-Computer Interaction
Topic 1.02

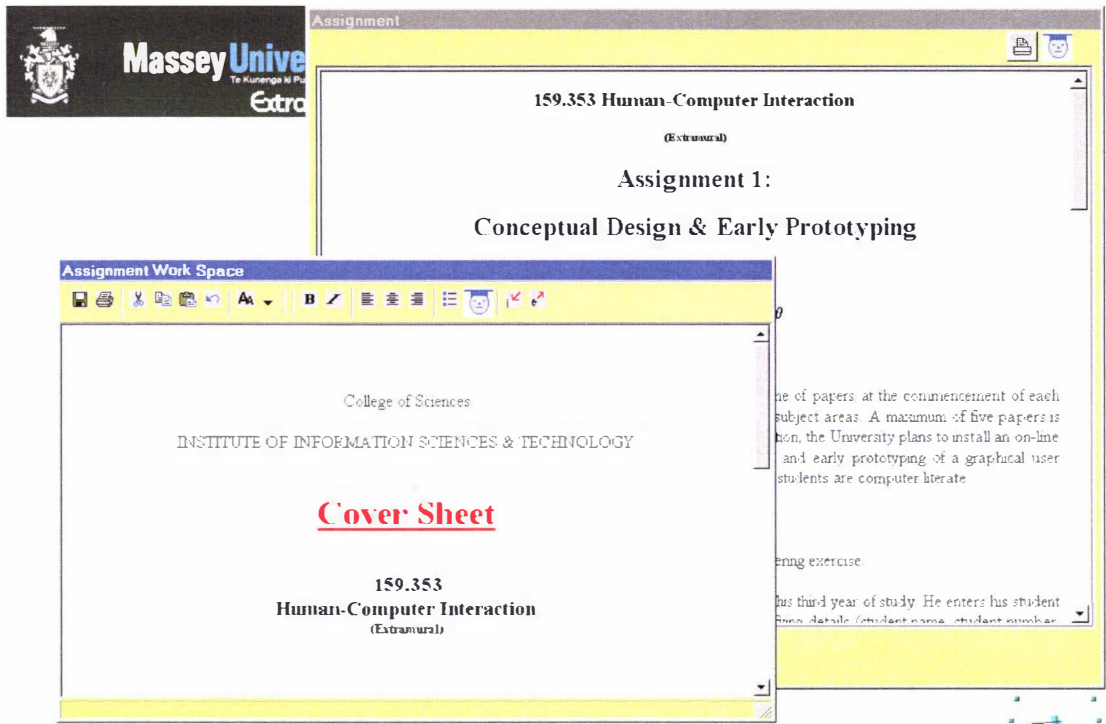
Exploration Mode

The screenshot shows a web browser window titled "Interaction Design: Chapter 2" with the URL "http://www.id-book.com/chapter2.htm". The page features a sidebar on the left with a navigation menu containing links for "Introduction", "Starters", "Chapters", "Case Studies", "Interactivities", and "Students' corner". Below the menu are buttons for "Buy the Book..." and "About the Book...". The main content area has a blue header with a table of contents (1-15) and an "Index" link. The main text is titled "Understanding and Conceptualizing Interaction" and includes a "Chapter Introduction" with links to "Web Resources", "Assignment Comments", and "Teaching Materials". The text describes the process of designing an email application and lists key concepts to be explained in the chapter.

Notes Type or paste your notes here...

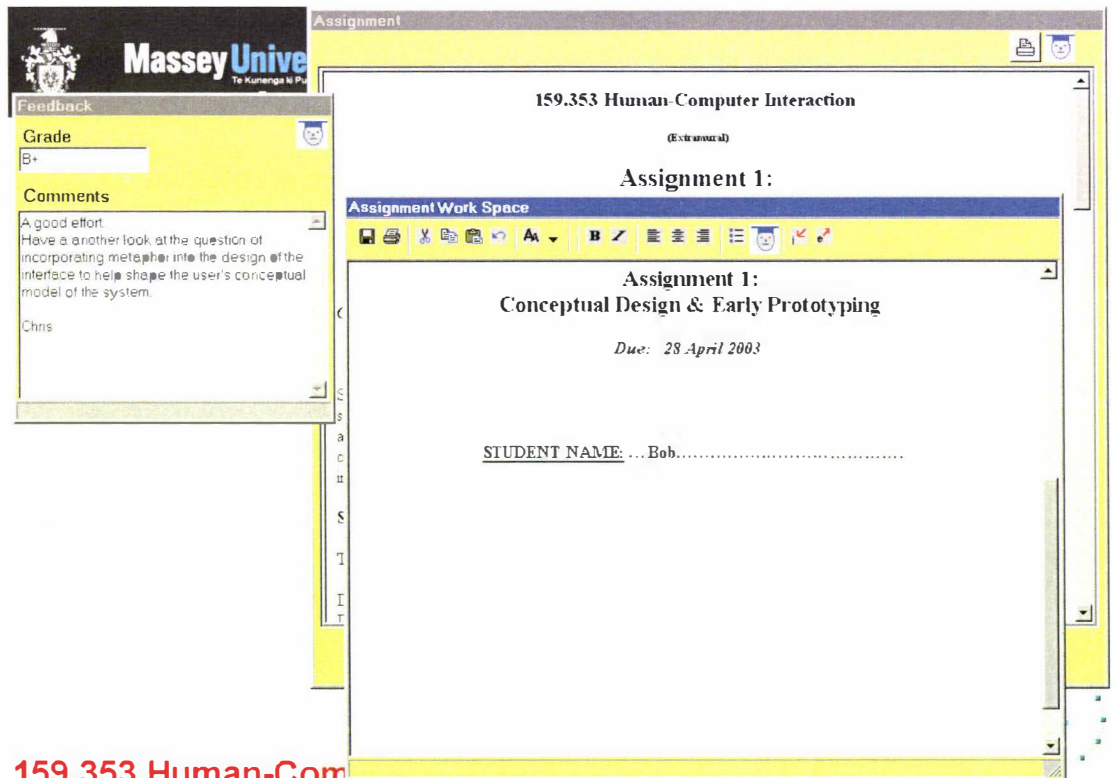
Downloading http://www.id-book.com/chapter2.htm

Exploration Mode – exploring the Web



159.353 Human-Computer Interaction Topic 1.04

Assignment Mode



159.353 Human-Computer Interaction Topic 1.04

Assignment Mode

Assignment mode – before and after assignment feedback received

Section 1.2 Practice Assignment

Do Prece Activity 2.1

WAF-enabled mobile phones have not been entirely successful. This exercise asks you to consider the assumptions which preceded their introduction.

Do Prece Activities 2.2, 2.3, 2.4

These exercises deal with various aspects of... Try at least some of them. They are designed to make you think about the issues.

Do Prece Activity 2.9

You should be familiar with at least one web employed in browser design.

159.353 Human-Computer Interaction

StudentNotes: Save work before closing?

Yes No

Student Notes

Interaction paradigms (patterns) have existed for user interfaces throughout the history of the digital computer, for example hypertext (the basis of the Web interface). The WIMP 'desktop' interface cited in **Section 2.5** has dominated over recent years. The paradigms discussed are largely experimental. One of the driving forces behind the search for new paradigms is the increasing expectation on the part of users that they should not have to be constrained to sitting in front of a desktop computer in order to interact with information. Some of the early attempts to realise this have however been far from usable, e.g. WAF-enabled mobile phones as a means of accessing the Internet (see Activity 2.1).

Interaction paradigms (patterns) have existed for user interfaces throughout the history of the digital computer, for example hypertext (the basis of the Web interface). The WIMP 'desktop' interface cited in **Section 2.5** has dominated over recent years. The paradigms discussed are largely experimental. One of the driving forces behind the search for new paradigms is the increasing expectation on the part of users that they should not have to be constrained to sitting in front of a desktop computer in order to interact with information. Some of the early attempts to realise this have however been far from usable, e.g. WAF-enabled mobile phones as a means of accessing the Internet (see Activity 2.1).

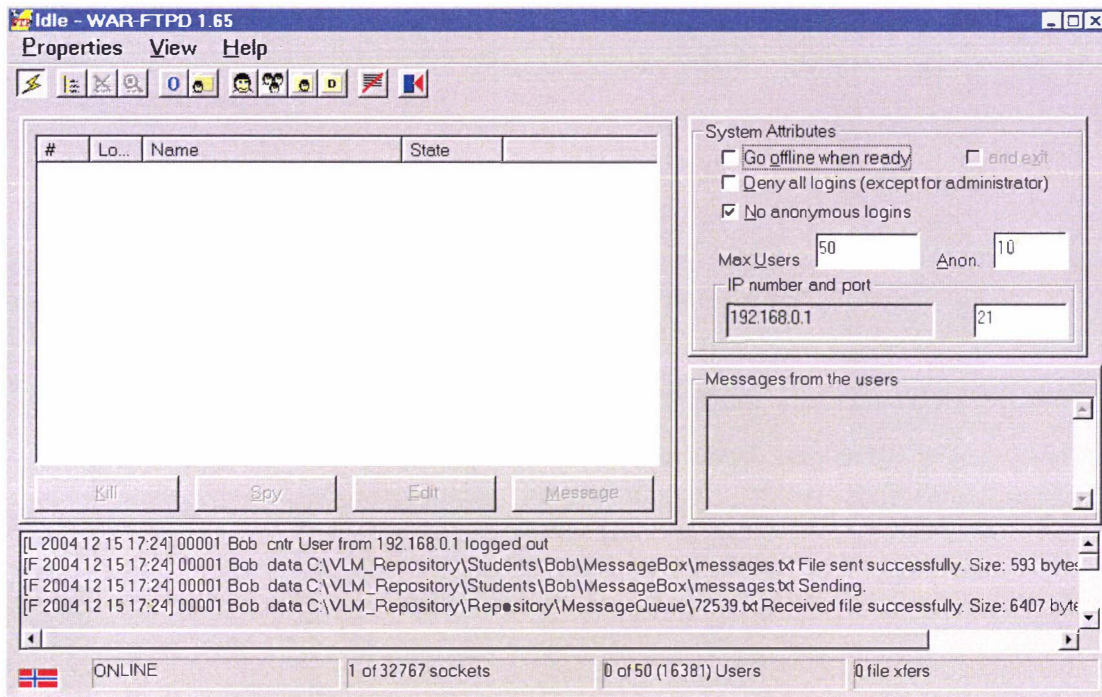
**159.353 Human-Compute
Topic 1.02**

Practice Assignment mode – showing use of Student Notes

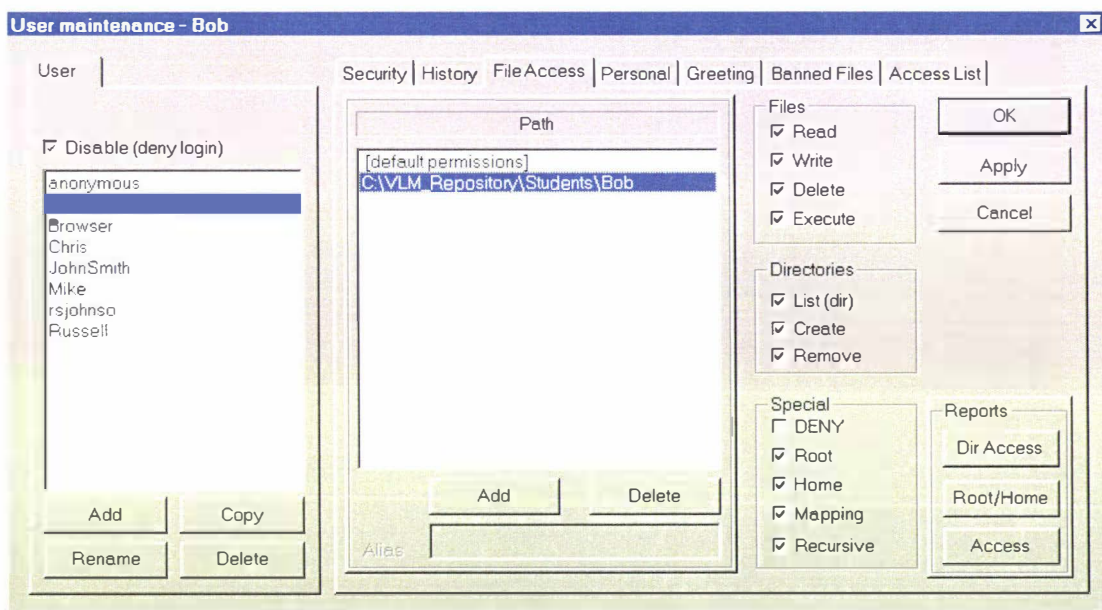
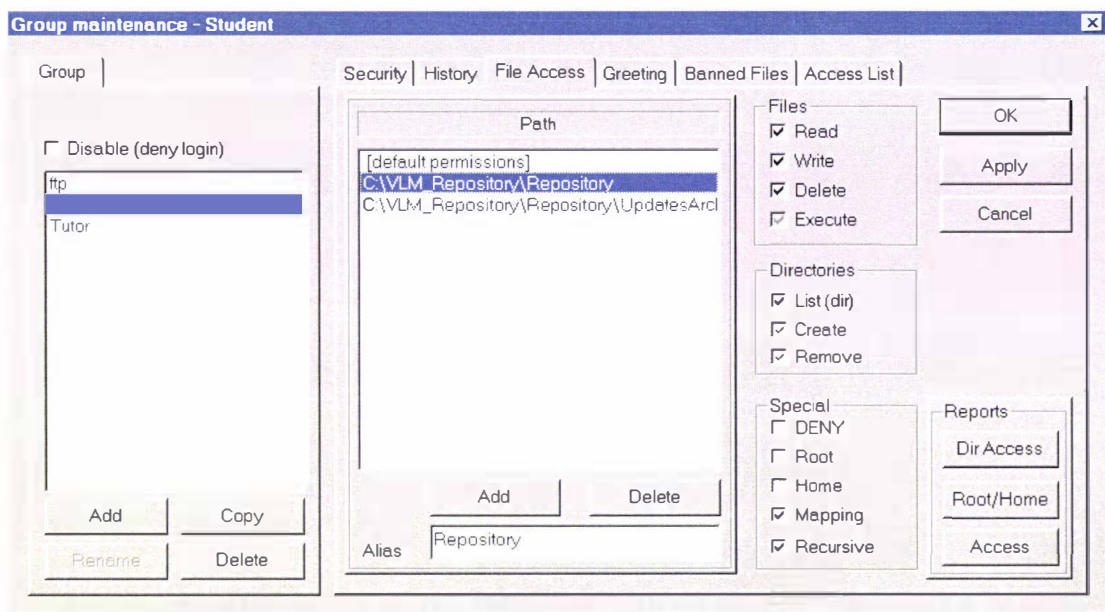
Appendix H

Communications Management

H1: FTP Server – Screen shots



Main screen after Bob logged out.



FTP Server. Defining Bob's access permissions as student

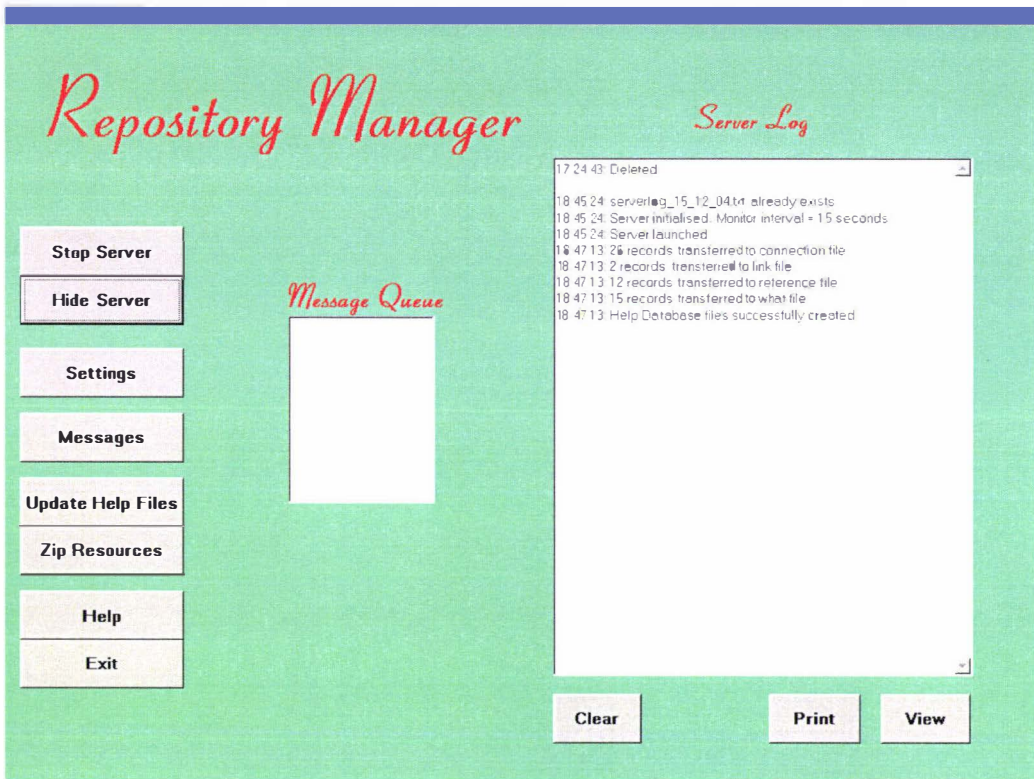
H2: Repository Manager – Screen shots

The screenshot displays the Repository Manager application window. On the left side, there is a vertical menu with buttons for 'Stop Server', 'Hide Server', 'Settings', 'Messages', 'Update Help Fi', 'Zip Resource', 'Help', and 'Exit'. The main area contains a 'Message Database' window with a table of message records. The table has the following columns: Entered, F_From, F_To, CopyTo, MsgType, ID, GroupNo, Section, and Topic. The data rows are as follows:

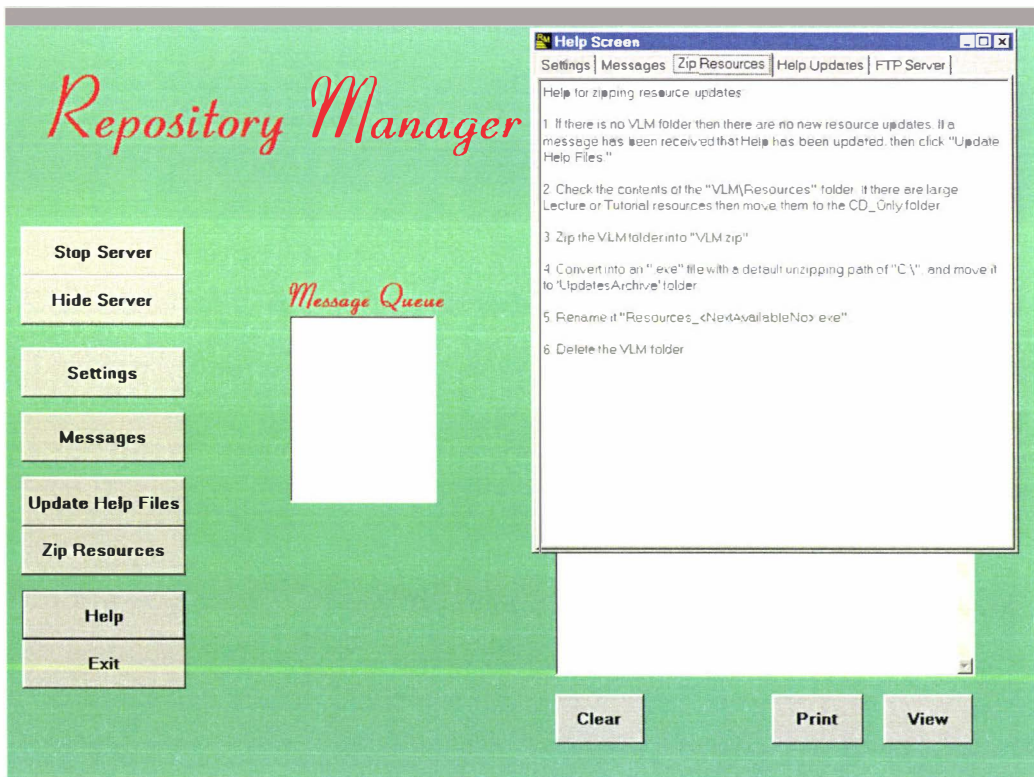
Entered	F_From	F_To	CopyTo	MsgType	ID	GroupNo	Section	Topic
03-Dec-03 13:47:05	Bob	Tutor		O	1	0	1	1
10-Oct-03 21 30 31	Mike	Tutor		O	5	4	1	1
15-Dec-04 16 54 38	Bob	Tutor		O	7	0	1	2
15-Dec-04 16 56 58	Bob	Tutor		O	8	0	1	2
15-Dec-04 16 58 17	Bob	Tutor		M	9	4	1	2
15-Dec-04 17 09 52	Bob	Tutor		O	10	0	1	2
15-Dec-04 17 11 21	Bob	Tutor		O	11	0	1	2
07-Oct-03 21 27 48	Chris	Mim		M	85	-1	1	1
07-Oct-03 21 28 49	Chris	Pom		M	86	-1	1	1
07-Oct-03 21 29 47	Chris	Bob		M	87	-1	1	1
07-Oct-03 21 30 31	Chris	Mike		M	88	-1	1	1
09-Oct-03 16 01 35	Chris	Bob	All	M	89	-1	1	2
09-Oct-03 16 01 35	Polly	Bob	MyGroup	M	89	4	1	2

At the bottom of the interface, there are three buttons: 'Clear', 'Print', and 'View'.

View of Repository message database.



Main Screen showing log after Extramural Support updated



Accessing Help to zip resources.

H3: Repository Manager – interface and selected source code

unit RepositoryManager;

```
{      Main unit Contains the code implementing the messaging protocols      }
{      message queue, and file-parsing techniques                          }
```

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, FileCtrl, ExtCtrls, vclFileManager, Db, Imdcompo, Imdclass, Imdnwgui, ComCtrls, DBTables, ImdnvS, Buttons;

type

```
TRepositoryManager = class( TForm )
  Label1: TLabel;
  ...
  ZipStarter: TLMDStarter;
  btHelp: TBitBtn;
  btUpdateHelp: TBitBtn;
  btView: TButton;
  procedure btLaunchClick( Sender: TObject );
  procedure btExitClick( Sender: TObject );
  procedure FormCreate( Sender: TObject );
  procedure TimerTimer( Sender: TObject );
  procedure btSettingsClick( Sender: TObject );
  procedure TrayIconDbClick( Sender: TObject );
  procedure btLogClick( Sender: TObject );
  procedure FormShow( Sender: TObject );
  procedure FormPaint( Sender: TObject );
  procedure btClearClick( Sender: TObject );
  procedure btPrintClick( Sender: TObject );
  procedure btSaveClick( Sender: TObject );
  procedure FileManagerFailure( Sender: TObject );
  procedure FileManagerSuccess( Sender: TObject );
  procedure btOKClick( Sender: TObject );
  procedure btMessageDBClick( Sender: TObject );
  procedure btZipClick( Sender: TObject );
  procedure btHelpClick( Sender: TObject );
  procedure btUpdateHelpClick( Sender: TObject );
  procedure btViewClick( Sender: TObject );
private
  running: boolean;
  autoSave: boolean;
  appendLog: boolean;
  isTutor: boolean;
  username: string;
  lastRcvdByUser: TDateTime;
  groupNo: integer;
  monitorInterval: integer;
  procedure processMsgList ( msgFile: string );
  procedure updateRepository( msgsOnlyFile: string );
  procedure deleteFile( fullpath: string );
  procedure checkMsgQueue;
  procedure updateUsersMessages;
  procedure saveMsgQueryResult;
  procedure cleanUpMsgQueue;
  procedure setUpServer;
  procedure saveTheLog;
  function getLogFileName: string;
  function fileExists( filename: string ): boolean;
  function getLastReceived: double;
  function processTransHeader( var tmpfile: textfile; const tnsfile: textfile;
```

```

var transType: integer ): boolean;
    procedure processLastRecvdRequest;
    function addTransHeader( var tnsfile: textfile; transType: integer ): boolean;
    procedure launchServer;
    procedure appendToLog( msgText: string );
    function getNextID: integer;
public
    procedure resetServer( monInterval: integer; viewQueue, viewLog, saveLog: string );
    procedure addToLog( msg: string );
    procedure getCurrentSettings( var monIntervalAsStr: string; var viewQueue,
        viewLog, saveLog: boolean );
end;

var
    RepositoryManager: TRepositoryManager;

implementation
uses
    vlaSystemDictionary, rpDataMod, rpServerSetup, vlaSystemUtilities,
    rpMessageDataBase, rpHelpScreen, rpUpdateHelpDB;

{$SR * DFM}

//assign directory for message-queue
procedure TRepositoryManager.FormCreate( Sender: TObject );
begin
    flbLogDir.Directory := RP_SYSTEM_LOG_DIR;
    appendLog := fileExists ( getLogFileName );
    addToLog( " );
    if appendLog then addToLog( getLogFileName + ' already exists' )
    else begin
        addToLog( getLogFileName + ' created' );
        appendLog := true;
    end;
    setupServer;
    launchServer;
end;

procedure TRepositoryManager.FormShow( Sender: TObject );
begin
    flbMsgQueue.Update;
    self.paint;
end;

procedure TRepositoryManager.FormPaint( Sender: TObject );
begin
    if meLog.Visible then
        Self.WindowState := wsMaximized
    else Self.WindowState := wsNormal;
end;

{*****
                                     INTERFACE
*****}

procedure TRepositoryManager.btLaunchClick( Sender: TObject );
begin
    launchServer;
end;

procedure TRepositoryManager.btExitClick( Sender: TObject );
begin
    cleanUpMsgQueue;
    application.Terminate;
end;

procedure TRepositoryManager.btSettingsClick( Sender: TObject );
begin
    ServerSetup.showModal;

```



```

        if ServerSetup.ModalResult = 1 then begin
            setUpServer ;
            self.paint;
        end;
end;

procedure TRepositoryManager.TrayIconDbClick( Sender: TObject );
begin
    Self.Show;
end;

procedure TRepositoryManager.btLogClick( Sender: TObject );
begin
    meLog.Clear;
end;

procedure TRepositoryManager.btClearClick( Sender: TObject );
begin
    meLog.clear
end;

procedure TRepositoryManager.btPrintClick( Sender: TObject );
begin
    rePrint.Lines.Assign ( meLog.Lines );
    rePrint.Print( 'Log Window' );
end;

procedure TRepositoryManager.btSaveClick( Sender: TObjec t);
begin
    meLog.Lines.SaveToFile( RP_SYSTEM_LOG_DIR + getLogFileName );
end;

{*****
                                     END OF INTERFACE
*****}

{*****
                                     UTILITIES
*****}

// at end of each interval
procedure TRepositoryManager.TimerTimer( Sender: TObject );
begin
    checkMsgQueue;
end;

procedure TRepositoryManager.deleteFile( fullpath: string );
begin
    addToLog( 'Deleting '+fullpath +'...' );

    with FileManager do begin
        Source := fullpath;
        Action := vDELETE;
        Execute;
    end;
end;

procedure TRepositoryManager.FileManagerFailure( Sender: TObject );
begin
    addToLog( 'Could not delete file' );
end;

procedure TRepositoryManager.FileManagerSuccess( Sender: TObject );
begin
    addToLog( 'Deleted' );
end;

```



```

*****}
procedure TRepositoryManager.setUpServer;
var
    infile: textfile;
    viewQueue, viewLog, saveLog: string;
begin
    flbMsgQueue.Directory := RP_MESSAGE_DIR;
    try
        assignFile( infile, RP_SYSTEM_DIR + 'settings.txt' );
        reset( infile );

        // monitoring interval
        readln( infile, monitorInterval );
        Timer.Interval := monitorInterval*1000;

        //view Message Queue
        readln( infile, viewQueue );
        if viewQueue = 'True' then begin
            lbQueue.Visible := true;
            flbMsgQueue.Visible := true;
            ServerSetup.cbQueue.Checked := true;
        end
        else begin
            lbQueue.Visible := false;
            flbMsgQueue.Visible := false;
            ServerSetup.cbQueue.Checked := false;
        end;

        //view activity log
        readln( infile, viewLog );
        if viewLog = 'True' then begin
            lbLog.Visible := true;
            meLog.Visible := true;
            plLog.Visible := true;
            ServerSetup.cbLog.Checked := true;
        end
        else begin
            lbLog.Visible := false;
            meLog.Visible := false;
            plLog.Visible := false;
            ServerSetup.cbLog.Checked := false;
        end;

        // automatically save log
        readln( infile, saveLog );
        if saveLog = 'True' then begin
            btSave.Visible := false;
            self.autoSave := true;
            ServerSetup.cbSave.Checked := true;
        end
        else begin
            btSave.Visible := true;
            autoSave := false;
            ServerSetup.cbSave.Checked := false;
        end;

        closefile( infile );
        addToLog( 'Server initialised. Monitor interval = '+
            intToStr( monitorInterval ) + ' seconds' );
    except
        on e: exception do begin
            addToLog( e.Message );
            addToLog( 'Could not initialise server. Try resetting.' );
        end;
    end;
end; // end of setupServer

```

```

procedure TRepositoryManager.resetServer( monInterval: integer; viewQueue,
                                         viewLog, saveLog: string );
var
  outfile: textfile;
begin
  try
    assignFile( outfile, RP_SYSTEM_DIR + 'settings.txt' );
    rewrite( outfile );
    writeln( outfile, monInterval );
    writeln( outfile, viewQueue );
    writeln( outfile, viewLog );
    writeln( outfile, saveLog );
    closefile( outfile );
  except
    on e: exception do begin
      addToLog( e.Message );
      addToLog( 'Could not reset Message Server. Contact system support.' );
    end;
  end
end;

end;

procedure TRepositoryManager.getCurrentSettings( var monIntervalAsStr: string;
                                               var viewQueue, viewLog, saveLog: boolean );
var
  infile: textfile;
  num: integer;
  vwQueue, vwLog, svLog: string;
begin
  try
    assignFile( infile, RP_SYSTEM_DIR + 'settings.txt' );
    reset( infile );
    readln( infile, num );
    monIntervalAsStr := intToStr( num );
    readln( infile, vwQueue );
    viewQueue := vwQueue='True';
    readln( infile, vwLog );
    viewLog := vwLog='True';
    readln( infile, svLog );
    saveLog := svLog='True';
    closefile( infile );
  except
    addToLog( 'Could not get current settings. Contact system support.' );
  end
end;

end;

{*****
  END OF SETUP SERVER routines
*****}

{*****
  ENGINE ROOM Transmission management routines
*****}

//message server's monitoring routine
procedure TRepositoryManager.checkMsgQueue;
var
  i: integer;
begin
  with flbMsgQueue do begin
    Update; //sort the message queue
    if items.Count<>0 then begin
      i:=0;
      while ( ( i<items.Count ) and ( items[i][1]='_' ) ) do // remove database temp
files
        inc( i );
    end;
  end;
end;

```

```

                if i< items.Count then
                    processMsgList( Items[i] );
                end;
            end;
end; // end of checkMsgQueue

//PROCESS THE MESSAGE LIST
procedure TRepositoryManager.processMsgList( msgFile: string );
var
    infile, outfile: textfile;
    tempfile : string;
    tnsType: integer;
begin
    tnsType: =0;
    try
        msgFile: = RP_MESSAGE_DIR+ msgFile;
        assignFile( infile, msgFile );
        tempfile: = REPOSITORY_DIR+ 'tempfile';
        assignFile( outfile, tempfile );

        if not ( processTransHeader( outfile, infile, tnsType ) ) then
            begin
                addToLog( 'Header Error: Wrong Receiver' );
                deleteFile msgFile );
                exit; //!!
            end;

            //process message update request
            if tnsType =tsMSG_LST then begin
                addToLog( 'Message update requested by '+ username );
                updateRepository( tempfile );
                updateUsersMessages;
            end

            //process last update received request
            else begin
                addToLog( 'Last received request from '+ username );
                processLastRecvdRequest;
            end;

        except
            addToLog( 'Could not process message list from: ' + username );
        end;
        // always delete processed file from message queue
        deleteFile( msgFile );
    end;

//update central message database
procedure TRepositoryManager.updateRepository( msgsOnlyFile: string );
var
    infile: textfile;
    line: string;
    uID: integer;
    uFrom,uTo, uCopyTo: string;
    uGroupNo, uSection, uTopic: integer;
    uMsgType: string;
    uEntered, latestEntry: TDateTime;
    uKeyword1, uKeyword2, uKeyword3: string;
    uHeader, uText: string;
    msgCount: integer;
begin
    latestEntry: =0;
    msgCount: =0;
    uID: =0;
    try
        assignFile( infile, msgsOnlyFile );

```

```

reset( infile );
readln( infile, line );
while ( ( not eof( infile ) ) and ( line <> " ) ) do begin
    uID := strToInt( line );
    readln( infile, uFrom );
    readln( infile, uTo );
    readln( infile, uCopyTo );
    readln( infile, uGroupNo );
    readln( infile, uSection );
    readln( infile, uTopic );
    readln( infile, uMsgType );
    readln( infile, uEntered );
    latestEntry := max( latestEntry, uEntered );
    readln( infile, uKeyword1 );
    readln( infile, uKeyword2 );
    readln( infile, uKeyword3 );
    readln( infile, uHeader );
    uText := "";
    readln( infile, line );
    while line <> '[END]' do begin
        uText := uText + line + #10;
        readln( infile, line );
    end;
    readln( infile, line );
    dsMsg.DataSet.AppendRecord( [uID, uFrom, uTo, uCopyTo, uGroupNo,
        uSection, uTopic, uMsgType, uEntered, uKeyword1, uKeyword2,
        uKeyword3, uHeader, uText] );
    inc( msgCount );
end;
except
    addToLog( 'Could not complete update repository from ' + username +
        '. From: '+uFrom + '. Message Type: '+ uMsgType + '. ID: '+intToStr( uID ) );
end ;
    addToLog( ntToStr( msgCount ) +' messages received from ' + username );
closeFile( infile );
end;

procedure TRepositoryManager.updateUsersMessages;
begin
    // run query on central database
    if isTutor then
        with DataMod.quUpdateTutorMsgs do begin
            close;
            Params[0].Value := lastRcvdByUser; // from processTransHeader
            Params[1].Value := username;
            Params[2].Value := username;
            open;
        end
    else
        with DataMod.quUpdateMsgs do begin
            close;
            Params[0].Value := lastRcvdByUser; // from processTransHeader
            Params[1].Value := username;
            Params[2].Value := username;
            Params[3].Value := groupNo;
            open;
        end;
        saveMsgQueryResult;
    end; //of updateUsersMessages

    //saving query result as file "newMsg.txt" in user's message directory
    procedure TRepositoryManager.saveMsgQueryResult;
    var
        outfile: textfile;
        queryFile: string;
        line: string;
        fLine: double;

```

```

i, msgCount: integer;
query: TQuery;
begin
  if isTutor then begin
    queryFile := AUTHOR_DIR+'MessageBox' +MESSAGE_LIST ;
    query := DataMod.quUpdateTutorMsgs;
  end
  else begin
    queryFile :=STUDENT_DIR+username+ 'MessageBox' +MESSAGE_LIST ;
    query := DataMod.quUpdateMsgs ;
  end;
  msgCount :=0;
  try
    AssignFile( outfile, queryFile );
    addTransHeader( outfile, tsMSG_LST );
    append( outfile );

    with query do begin
      while not Eof do begin
        for i :=0 to FieldCount-1 do begin
          if ( Fields[i].FieldName <> 'Entered' )then begin
            line:= Fields[i].AsString;
            writeln( outfile,line );
          end
          else begin
            fLine:= Fields[i].AsDateTime;
            writeln( outfile, fLine );
          end
        end;
        writeln( outfile, '[END]' );
        inc( msgCount );
        Next;
      end;
    end;
    closefile( outfile );
    addToLog( intToStr( msgCount ) +' messages transferred to '
              + username +' directory' );
  except
    addToLog( 'Write error: Unable to update messages for '+ username );
    closefile( outfile );
  end;
end; //of saveQueryResult

procedure TRepositoryManager.cleanUpMsgQueue;
begin
  DataMod.quUpdateMsgs.Close; // or else
  DataMod.quUpdateTutorMsgs.Close;
  with flbMsgQueue do begin
    update;
    while( ( items.Count<>0 ) and ( items[0][1]='_' ) ) do begin
      //showMessage( 'delete: '+items[0] );
      deleteFile ( RP_MESSAGE_DIR+Items[0] );
      update;
    end;
  end
end;

function TRepositoryManager.getLastReceived: double;
begin
  try
    with DataMod.quLastRecvd do begin
      close;
      Params[0].Value:= username;
      open;
      if not eof then begin
        First;

```

```

        Result := Fields[0].AsFloat;
    end
    else begin
        Result := 0.0;
        addToLog( 'No stored messages for "' + username + '" );
    end;
end;
except
    addToLog( 'Could not retrieve last received for ' + username );
    Result := -1.0;
end;
end;

{
    Processes the header information in 'tnsfile' returning true if it is a valid
    transmission false otherwise Returns the transmission stripped of the
    header information as 'tmpfile' and the transmission type as 'transtype'
}
function TRRepositoryManager.processTransHeader( var tmpfile: textfile;
const tnsfile: textfile; var transType: integer ): boolean;
var
    line: string;
    fline : double;
begin
    Result := true;
    reset( tnsfile );

    readln( tnsfile, line );
    if line <> REPOSITORY then // wrong receiver so
        Result := false
    else begin
        readln( tnsfile, username );
        readln( tnsfile, transType );
        readln( tnsfile, groupNo );
        if groupNo = -1 then
            isTutor := true
        else isTutor := false;
        if transType <> tsLAST_RCVD then begin // else rest of transmission irrelevant
            readln( tnsfile, fline );
            lastRcvdByUser := fline; //used by updateUsersMessges
            {
                ends header
            }

            //return transmission stripped of header
            rewrite( tmpfile );
            while not eof( tnsfile ) do
                begin
                    readln( tnsfile, line );
                    writeln( tmpfile, line );
                end;
            closefile( tmpfile )
        end;
    end;
    closefile ( tnsfile );
end;

procedure TRRepositoryManager.processLastRecvdRequest;
var
    outfile: textfile;
    lastReceived: string;
begin
    if isTutor then
        assignfile ( outfile, AUTHOR_DIR+'MessageBox' + ANSWER )
    else
        assignfile ( outfile, STUDENT_DIR +username+'MessageBox' + ANSWER );
    if addTransHeader( outfile, tsLAST_RCVD ) then begin
        DateTimeToString( lastReceived, 'dd mm yy hh mm ss', getLastReceived );
        addToLog( 'Last received request processed for ' +username+ ' : ' + lastReceived );
    end
    else
        end
end

```



```

        addToLog( 'Could not process last request for '+ username );
end;

{*****
  ADD HEADER TO FILE BEFORE TRANSMITTING WITH FORMAT
  <TO RCVR>
  <FROM SNDR>
  <MESSAGE TYPE>
  <GROUP NUMBER>
  <DATETIME STAMP OF LAST TRANSMISSION RECEIVED FROM RCVR> default = 0 ( send all )
  *****/}
function TRepositoryManager.addTransHeader( var tnsfile: textfile; transType: integer ): boolean;
begin
    try
        rewrite( tnsfile );
        writeln ( tnsfile, username );
        writeln( tnsfile, REPOSITORY );
        writeln( tnsfile, transType );
        writeln( tnsfile, DEFAULT_GROUP );
        writeln ( tnsfile, getLastReceived );
        closefile( tnsfile );
        Result := true;
    except
        addToLog ( 'Could not add header.' );
        Result = false;
    end;
end;

{*****
  end of Transmission mangement routines
  *****/}

procedure TRepositoryManager.btOKClick( Sender: TObject );
begin
    self.Hide;
end;

procedure TRepositoryManager.launchServer;
begin
    if not running then begin
        running := true;
        Timer.Enabled := true;
        btLaunch.Caption := 'Stop Server';
        btLaunch.Hint := 'Halt Message Server';
        self.Hide;
        addToLog( 'Server launched' );
    end
    else begin
        running := false;
        Timer.Enabled := false;
        btLaunch.Caption := 'Launch Server';
        btLaunch.Hint := 'Start Message Server';

        addToLog( 'Server halted' );
    end
end;

function TRepositoryManager.getNextID: integer;
begin
    with dsNextID.DataSet do begin
        Close;
        Open;
        Last;
        Result := Fields[0].AsInteger + 1;
    end;
    addToLog( 'Next ID: ' + intToStr( Result ) );
end;

```

```
end;

procedure TRepositoryManager.btMessageDBClick( Sender: TObject );
begin
    MessageDatabase.show;
end;

procedure TRepositoryManager.btZipClick( Sender: TObject );
begin
    with ZipStarter do begin
        command: = 'Winzip32.exe';
        Execute;
    end;
end;

procedure TRepositoryManager.btHelpClick( Sender: TObject );
begin
    HelpScreen.Show;
end;

procedure TRepositoryManager.btUpdateHelpClick( Sender: TObject );
begin
    RMUpdateHelpDB.prepareHelpUpdateFolder;
end;

procedure TRepositoryManager.btViewClick( Sender: TObject );
begin
    with meLog do begin
        Clear;
        Lines.LoadFromFile( RP_SYSTEM_LOG_DIR + getLogFileName );
    end;
end;

end.
```

H4: Learning Shell: Update Resources – interface and selected code

```

unit vIaUpdateResources;
{*****
 *           This unit implements the functionality for updating the Learning Shell's
 *           *****}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Lmdcompo, Lmdclass, LmdnonvS, Psock, NMFtp, ExtCtrls, Buttons,
  vIaHelpBtn, FileCtrl, vIaFileManager, vIaFolderManager;

const
  DEBUG = false ; //true://

type
  TUpdateResources = class( TForm )
    LMDStarter1: TLMDStarter;
    NMFtp1: TNMFtp;
    btCancel: TButton;
    RadioGroup1: TRadioGroup;
    btOK: TButton;
    vIaHelpBtn1: TvIaHelpBtn;
    FileList: TFileListBox;
    FileManager: TvIaFileManager;
    FolderManager: TvIaFolderManager;
    Timer: TTimer;
    tmWhenToClose: TTimer;
    procedure btCancelClick( Sender: TObject );
    procedure FormHide( Sender: TObject );
    procedure btOKClick( Sender: TObject );
    procedure FormShow( Sender: TObject );
    procedure FormKeyDown( Sender: TObject; var Key: Word;
      Shift: TShiftState );
    procedure FormClose( Sender: TObject; var Action: TCloseAction );
    procedure vIaHelpBtn1Click( Sender: TObject );
    procedure TimerTimer( Sender: TObject );
    procedure FolderManagerFailure( Sender: TObject );
    procedure FolderManagerSuccess( Sender: TObject );
    procedure tmWhenToCloseTimer( Sender: TObject );
  private
    closeNow : boolean;
    latestUpdateFile: integer;
    Save_Cursor: TCursor;
    procedure updateCaption( msg: string );
    procedure createUpdateDir;
    //update VLM & remove temp folder
    procedure updateResources;
  public
    procedure unZipResources( dr: char );
    procedure transferResources;
    procedure updateFromInternet;
    procedure updateFromDisk;
  end;

var
  UpdateResources: TUpdateResources;

```

implementation

uses LMDMSG, vlaSystemDictionary, vlaUserOptions, vlaSystemUtilities;

```
{SR * DFM}
```

```
{ TUpdateResources }
```

```
procedure TUpdateResources.unZipResources( dr: char );
```

```
var
```

```
    i: integer;
```

```
begin
```

```
    tellUser( self.name, 'Drive = '+ dr, debug );
```

```
    with FileList do begin
```

```
        Drive: = dr;
```

```
        Directory: = ExtractFileDir( drive + ':' + UPDATE_DIR + 'dummy.txt' );
```

```
        tellUser( self.name, 'Directory = '+ directory, debug );
```

```
        for i : =0 to Items.Count-1 do
```

```
            if getFileNumber( items[i] ) > UserOptions.getUpdateFileNo then begin
```

```
                tellUser( self.name, 'items[i] = '+items[i]+ ', UpdateFileNo =
```

```
                    '+intToStr( UserOptions.getUpdateFileNo ), debug );
```

```
                // Unzip resource files from update dir
```

```
                // to temp folder on c drive
```

```
                LMDStarter1.Command : = Directory + '\'+items[i];
```

```
                LMDStarter1.Execute;
```

```
                // get number of latest file update
```

```
                latestUpdateFile: = getFileNumber( items[i] );
```

```
            end;
```

```
    end ;
```

```
    { Timer waits until unzipping is finished  
    and then calls updateResources procedure }
```

```
    Timer.Enabled: =true;
```

```
end;
```

```
procedure TUpdateResources.UpdateFromDisk;
```

```
var
```

```
    drive: char;
```

```
begin
```

```
    Save_Cursor : = Screen.Cursor;
```

```
    drive : = UserOptions.getDrive( vUPDATE );
```

```
    //createUpdateDir
```

```
    if LMDMessageDlg( 'Place disk in drive "' + drive + '" and then click, "OK"',
```

```
        mtConfirmation, [mbOK, mbCancel], 0 ) = mrOK
```

```
    then begin
```

```
        Screen.Cursor : = crHourglass; // Show hourglass cursor
```

```
    try
```

```
        unZipResources( drive );
```

```
    finally
```

```
        Screen.Cursor : = Save_Cursor; // Always restore to normal
```

```
    end;
```

```
    tellUser( self.name, 'Resources have been updated.', true, false );
```

```
    updateCaption( 'Resources have been updated.' );
```

```
    end ;
```

```
    self.close;
```

```
end;
```

```
procedure TUpdateResources.UpdateFromInternet;
```

```
var
```

```
    NoUpdates: boolean;
```

```
    Save_Cursor: TCursor;
```

```
begin
```

```
    Save_Cursor : = Screen.Cursor;
```

```
    btOK.enabled: = false;
```

```
    NoUpdates: =false;
```

```

createUpdateDir;
Screen.Cursor := crHourglass; // Show hourglass cursor
try
    try
        with NMFTP1 do begin
            Host := UserOptions.getRepositoryIP;
            UserID := UserOptions.getUserName;
            Password := UserOptions.getUserPassword;
            Connect;
            ChangeDir( UPDATES_ARCHIVE );
            List;
            if FTPDirectoryList.name.Count = 0 then
                NoUpdates := true
            else
                transferResources;
        end;
        if NoUpdates then begin
            tellUser( self.name, 'No updates available', LOG_IT, false );
            updateCaption( 'No updates available' );
        end;
    except
        updateCaption( 'Connection Failed! Could not update resources.' );
        tellUser( self.name, 'Connection Failed! Could not update resources.',
            LOG_IT, true );
        NoUpdates := true;
        raise;
    end ;
finally
    if NoUpdates then begin
        Screen.Cursor := Save_Cursor; // restore to normal before closing
        closeNow := true;
    end
end;

end;

procedure TUpdateResources.transferResources;
var
    i: integer;
    dir: TFTPDirectoryList;
    fileNo: integer;
begin
    tellUser( self.name, 'Transferring resources...', LOG_IT, false );
    updateCaption( 'Transferring resources...' );
    fileNo := UserOptions.getUpdateFileNo;
    // for each file in remote update directory
    // download to local directory
    dir := NMFTP1.FTPDirectoryList;
    for i := 0 to dir.name.Count-1 do begin
        if getFileNumber( dir.name[i] ) > fileNo then
            NMFTP1.Download( dir.name[i], c:\'+UPDATE_DIR+ dir.name[i] );
        end;
    unZipResources( 'C' );
end;

procedure TUpdateResources.FormHide( Sender: TObject );
begin
    UserOptions.RadioGroup1.enabled := true;
    UserOptions.RadioGroup1.ItemIndex := 0;
    UserOptions.btOK.Enabled := true;
end;

procedure TUpdateResources.btOKClick( Sender: TObject );
var
    str: string;
begin

```

```

case RadioGroup1.ItemIndex of
  0: begin
      str := 'Connecting to university...';
      tellUser( self.name, str, LOG_IT, false );
      updateCaption( 'Please wait. '+str );
      updateFromInternet ;
    end;
  else begin
      str := 'Updating from Disk..';
      tellUser( self.name, str, LOG_IT, false );
      updateCaption( str );
      updateFromDisk;
    end;
end;

end;

procedure TUpdateResources.FormShow( Sender: TObject );
begin
  closeNow := false;
  btOK.SetFocus;
  RadioGroup1.ItemIndex := 0;
end;

procedure TUpdateResources.FormKeyDown( Sender: TObject; var Key: Word; Shift: TShiftState );
begin
  if Key = VK_Return then
    btOKClick( nil )
  else
    if ( Key=VK_F1 ) then
      vlaHelpBtn1.execute ( self );
end;

procedure TUpdateResources.FormClose( Sender: TObject; var Action: TCloseAction );
begin
  UserOptions.RadioGroup1.enabled := true;
  UserOptions.RadioGroup1.ItemIndex := 0;
  UserOptions.btOK.Enabled := true;
  Action := caFree;
end;

procedure TUpdateResources.vlaHelpBtn1Click( Sender: TObject );
begin
  vlaHelpBtn1.execute( self );
end;

procedure TUpdateResources.createUpdateDir;
var
  str: string;
begin
  Str := 'c:\'+UPDATE_DIR + 'DUMMY.TXT';
  str := ExtractFileDir( str );
  if not DirectoryExists( str ) then
    Mkdir( str );
end;

//update VLM & remove temp folder
procedure TUpdateResources.updateResources;
begin
  tellUser( self.name, 'Updating resource files...', LOG_IT, false );
  updateCaption( 'Updating resource files...' );
  try
    with FolderManager do begin
      Source := UPDATES_TEMP;
      Destination := ExtractFileDir( ROOTPATH + 'dummy.txt' );
      Action := vMOVE;
      Execute;
    end ;
  finally

```

```

        Screen.Cursor := Save_Cursor; // restore to normal before closing
        closeNow := true;
    end;
end;

//check whether resources have been unzipped
// if they have then update resource folder
procedure TUpdateResources.TimerTimer( Sender: TObject );
begin
    if FolderManager.dirExists( UPDATES_TEMP )
        and not FolderManager.isEmptyFolder( UPDATES_TEMP ) then begin
        updateResources;
        Timer.Enabled := false;
    end;
end;

procedure TUpdateResources.FolderManagerFailure( Sender: TObject );
begin
    tellUser( self.name, 'Could not update resource files.', LOG_IT, false );
    updateCaption( 'Could not update resource files.' );
end;

procedure TUpdateResources.FolderManagerSuccess( Sender: TObject );
begin
    //update studentModel
    UserOptions.setUpdateFileNo( latestUpdateFile );
    tellUser( self.name, 'Resource files updated. '+
        'File no: '+intToStr( latestUpdateFile ), LOG_IT, false );
    updateCaption( 'Resource files updated. '+
        'File no: '+intToStr( latestUpdateFile ) );
end;

procedure TUpdateResources.tmWhenToCloseTimer( Sender: TObject );
begin
    if closeNow then
        self.close;
end;

procedure TUpdateResources.updateCaption( msg: string );
begin
    self.caption := msg;
end;

end. //viaUpdateResources

```

H5: Learning Shell: Update Extramural Support – interface and selected code

```

unit vlaUpdateExtramuralSupport;
{
  This is a hidden form whose main UpdateExtramuralSupport procedure is called by the
  Controller object's UpdateExtramuralSupportFiles routine to utilise Delphi's database
  and file manipulation support to simplify updating the Extramural Support database
}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, FileCtrl, Db, vlaFolderManager, vlaFileManager, dbtables;

type
  TUpdateExtramuralSupport = class( TForm )
    DataSource: TDataSource;
    FileListBox: TFileListBox;
    FolderManager: TvlaFolderManager;
    FileManager: TvlaFileManager;
    procedure FormCreate( Sender: TObject );
  private
    procedure deleteOldRecords( table: TTable );
    procedure updateConnection;
    procedure updateLink;
    procedure updateReference;
    procedure updateWhat;
    procedure deleteFile( filepath: string );
    procedure activateTables;
  public
    procedure UpdateExtramuralSupport;
  end;

var
  UpdateExtramuralSupport: TUpdateExtramuralSupport;

implementation

{$R * DFM}

uses vlaSystemDictionary, vlaController, vlaSystemUtilities, vlaMsgData;

{ TUpdateExtramuralSupport }

procedure TUpdateExtramuralSupport.FormCreate( Sender: TObject );
begin
  FileListBox.Directory := HELP_UPDATES;
end;

procedure TUpdateExtramuralSupport.UpdateExtramuralSupport;
begin
  // if HelpUpdates folder is not empty then
  if not FolderManager.isEmptyFolder( HELP_UPDATES ) then begin
    tellUser( self.name, 'Updating Help...', true, false );
    {update each table in turn by deleting all current records
    appending all records in update file }

    deleteOldRecords( dmMsgData.tbConnection );
    updateConnection;
  end;
end;

```



```

deleteOldRecords( dmMsgData.tbWhere );
updateLink;

deleteOldRecords( dmMsgData.tbWho );
updateReference;

deleteOldRecords( dmMsgData.tbWhat );
updateWhat ;

tellUser( self.name, 'Help updated', true, false );
end
else begin
    tellUser( self.name, 'No help updates available.', true, false );
    activateTables;
end;
end;

procedure TUpdateExtramuralSupport.deleteOldRecords( table: TTable );
begin
    try
        with table do begin
            active: =false;
            Exclusive: =true;
            active: =true;
            EmptyTable;
        end ;
    except
        tellUser( self.name, 'Could not delete records from '+table.Name, true, false );
        raise;
    end;
end;

procedure TUpdateExtramuralSupport.updateWhat;

var
    infile: textfile;
    tableFile: string;
    line: string;
    recCount: integer;
    concept: string;
    section, topic: integer;
    elaboration, keyIdea: string;
    entered: double; //TDateTime
begin
    tableFile : =HELP_UPDATE_DIR +'What.txt';
    recCount: =0;
    try
        assignFile( infile, tableFile );
        reset( infile );
        while not eof( infile ) do begin
            readln( infile, concept );
            readln( infile, section );
            readln( infile, topic );
            elaboration: =";
            readln( infile, line );
            while( line <> 'True' ) and ( line <> 'False' ) do begin //keyIdea
                elaboration: = elaboration+line+ #10;
                readln( infile, line );
            end;
            keyIdea: =line;
            readln( infile, entered );
            dmMsgData.tbWhat.AppendRecord( [concept, section, topic, elaboration,
                keyIdea, entered] );
            inc( recCount );
            readln( infile ); // chuck away '[END]' of record
        end;
        closeFile( infile );
    end;
end;

```

```

//housekeeping
deleteFile( tableFile );
TellUser( self.name,intToStr( recCount ) +
' records transferred to What
table', true, false );
    except
        TellUser( self.name,'Read error: Unable to update '+
'WHAT table; record count = '+ intToStr( recCount ), true,
controller.debug );
        closefile( infile );
    end;
end;

procedure TUpdateExtramuralSupport.deleteFile( filepath: string );
begin
    with FileManager do begin
        source : =filepath;
        action: = vDELETE;
        execute;
    end;
end;

procedure TUpdateExtramuralSupport.activateTables;
begin
    with dmMsgData do begin
        tbConnection.Active: =true;
        tbWhere.Active: =true;
        tbWhat.Active: =true;
        tbWho.Active: =true;
    end;
end;

end. viaUpdateExtramuralSupport

```

H6: Learning Shell: Messaging – interface

```

unit vlaMessaging;
{   This unit provides the message system functionality for the Learning Shell. Its
{   implementation closely parallels that for the Repository end
}

interface

uses
    Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
    Db, Grids, DBGrids, DBTables, StdCtrls, DBCtrls, ExtCtrls, Mask, ComCtrls,
    Lmdcompo, Lmdclass, LmdnonvS, Psock, NMFtp, vlaFileManager, vlaForwardMessage,
    vlaMessageList, Buttons, vlaHelpBtn;

const
    DEBUG = true; //false; //

type
    TMessageSystem = class( TForm )
        dsMsg: TDataSource;
        ...
        dsUpdateMsg: TDataSource;
        FTP_Server: TNMFTP;
        FileManager: TvlaFileManager;
        ...
        tmKeepTrying: TTimer;
        tmGiveUp: TTimer;
        vlaHelpBtn1: TvlaHelpBtn;
        procedure btNewClick( Sender: TObject );
        procedure btUpdateClick( Sender: TObject );
        procedure btCloseClick( Sender: TObject );
        procedure btReplyClick( Sender: TObject );
        procedure FTP_ServerConnectionFailed( Sender: TObject );
        procedure FormShow( Sender: TObject );
        procedure StatusBarClick( Sender: TObject );
        procedure btDeleteClick( Sender: TObject );
        procedure btForwardClick( Sender: TObject );
        procedure btPrintClick( Sender: TObject );
        procedure FormCreate( Sender: TObject );
        procedure FormPaint( Sender: TObject );
        procedure tmKeepTryingTimer( Sender: TObject );
        procedure FileManagerFailure( Sender: TObject );
        procedure FileManagerSuccess( Sender: TObject );
        procedure tmGiveUpTimer( Sender: TObject );
        procedure vlaHelpBtn1Click( Sender: TObject );
        procedure FormKeyDown( Sender: TObject; var Key: Word;
                               Shift: TShiftState );

    private
        sendMsgFile: string ;
        lastRcvdByRepository: TDateTime;
        Save_Cursor: TCursor;
        MessageList: TMessageList;
        procedure saveToSendList;
        procedure updateMessages;
        procedure sendNewMsgs;
        procedure fetchTransmission( transName: string );
        procedure openNewMessage;
        function getMsgName: string;
        procedure createToSendList ( lastUpdate: TDateTime );
        procedure requestLastRcvd;
        function processTransHeader( var tmpfile: textfile; const tnsfile: textfile;
                                     var transType: integer ): boolean;
        procedure addTransHeader( var tnsfile: textfile; transType: integer );

```

```
    procedure SendAndWait;
    procedure sendTransmission( tns: string );
    procedure deleteFile( fullpath: string );
    procedure WaitForReply( transType: integer );
    procedure synchroniseWithRepository;
    procedure clearMessageBox;
    function messageBoxContains( something: string ): boolean;
    function processAnswer: boolean;
    procedure connectToFTPServer;
    procedure FetchAndFinish;

public
    procedure setMsgList( MsgList: TMessageList );
    procedure replyToMsg;
    procedure tellUser( pos: integer; msg: string );
    procedure clearUserMsg;
    procedure forwardMsg( FMsg: TForwardMessage );
    procedure printMsg;
    procedure addToLog( msg: string );
    procedure connectToRepository;
    procedure enableUpdates;

end;

var
    MessageSystem: TMessageSystem;

implementation

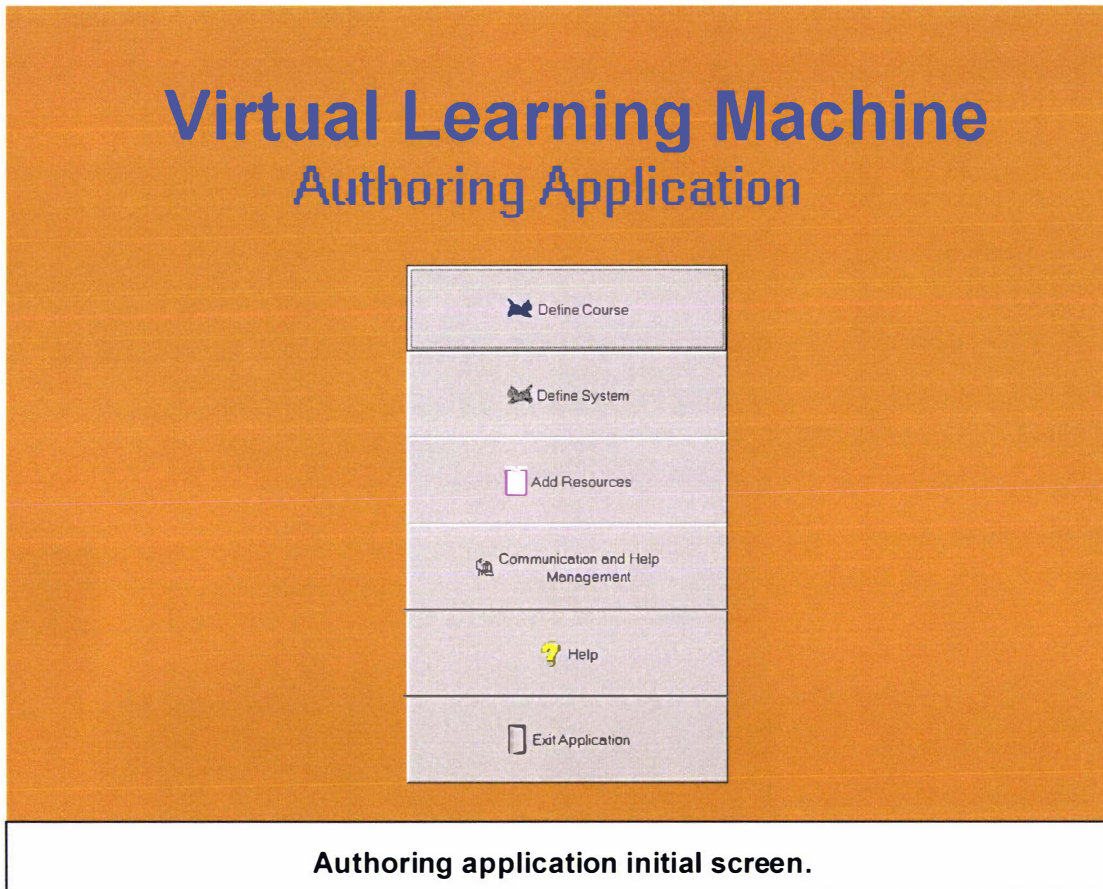
uses
    v1aMsgData, v1aAddMessage, v1aSystemDictionary, v1aReplyMessage,
    v1aSystemUtilities;

{SR *DFM}
```


Appendix I

Course authoring and management application

I1: Authoring application – screen shots



Virtual Learning Machine

Message List

From	To	Cc	Date	Subject
System	Tutor		04-May-03 15:02:14	Section 1.2 Explain "query" System response \
Russell	Tutor		24-Mar-03 15:55:47	Section 1.2 Explain "query" System response \
Chris	Mim		07-Oct-03 21:27:48	Welcome to 159.353 Extramural
Chris	Tom		07-Oct-03 21:28:49	Welcome to 159.353 Extramural
Chris	Bob		07-Oct-03 21:29:47	Welcome to 353 Extramural
Chris	Mike		07-Oct-03 21:30:31	Welcome to 353 Extramural
Chris	Bob	All	09-Oct-03 16:01:35	Assignment 1
Chris	Bob		12-Oct-03 15:43:06	Are you ready to rumble?

Get going!

Debug

Tutor's message list

From: System

To: Tutor

Cc:

Date: 04-May-03 15:02:14

Subject: Section 1.2 Explain "query" System response was rejected

Section: Topic:

NO USEFUL ANSWER FOUND IN SECTION 1.2
SELECT "RETRY" FOR A BROWDER SEARCH

Extramural Support system notification. Database may be edited from Message List.

Edit Help Database

Concept: Elaboration:

Section: Topic: Key Idea:

Concept	Section	Topic	KeyIdea	Elaboration	LastU
▶ Conceptual frameworks	1	2	False	(MEMO)	18-De
Conceptual frameworks	1	2	True	(MEMO)	25-Au
Conceptual frameworks	1	3	False	(MEMO)	18-De
Conceptual models	1	2	True	(MEMO)	19-Au
Design process	1	1	True	(MEMO)	25-Jan
Design team	3	2	False	(MEMO)	22-Au
Interaction paradigms	1	2	True	(MEMO)	19-Au
Interaction styles	1	2	True	(MEMO)	21-Au
Norman's model	1	4	True	(MEMO)	21-Au

Buttons: Add, Edit, Themes, Links, References, Finish, Cancel

Adding to Extramural Support database.

Add References

Add references for "Towards physical design"

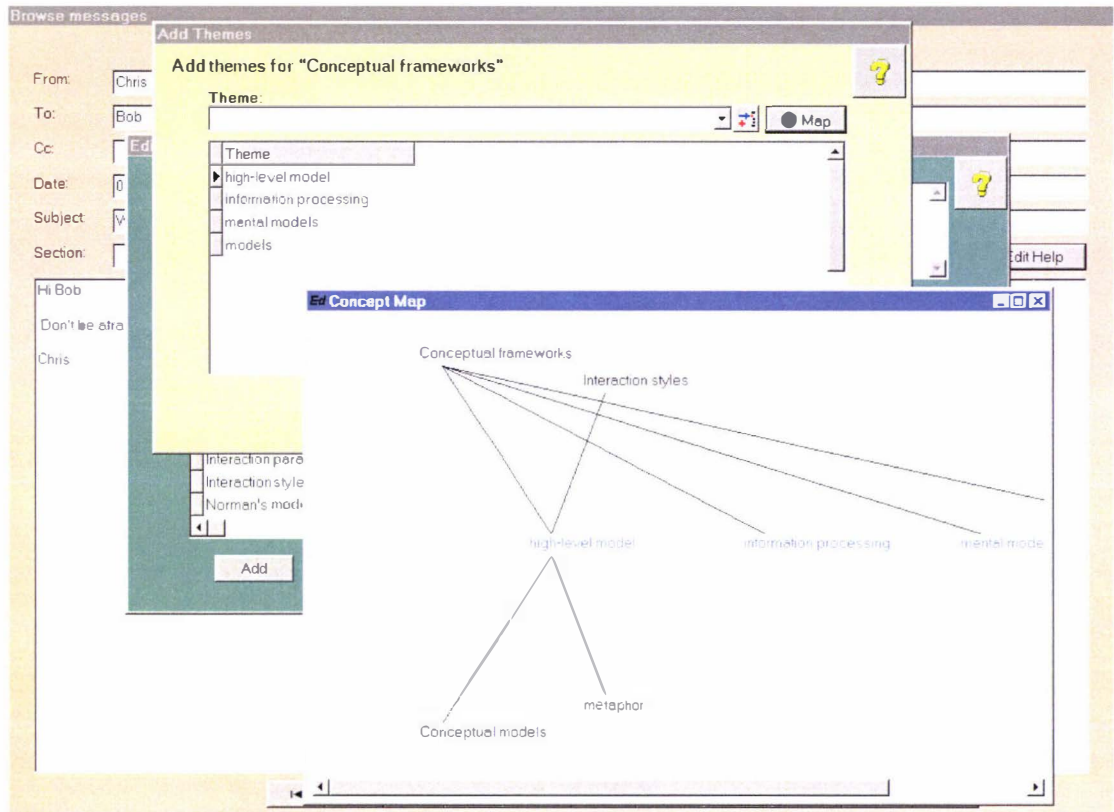
Reference:

Chapter, page numbers:

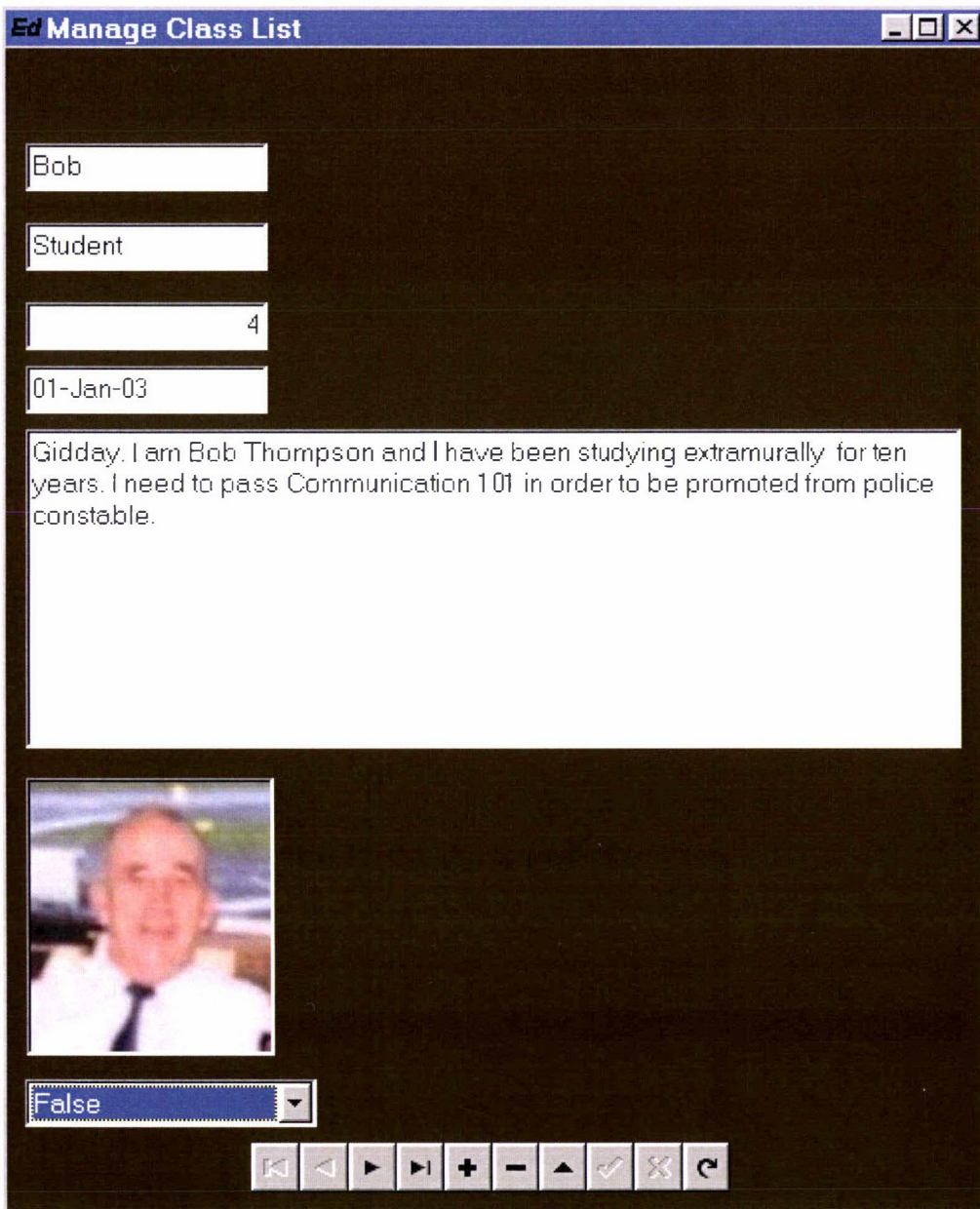
Reference
▶

Buttons: Add, Edit, Finish

Adding to Extramural Support database.



Tutor can view Concept Map after editing Extramural Support database.



The screenshot shows a software window titled "Ed Manage Class List". The window has a dark background and contains several input fields and a text area. The input fields contain the following text: "Bob", "Student", "4", and "01-Jan-03". Below these fields is a large text area containing the message: "Giddyay. I am Bob Thompson and I have been studying extramurally for ten years. I need to pass Communication 101 in order to be promoted from police constable." Below the text area is a small portrait photograph of a man in a white shirt and tie. At the bottom left of the window is a dropdown menu showing "False". At the bottom center is a row of navigation icons: a left arrow, a right arrow, a plus sign, a minus sign, an up arrow, a checkmark, an 'X', and a refresh symbol.

Ed Manage Class List

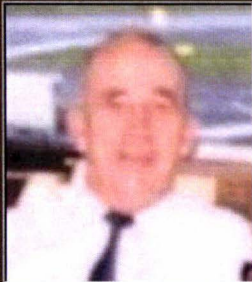
Bob

Student

4

01-Jan-03

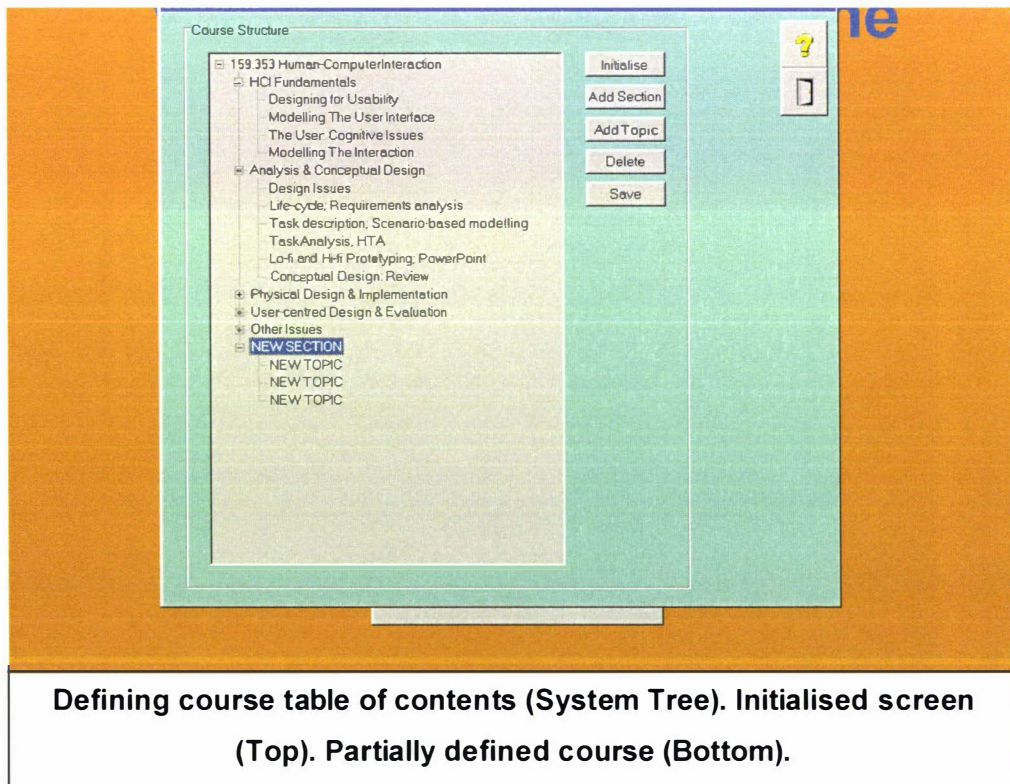
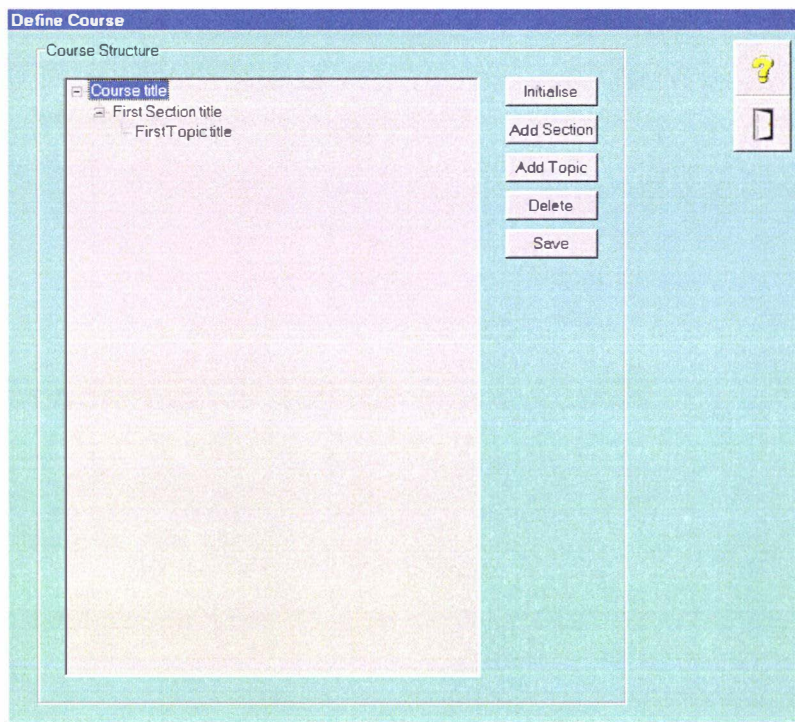
Giddyay. I am Bob Thompson and I have been studying extramurally for ten years. I need to pass Communication 101 in order to be promoted from police constable.



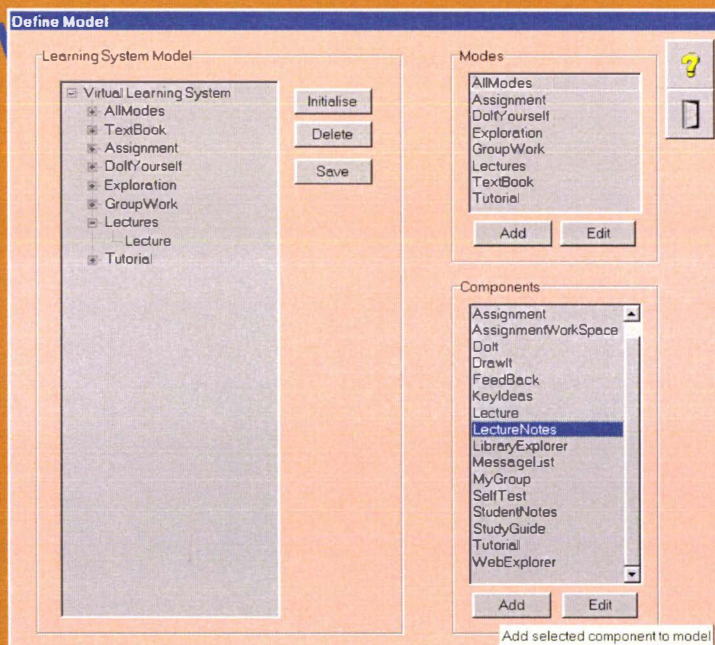
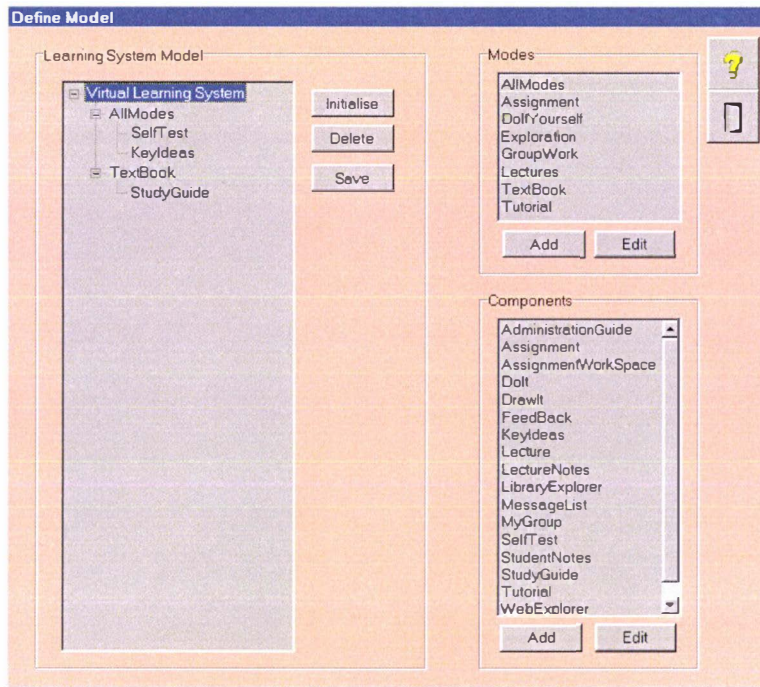
False

Navigation icons: left arrow, right arrow, +, -, up arrow, checkmark, X, refresh

Class List can also be managed from Message List



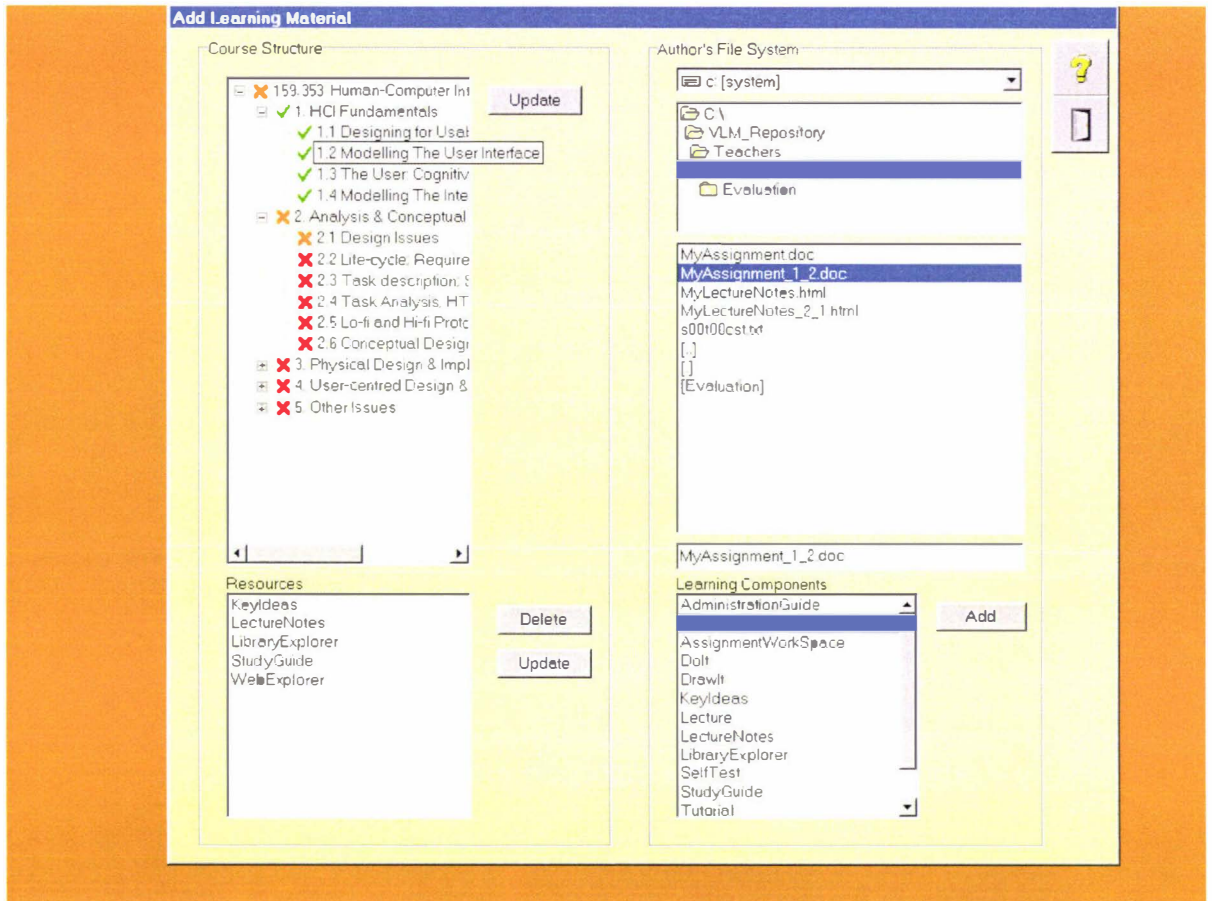
Defining course table of contents (System Tree). Initialised screen (Top). Partially defined course (Bottom).



Defining course study modes (System Model). Initialised screen (Top).

Partially defined model (Bottom).

I2: Add Learning Material – screen shot and class interface



```
unit docAddLearningMaterial;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ComCtrls,
Buttons, Imdcompo, Imdclass, ImdnonvS, ExtCtrls, FileCtrl, vlaSystemTreeManager,
vlaResourceManager, vlaFileManager, vlaComponentList, ImgList, vlaFolderManager;
```

```
type
```

```
TAddLearningMaterial = class(TForm)
...
  vlaFolderManager1: TvlaFolderManager;
  procedure FormShow(Sender: TObject);
  procedure btHelpClick(Sender: TObject);
  procedure btExitClick(Sender: TObject);
  procedure TreeViewClick(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure lbxNodeViewClick(Sender: TObject);
  procedure btAddResourceClick(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure FileManagerSuccess(Sender: TObject);
  procedure btDeleteClick(Sender: TObject);
  procedure FileManagerFailure(Sender: TObject);
  procedure btUpdateClick(Sender: TObject);
  procedure btKeyIdeasClick(Sender: TObject);
  procedure btSelfTestClick(Sender: TObject);
```

```

procedure btUpdateStatusClick(Sender: TObject);
procedure vlaFolderManager1Failure(Sender: TObject);
procedure vlaFolderManager1Success(Sender: TObject);
private
    courseTree: TSystemTreeManager;
    resMan:TResourceManager;
    section,topic: integer;
    virtualFile: string;
    selectedTopic: string;
    studentOnly: TCompList;
    noSpecialDirComp: TCompList;
    specialWriteComp: TCompList;
    basicComp: TCompList;
    addImage: boolean;
    imageFile: string;
    statusDefaultsProcessing: boolean;
    resourceUpdated: boolean;
    function folderSelected: integer;
    procedure addFolderResource( component: string );
    procedure addStartUpFile( virFile, startFile: string );
    procedure createCourseTree;
    function isSection( tnode: TTreeNode ) :boolean;
    function isTopic( tnode: TTreeNode ) : boolean;
    function contains (tparent,tnode:TTreeNode):boolean; overload;
    procedure loadComponents;
    procedure parseTopic (title:string);
    procedure addResource(component: string);
    function getSuffix(filename: string): string;
    function GetComponent(lbx:TListBox): string;
    procedure deleteResource;
    procedure updateNodeView;
    procedure loadStudentOnly;
    procedure loadNoSpecialDirComp;
    procedure loadSpecialWriteComp;
    procedure loadBasicComp;
    procedure updateTopicStatus ( sect,top: integer );
    procedure updateCourseTree;
    procedure addSpecialDirRes( virFile, auFile, component: string );
    procedure addSpecialWriteRes ( virFile, component: string );
    function getStartFile( component: string ):string;
public
    procedure setImageFile(imgFile: string);
    function getVirtualFile:string;
    function getSelectedTopic: string;
    procedure getSectionTopic(var sect, top:integer);
    {copy all updates of course material to the Respository archive}
    procedure updateCourseArchive;
    procedure initialiseForm;
end;

var    AddLearningMaterial: TAddLearningMaterial;

implementation

uses    vlaSystemDictionary, auCourseModelEditor, auSelectImageDlg, auAddSelfTest,
        vlaSystemUtilities, auIndexFileDlg;

end.

```

Appendix J

Evaluation – handy hints, scenarios for user testing

J1: Handy Hints sheet

Confused?

If you become confused while using Massey Extramural, keep these options uppermost in your mind:

1. Get help by pressing **F1** or by clicking on the **Tutor** (Help) icon on a component, and read the tips on using that component.
2. Open the **Desktop Menu** by right-clicking with the mouse on the **desktop** (background screen). This menu provides access to many of the additional tools you need to complete a task.
3. Open the **Course Explorer** from the **Desktop Menu**. You can shift to any topic or any study mode in the course by selecting it in the **Explorer**.

Use of the mouse

1. **Click** means hold the mouse pointer over the screen object and press the **left-hand** button on the mouse.
2. **Right-click** means hold the mouse pointer over the screen object and press the **right-hand** button on the mouse.
3. **Double-click** means hold the mouse pointer over the screen object and press the **left-hand** button on the mouse twice in quick succession.
4. **Select** text in a document by placing the mouse pointer at the beginning of the text to be selected, hold down the **left-hand** button on the mouse and drag the pointer to the end of the text to be selected. The **selected text** will be highlighted on the screen.
5. **Select** an item in a list by clicking anywhere on that item. The **selected item** will be highlighted on the screen.

Use of keyboard shortcuts

The following key board shortcuts can be used wherever appropriate throughout the system.

1. **Help (F1)**. Click on a component to select it (the bar across the top turns blue). Locate the **F1** key (sometimes labelled "Help") in the top left-hand corner of the keyboard, and press it down firmly to access the help page on the selected component.
2. **Copy (Ctrl+C)**. Select the text to be copied with the mouse. Locate the **CTRL** key in the bottom left-hand corner of the keyboard. While holding the **CTRL** key down, press the **C** key.
3. **Paste (Ctrl+V)**. Click the position for the text to be pasted with the mouse. Locate the **CTRL** key in the bottom left-hand corner of the keyboard. While holding the **CTRL** key down, press the **V** key.
4. **Cut (Ctrl+X)**. Select the text to be cut with the mouse. Locate the **CTRL** key in the bottom left-hand corner of the keyboard. While holding the **CTRL** key down, press the **X** key.

J2: Initialisation (Scenario 1)

{In this scenario you set up the course software for your personal use, and familiarise yourself with three of the most important features of the system – the Course Explorer, the Desktop Menu, and the Help system.}

1. Click the Massey Extramural icon on the **Desktop**. The **Windows Desktop** is replaced by the **Massey Extramural Desktop**. You are presented with a **Logon** screen which welcomes you to the 159.353 Human-Computer Interaction course, and prompts you to enter a username and password.
2. Enter your assigned username and password (Use your username without any capital letters as the password).
3. Confirm your password by re-entering it.
4. Click the OK button (or hit ENTER). You are now asked to confirm your username.
5. Click OK. You are now presented with a screen which requests additional information the system needs to run **Massey Extramural** on your machine.
6. Click the OK button or hit ENTER to accept the default computer settings.
7. Click OK again to confirm your settings. You are now presented with the **Desktop Help** screen providing an overview of **Massey Extramural**.

Open the Course Explorer

8. After reading the overview close it by clicking on . You are now presented with the **Course Explorer**. The **Course Explorer** is set to the start of the course.
9. Click on the **Help** (Tutor) icon. A page opens outlining the features of the **Course Explorer**. After reading this, click on to exit **Help**.
10. Now try pressing the F1 key on the keyboard. This will also access Help. In general, you can open a **Help** screen for any component by using the mouse to click the Tutor icon in the right-hand top corner of a component or by pressing the F1 key on the keyboard
11. Exit **Help** as before by clicking on . The system closes most components for you when you move around in the course. Any component that you can close yourself will have in the right-hand top corner and possibly also a Close button
12. Use the mouse to explore the **Course Tree**. The left side of the **Course Explorer** displays the contents of the course. If you click on a [+] sign to the left of a section it will expand to show its topics. If you click on a [-] sign next to a section, its topics will be hidden. The coloured squares next to each item in the contents indicate the status of that part of the course: RED means you have not yet attempted it, AMBER that you have attempted but not completed it, and GREEN that you have completed all the material within it.
13. Select Topic 1.3. Note how the top right-hand box changes to display information about the currently selected topic. The bottom right-hand box changes to display the available study modes for the selected topic.

14. Select Topic 1.1. Select TextBook Mode.
15. Click on the OK button. The Explorer component closes and the components of the Text Book Mode open at Topic1.1. Note that the **Desktop** also displays the course title; the current section, topic and topic title; and the current study mode.

Access the Administration Guide

16. Right click (i.e. use the right-hand button on the mouse) on the **Desktop** and a **Menu** of options pops-up. This Menu provides a means of accessing general learning aids and other features available in any study mode.
17. Select Learning Aids with the mouse.
18. Click on AdmistrationGuide. A screen opens displaying the **Administration Guide** which provides details of the requirements for completing the course.
19. Find out who the teaching staff for this paper are.
20. Click on the in the top right-hand corner to close the Guide.
21. Open the **Desktop Menu** again and click on Exit. A box appears asking you to confirm that you wish to exit the course.
22. Click on Yes. A box appears asking if you wish to backup your work.
23. Click on No to exit **Massey Extramural**.

<End of initialisation scenario>

J3: Start-up, browse the Study guide (Scenario 2)

(Prerequisite: Ensure the computer is connected to the university)

{In this scenario you access the Study Guide, change study modes, and familiarise yourself with two of the tools for supporting your learning in any mode.}

Logon

1. Launch **Massey Extramural**
2. Enter your username and try "1234" as password.
3. Enter your correct password and continue. A **User Options** form appears offering a number of choices.

Browse the Study Guide

4. Use the **Course Explorer** to navigate to Section 1.2 of the **Study Guide (Textbook Mode)**. (Click the **Help** icon if you need guidance).
5. Open your **Student Notes**.
6. Browse the Study Guide for a few minutes. Find the paragraph on "Interaction Paradigms".
7. Copy the paragraph on "Interaction Paradigms" into your notes.
8. Close your Notes. When prompted, save your changes.
9. Each item in **Key Ideas** represents a key concept or learning goal for this topic in the course. The face to its left indicates the status of this concept: RED, means that you have not yet demonstrated an understanding of this concept, AMBER, means you have indicated partial understanding, and GREEN, a good understanding.
10. Open **Course Explorer** and double-click Lectures in Study Options. You are now in Lecture Mode. The same **Key Ideas** and **Student Notes** can also be accessed from here.
11. Open **Key Ideas** and **Student Notes** using the **Desktop Menu**.
12. Exit **Massey Extramural**. Do not backup your work.

<End of start-up and browse scenario>

J4: Work on an Assignment (Scenario 3)

(Prerequisite: Ensure the computer is connected to university)

{In this scenario, you familiarise yourself with the aspects of the system for communicating and collaborating with others, and practise copying and pasting data between components.}

Logon

1. Launch **Massey Extramural**.
2. Enter your username and password.
3. Return to where you left off in the course previously. (Click the **Help** icon if you need guidance).

Access and print your assignment

4. Use the **Course Explorer** to change to the Practice mode.
5. This is only a practice assignment. Use the **Course Explorer** to search the topics in Section 1 for the Assignment study option. Open it.
6. Find the section of the **Assignment** requiring you to produce a PowerPoint simulation. Copy this section to your **Student Notes**.
7. In the **Assignment Work Space** enter your username on the cover sheet.
8. You will submit your assignment by mail. Print the Assignment cover sheet.

Discuss the assignment in your group

9. Change to Group Work mode. (Each student is assigned to a workgroup with whom you collaborate on assignments and other tasks.)
10. Press the F1 key . Read the **Help** page.
11. Find out where your group leader works.
12. Update your messages. This will take a few minutes. Be sure to wait until the process is completed before continuing.
13. View All your new messages.
14. Delete the Welcome message from Chris.
15. Reply to your group leader's query. C.C. a copy to members of your group. Send this message Later. Close **All New Messages**.
16. Open **Student Notes**. Use the mouse to select (highlight) the section on the Power Point simulation. Use CTRL+C to copy this section (Hold down the CTRL key and then press the C key).
17. Use CTRL+V to paste this into a new message to Chris. Ask Chris what he means by this. C.C. a copy to all members of the class. Send this message now.
18. Exit **Massey Extramural**. Do not backup your work.

<End of Assignment scenario>

J5: Access a Lecture, Ask for Learning Support (Scenario 4)

(Prerequisite: Ensure the computer is connected to university)

{In this scenario, you access a video presentation, seek support in understanding one of the concepts raised in the lecture, and then evaluate your understanding of these concepts.}

1. Log on to **Massey Extramural**
2. Use the **Course Explorer** to navigate to the lecture at Topic 1.2. (Click the **Help** icon or press F1 if you need guidance).
3. Load and start the lecture (For this trial it is only a sample clip. In a real situation you would follow the presentation with the help of the slides contained in **Lecture Notes**.)
4. Use **Desktop Menu** to open **Key Ideas**. This is the gateway to **Extramural Support**.

Getting help with Key Ideas

5. You are confused about "metaphor". Select this concept and open **Extramural Support**.
6. Press F1 and read the **Help** page.
7. Ask **Extramural Support** to explain "metaphor".
8. RETRY for further information.
9. Response is inadequate. Click NO.
10. ASK the tutor for more information on "metaphor".
11. Open the **Concept Map**
12. Select the related item "conceptual models" there.
13. Ask **Extramural Support** to explain "conceptual models".
14. RETRY.
15. Click NO.
16. Exit **Help**.

Self Assessment

17. You decide to register how well you understand Topic 1.2. Note the colour of the icons next to each concept in **Key Ideas**..
18. Find and start the **Self-Assessment** questionnaire. Give a range of responses over the questions. Rate "metaphor" and "conceptual models" as poor.
19. Update **Key Ideas**. Note the changes to the colours of the faces to the left of the concepts.
20. Open the **Course Explorer**. Note the changed colour at Topic 1.2.
21. Change study mode to Group Work. Update your messages. Wait until the process is complete before continuing.
22. Exit **Massey Extramural**. Do not backup your work.

<End of Lecture and Ask for Help scenario>

J6: Monitor group discussion, get assignment feedback (Scenario 5)

(Prerequisite: Ensure the computer is connected to university)

{In this scenario you use the internet to update your course material and messages from the university.}

1. Logon to Massey Extramural.

Update course material

2. From **User Options**, select the option for updating your course material and click OK.
3. Click the **Help** icon and read carefully the instructions for updating your course material from the Internet.
4. When you are clear how to proceed, close **Help** and click OK. A screen appears from where you will “unzip” (unload) material received from the university.
5. Click UNZIP. Wait for the unzipping process to complete.
6. Click CLOSE. Wait for the updating process to complete and return you to **User Options**.

Update your messages

7. Navigate to Topic 1.3 in Group Work mode
8. Update your messages. Wait for the process to complete before continuing.
9. Read all the new messages you have received.
10. Close **All New Messages**.
11. Find and open Assignment mode again .
12. Check your grade.
13. Exit **Massey Extramural**. Do not backup your work.

<End of monitor scenario>

J7: Exploring a Topic (Scenario 6)

(Prerequisite: Close the connection to university, so that you can access the Internet automatically)

{In this scenario you will explore learning resources on the Internet.}

Logon

1. Log onto **Massey Extramural**.
2. Return to where you left off in the course last time.

Exploring the Internet

3. Change to Exploration mode. You are presented with two components:
 - **Explore the Web**; and
 - **Explore the Library**.
4. Use the **Desktop Menu** to move to Topic 1.1
5. Click **Explore the Web**. A screen opens containing a web page viewer, a list of web page references and a workspace for you to enter notes.
6. Choose the UsabilityFirst item in the list and click GO. Wait for the web page to load.
7. You find the page unhelpful. Click DELETE and the link is deleted from the web references list.
8. Close the component. Save your changes.
9. Use **Key Ideas** to search for help on "Usability adoption by industry".
10. Ask "Where do I find more on usability adoption by industry?" RETRY until no additional information is available.
11. Click COPY and then EXIT.
12. Click on the Explore the Web option again. The web references list will now include the additional references you located through **Extramural Support**.
13. Go to the Bad Designs item you located through **Extramural Support**. Copy a paragraph to your notes, and then close the component.
14. Open your **Student Notes**. Your new notes have been added.
15. Try the Explore the Library option. This links you to the Massey University Library web site where you may search for and borrow many books relevant to the course.
16. Close the component.

<End of Exploring scenario>

J8: Completing an interactive tutorial (Scenario 7)

{In this scenario you try the option of learning through an interactive tutorial.}

1. Yesterday you had difficulties with Topic 1.2, especially "metaphors" and asked for help from the course tutor. Today you decide to look at the topic from a new angle. Use **Course Explorer** to navigate to Topic 1.2 and select Tutorial.
2. Open **Key Ideas**. Select "metaphors" again and re-explore the available help. Note the update of Extramural Support. The course tutor has responded to your request for extra help on this concept.
3. Click OK and then EXIT.
4. Launch the tutorial. Explore its features.
5. Close the tutorial.
6. Update the **Self-Assessment**. Rate yourself with a good understanding of all concepts. Update **Key Ideas** and note the changes.
7. Use the **Desktop Menu** to move back to Topic 1.1. No tutorial is available here.
8. Complete the self-assessment for Topic 1.1. Check the status updates in **Key Ideas** and **Course Explorer**.
9. Exit **Massey Extramural**. Do not backup your work.

<End of Tutorial scenario>

Appendix K

Evaluation: Information sheet, questionnaire, and interviews

K1: Information sheet for participating volunteers

RESEARCH PROJECT ON COMPUTER-BASED

DISTANCE LEARNING

INFORMATION SHEET

What is the study about?

This experiment is investigating a potential means for delivering university-level distance education as an alternative both to traditional paper-based correspondence courses and to computer-based systems which require fast and reliable Internet service to work. We are testing the viability of what we call the Virtual Learning Machine as a method for delivering computer-based courses that works as well in remote districts as it does in the cities. This is why we are carrying out the experiment at Akitio.

Who is conducting the study?

The study is being conducted by Russell Johnson as a part of his Ph.D. research at Massey University. The research supervisor is Associate Professor Elizabeth Kemp from the Institute of Information Sciences and Technology, supported by Associate Professor Ray Kemp from the Institute of Information Sciences and Technology and Mr. Peter Blakey from the College of Business, at the Turitea Campus of Massey University in Palmerston North. (See contact details at the end of this document.)

What will participants do?

The experiment will be held at Akitio School. You will complete a short questionnaire summarising your previous computing experience. You will then be asked to complete several tasks using the VLM program over two sessions of up to an hour each. Following that I will conduct a short interview with you to gather your impressions of the experience.

By agreeing to take part in this study, you will be giving permission for me to analyse your data for these tasks as part of this PhD experiment.

How much time is involved?

I expect that participants will spend 2-4 hours on the experiment. This will involve two sessions spread out over a week.

What will happen to the information?

The results will be anonymous, and will be used only for the purpose of this study. The results will be published in the researcher's PhD thesis, and in addition may be published in professional journals or conference proceedings. A brief report of the initial results will be sent to every participant who is interested. When the research is completed, the raw data will be held securely for a suitable time period, and then destroyed.

Summary of your rights

While participating in this research project you have the right to:

- **refuse to answer** any particular question
- **withdraw** from the experiment at any time
- **ask questions** about the experiment at any time
- **provide information** on the understanding that your name will not be used
- **receive a summary** of the findings from the experiment when it is completed

If you have any questions or concerns about the experiment, or would like further information about the experiment, please contact any of the people whose names appear below.

Contact details:

Researcher: Russell Johnson

Institute of Information Sciences and Technology

Tel: 374 3898

Email: R.S.Johnson@massey.ac.nz

Principal Supervisor:

Dr E. A .Kemp

Institute of Information Sciences and Technology

Tel: 350 5799 x 2469

Email: e.kemp@massey.ac.nz

K2: Questionnaire for participants

Participant Profile

1. Username:.....
2. Gender (circle one): Female / Male
3. Are you currently studying at university or polytechnic level? (circle one) Y / N
4. If not, have you studied at university or polytechnic level? (circle one) Y / N
5. Major subject
6. Have you ever studied by correspondence (extramurally)? (circle one) Y / N
7. Computer usage.

In what year did you first use MS Windows?

In a typical week, how many hours, including work and leisure use, would you spend at a computer?

Now please detail how many hours per week on average over the last year you have used computers for the following tasks, by circling the appropriate number.

	(Hours per week)					
Write and send emails	0	1	2	3	4	5 or more
Browse the internet for news/information	0	1	2	3	4	5 or more
Write letters or reports with a word processing program (e.g. Word, WordPerfect, MS Works, etc.)	0	1	2	3	4	5 or more
Keep accounts and budgets with a spreadsheet program (e.g. Excel, Lotus 123, etc.)	0	1	2	3	4	5 or more
For your own university or polytechnic studies	0	1	2	3	4	5 or more
To help others(e.g. your children) with their education	0	1	2	3	4	5 or more
Other (please indicate)						
.....	0	1	2	3	4	5 or more
.....	0	1	2	3	4	5 or more

END OF QUESTIONNAIRE

K3: Outline for semi-structured interviews with each participant

(Each participant interviewed at the end of each session)

Day 1

1. Go through each scenario (Initialise, Start-up, Assignment, & Lecture) using the written guide. Were you able to complete? Did you have any difficulties? Where? Major/minor?
2. Today you carried out a number of computer tasks, including
 - Set-up a complex computer program on your machine
 - Searched the computer directory system to find the correct programs to run
 - Opened files for those programs
 - Browsed web pages
 - Created a document, edited and saved it in a word processor
 - Printed a document
 - Received, composed and sent messages across the internet
 - Manipulated multimedia files
 - Updated a database
 - "Talked" with interactive software

Were you aware you were doing these things? Have you done any of these things on your own computer before? Did you find it harder/easier with this special-purpose system?

3. What aspects of Massey Extramural did you like compared to doing things with your own computer?
 4. What aspects of Massey Extramural did you not like compared to doing things with your own computer?
 5. What things would you like to see added?
 6. What things would you like to see removed?
-

Day 2

1. Go through each scenario (Monitor, Explore, Tutorial) using the written guide. Were you able to complete? Did you have any difficulties? Where? Major/minor?
2. Today, in addition to those you carried out yesterday, you have completed a number of new computer tasks, including
 - Transferred data files from a remote computer to your computer
 - Synchronised a database on your machine with a database on a remote computer
 - Searched a database for information on a particular subject

- Searched for and located a web-site on a particular subject
- Worked with an interactive software programme on a remote computer

Were you aware you were doing these things? Have you done any of these things on your own computer before? Did you find it harder/easier with this special-purpose system?

3. What aspects of Massey Extramural did you like compared to doing things with your own computer?
 4. What aspects of Massey Extramural did you not like compared to doing things with your own computer?
 5. What things would you like to see added?
 6. What things would you like to see removed?
 7. Do you feel able to use the system by yourself now?
-

K4: Interviews with participants

First Interview with User 1, 13/10/03

Q: What I want to do is ask you just to go through your experience working through the scenarios. Were you able to complete initialisation one?

A: Yes

Did you have any difficulties?

No.

Q: That was the one we went through together. The second one that we went through together was Start-up and Browse. Did you have any difficulties with this?

A: Yes

Q: Was there anything that you got caught up on?

A: No. I did make a mistake, but I was aware that I had made a mistake. It was just about how to go back and fix my mistake. It was step 5, Open your student notes. I had skipped that and gone into the notes and when it became time to put it into your student notes I realised that I hadn't opened it and that there was nowhere to put it. It wasn't a problem It was just a matter of sitting and thinking where I had to go back to and starting from there

Q: The next scenario is where you were working it through on your own. I understand we had a couple of technical problems with the computer.. You should try to put that aside. But if it was a problem anywhere you should mention that. Were you able to complete the assignment scenario?

A: Yes

Q: Did you have any difficulties?

A: No. And I thought there, especially when it asks you to click on Help at the start, to read about how to go about it. Say if you go down to discuss the assignment and you go click F1 and read the Help page. Well, there you didn't need to commit that to memory either, because I knew then that if I had problems I could still go back to that, which I did. So I had a quick flick, then felt confident enough to go on. And then when I did need to check some things, then F1 and it was very easy to follow and go about it, especially like when we went into the Group work mode and then I had to find out about the lecturer. I didn't really look at how to do that. I just knew that when I go to something I didn't know then F1 and it was there. And I found it very easy. Quick, too. The fact that I didn't have to read all that and commit it to memory. Just, Hey! F1 will help me when I have got a query. It was very straightforward.

...Down at Step 15. Close All New Messages. That was where it did take me a couple of "figuring it outs" because in the other things, there is no Close in them, and you know I figured I had to go to [All] New Messages to get that close button because the others didn't have close on them so I just whipped along the bottom there to have a look to see where my close option was because there is no X, but there on New Messages I saw Close,

But you would have preferred to have seen the X on that one as well?

Well. If it's not going to alter anything, I think it would be easier. Because otherwise you are opening up new fields that maybe you don't have to open up. Like that New Messages. It says View All but then you have to open up All New Messages

So you actually did the delete and reply from with All New Messages and then you closed that, you had to close that to go back to the part of the course you were in...

...because then I had to go back to Student Notes and then add that message in. It took me a minute or two to figure out how to close it. I went along each icon [button] to see what they had in them. As soon as I saw Close I knew I was on the right track then...

And, any other difficulties...

That was where there was a technical difficulty [in which an image from a previous screen stayed on screen], so I could not see what I was typing... But then, after I completed that task and did my next click, I could see my message. So it was working behind it, I just couldn't see it. So I had to rely on the fact that it was going in.

Yes that was unfortunate it was a question of the [memory] capacities of the computer. So then you accessed the lecture and the learning support system?

Yes.

How did that go?

Good. Again. F1, Help page. I think by that stage I had figured out how to use this so I didn't really need the F1. There were only three choices there anyway so it was pretty much straightforward. You've got Retry, you've got your Ask underlined here [in the scenario]... what you needed to use. So it was quite straightforward. I think that [scenario] 4 was easier than 3. Then, I know I wasn't supposed to, I did 5, which was also easier than the 3. I think it got easier. Whether that was because I had figured out how to do it I am not sure, but I found 4 and 5 easier than 3 to be quite honest..

I think that is very likely because there is a learning curve with this like with anything. But there are only a few basic moves with this system, and once you have got those it will be interesting to see when you do the next part tomorrow, whether you fly through or what have you. Another way of looking at what we're doing today...

You know with your ticks down there at the bottom. This is on access the lecturer, about the metaphor. Do you remember the Retry's, the tick for the OK's, so there it could be F1 that I had to look at. It took me a couple of seconds to figure out that I had to tick everything. So you know how when you have got, you have to click No or something, there was one there that in particular that it wasn't showing up as an option until I clicked on that tick box, then they all come up again. But that was again straightforward. You know why isn't this showing up It's a process of elimination basically. If I tick that... But I was a little worried that if I tick that box I was going to exit out of it [Extramural Support], but no I figured after I had ticked it once that No. obviously you stay in there. And all those options come up..

What you're doing with all those buttons is that you are having a dialogue with the system - you give them a question and they give you an answer and you give them a response and they give you another response and that interchange is going on..

And again it is one of those things that once you have used it once, you know what that tick box is for, as you say, it is like a sentence, it's like a full stop to me. I have finished doing that and then....

Fine. We'll come back to some of those things ... This extramural learning prototype is a simplified system computer-wise, but it is not the same thing as a simple system because the tasks are quite complex. In fact I will just go through a number of the things which you actually did computer-wise today. You set up a complex computer program on your machine, you searched the computer directory system to find the correct programs to run, you opened files for those programs so they could run; you browsed , you created a document, you edited and saved it on a word processor, you browsed web pages, you printed a document, you received composed and sent messages across the internet, you manipulated multimedia files, you updated a database and you "talked" with some interactive software. My question on this is: were you aware your were doing these things?

Some, but not all.

Like which?

Like right back at the start you were creating and. I was aware I was talking to the computer obviously because I was getting replies and reading what it was doing. Printing a document was straightforward. Right at the start. Yes I wouldn't have gathered that it was a complex computer program. It seemed too easy to be doing that to be quite honest. If you told me that you wanted me to create a complex program, then I would have panicked straightaway.

Yes, because the system is doing most of the work for you. But have you done any of these things on your own computer before?

A little bit. But not everyday. As you say it is pretty well done for you. I mean: Type in what you want and here it all comes sort of thing. So you are not really searching the same way. For example, if I am using Encarta or something, I don't really have to do much at all. If I want to find out about Botswana, then type it in and bang, it all comes up. So different.

Yes the way the system is set up it is to keep uppermost what you have to do rather than how you have to do it. So you are opening a tutorial, not a particular piece of software.. Did you find doing these things harder or easier with this system than when you have done it before?

I'd say easier. And the fact that you have highlighted the keys that you need to use. And still, on that keyboard I don't think that I know all the controls when it comes to using it for Microsoft or something like that. There's so many .. hold two keys down to do this or that and that's not clearly explained. But as you have set it out this key will do this and this key this. And its again remembering what key will do what task but there is only three to choose from. If one doesn't work we go to the other one and pretty much eliminate each key and one of them will get you to where you want to go. So I think easier to be quite honest and, once you get the hang of it, very straightforward as to how to do it. As I say again to go from activity 1 to 5, I felt a lot more confident as I went and I didn't need to use the F1 or the little lecturer to help me so much because I could figure it out myself.

OK. So .. What do you like about this extramural system, what aspect of it, compared to doing things with your own computer?

It is pretty much, what would you say, standardised. Because you are using the same three keys for every area. So you go into a new area and these three keys are applicable. Whereas if you go into different things on say your normal computer, well sometimes this for that and that is for that, and that and that.. It's not like a standardised...

So it's the consistency across the whole thing...

Yes.

Was there anything else that comes to mind?

Well, I'd call it user-friendly. The instructions are pretty well.. I think the Help, the Handy Hints are great, because it's all there for you and anything you need to do is on the Handy Hints. So I would suggest that if I had that at home as an extramural student, well, you can't go wrong. I feel quite confident that I have got it all, so it's like my Help Phone number. Whereas with my own computer here, I have had problems and rung up and that was six months ago and I still haven't had a reply and now they have gone under, the PC Company, so I'll never find out why the machine was doing these things.

What about aspects that you didn't like compare to doing things with your own computer?

Nothing really. You have got to think while you are doing it, which is not a bad thing. Whereas with Microsoft Word it's just word, font, type.. basically. Other than that I am not using my computer in the same way here [when working at school]. I'm not using it as a learning tool. It's more of a processing tool... It's different for me, because I just have to make sure that the information is on that computer for the kids to access it, and show them how to get it. Whereas here I have got to think because it is something new. But again, once you get the hang of it, it would be straightforward.

So if I understand you, you are saying that you haven't really used a computer for a task so broad..

Well one.. something different and something which I have had to think through step by step to do.

Because there are actually many tasks aren't there? From a computer point of view, it is just about everything that you would normally do separately...

There a couple of bits of language in the scenarios., I'll find it exactly: Find and open Assignment mode again. Like what do you mean again. But I thought. I will go find this and I found it. But I had a quick look back and Where's Assignment Mode, and it wasn't on here, except for when I printed that... But it actually wasn't too hard to find. So I thought it was quite good that I could do that, from only spending half an hour on it, now I can find most things on there.

Anything you would like to see added from that first experience? From the point of making it easier to use?

The only thing where I did panic, well not panic, I knew that I had to rethink about how to keep out of somewhere and I knew that I had to go back but there was no way to go back except exit completely, but I needed to go one step backwards.

in the scenario...

Yes. But then again. If I thought about them I think that I went into the Course Explorer and that helped me through to where I wanted to go But a couple of times I wished there was a back key so I could just go... I know there is one, but it didn't come up as an option..

It's not a back key in the sense that you are talking about...

No. Just can I go back to my last move. Because you pretty much know when you have gone too far the wrong way, but it's not until you get there that you realise that Hey. I shouldn't have gone there. But I didn't want to exit and have to start all over again.

Yes. If you exit you will always come back exactly to where you were unless you choose not to. The way to go back is through the Course Explorer as you discovered.

As you say it took me exactly where I wanted to go. But I was wondering there whether this is going to remove where I am, that is, take it off the screen. With those technical things [in which previous screens were not closed properly by the computer] I was beginning to wonder is this supposed to stay like this or is it something I have done? I didn't realise it was technical. I thought. Am I doing this wrong? And backing up all these screens. I did wonder because I had followed the instructions. Is it supposed to stay like this?

Anything you would like to see removed?

No.

<End of first interview with User 1>

First interview with User 3, 13/10/03

Want I want to do is go through those scenarios one by one and just get your impressions. Were you able to complete the first scenario?

Yes. That was all pretty basic

And the second one which I had also gone through?

Yes that was fine.

So we'll go to the third one which is the Assignment one. Did you have any trouble, did you get caught up on anything there?

Not on this one. Just waiting for it to stop fetching messages [because "fetching messages" message was not cleared from status bar after process was completed]. No that was all pretty fine, that was all pretty easy actually. Putting in your cc it was all in there, everything was there for you.

Did you find you needed to refer to the Help at all?

Yes, I did actually. Just to get a handle around things like clicking on a person to find out where they were from and things like that.

OK You found that one quite straightforward. And what about this fourth one?

That was good. Although I did have to go into the Help when it said to open the Concept Map. And then it said to select the related item. And here I was clicking on the actual word and it wouldn't work. So I went to Help and it said click to the left of it, so as soon as I did that straightaway it was fine.

And you had no other problems with it?

No

You haven't actually studied anything by computer before?

No.

If you actually sit back and look at what's covered here, ... you are doing quite a lot of things with that computer?

Yes. Well I could see that I could use it for mine quite easily. I think. God. It would just make it a lot easier.

You actually set up the software on the machine. When you click on that explorer you are looking for a program and a file. A lot of what you are doing is browsing web pages. You create a document, and you were editing, saving and printing them...

The other good thing was when you log off, then, when you log back on again, you can go back to what you were doing previously. There is no need to open this, open that and go around trying to find out what you were doing before. It was all just there at the click of a button. That was really good.

You were sending, copying, and editing emails. You were manipulating multimedia, you were updating a database and you were having a conversation so to speak with interactive software. How much did you have a concept that you were doing these kind of things?

To start off with probably not. But because I did those first two twice, and when I went through a second time - as you are going through it you get more familiar with it I suppose - it was easier. Just right-click and away I go. It all became a lot easier just from doing those first two twice. The rest of it seemed to fit into place. OK this is where I am going. From your Course Explorer you go here, you go there, and if you want to, you can close everything down and go back to the start.

Have you done many of those things on a computer before?

Cutting and pasting, I have. But on my keyboard I have a Cut, a Copy and a Paste key you see. Sort of some of it I have done before.

Everything you need to study your course you are doing in a special environment. Did you like it that way?

Yes it was easier. Half the problem with doing a lot of things on a computer is that a lot of it is just irrelevant stuff, and you get sick of waiting just to get to your original stuff. I think that is where a lot of people think, I can't be bothered doing that anymore. They lose patience, because you have to go different ways around to do different things. You have to close boxes down, you've got to go back to the start when you log off, you've got to find your way back around again. Whereas this is a lot easier because you just close everything at the same time, you don't have to worry about closing boxes here and there, and when you log back on, it comes straight back exactly to where you started from. It was a lot easier.

What was some of the things that you didn't like about the system, that you ran up against, and wished wasn't there?

No. It actually wasn't too bad. Another good thing that I learnt was click on your bars and it brought it [the selected window] forward. So you could move things to the foreground or to the background as you needed, for copying and pasting the notes, listening to the tutor and things like that. The Help buttons were all visible, and if you didn't find them, you always had the F! button. Not that I didn't find them but the Help screens were good. When you went into Help they were pretty self-explanatory.

And you think something along these lines could be quite useful for your own study?

Absolutely. When I do my study I get a textbook. And I get masses of photocopies of what the tutor pulled out of the textbook basically and I have to sit down and go through all that. If you had it all up on your computer and did it all that way. Emailing him. Or sending him a question. You know: What do I do with this Power Point thing? That was all easy because, if I wanted to do that with my own course, I had to close down, hook up to the Internet, bring up all that. It would take a long time, and that's when I think, I can't be bothered doing all that. I'll just struggle through. Whereas with this one it was right there, and you could just easily do it. Your notes were just down in the side. Push that to the back and, well I'll have that

now. And push that to the front and whack it in. It was a lot easier and more specific to me, rather than ... to everything.

It's just purpose built. It's for the one job. It's not for everything. If I understand what you were saying earlier, when you want to do something on your computer, there's a whole lot of things there that you don't want to deal with..

But you still have to deal with them because they are part of the software

And so you thought this way was..

..easier. Because everything you have got in there you obviously need. At one point or another you are going to need everything that's there. Whereas with a normal computer, and other software, there's a lot of stuff that you just don't need. And to get around to doing the task that you want to do, it's harder to get there because you've got obstacles in the way. Whereas with this it's just all there. With the workgroup it came up on just that topic, things that were relevant to just that topic, rather than every single topic and having to weave your way through it. It was just what you wanted to know, when you wanted to know it, not like other systems...

Of course, some would see that as a downside because you can't see everything at once...

But even with that though, all you have to do is click Exit and it's gone and you can bring up your next thing ... So its easily interchangeable, because when you do log back on it's right there again. You don't have to go back through and do everything again.

You obviously found a couple of things like a message that was out of date, that kept you waiting. Was there anything else like that? Small things:?

I don't think so really. I didn't notice anything.

Anything that you thought could have been there that wasn't? Or would have helped if it was there?

Displaying the [status] messages better [on GroupWork]. That was the only thing. It was there for ages obviously. And I was going: Oh My God. I wondered if something had happened to the connection or something. It was still fetching messages... There can't be that many messages...

No. That was exactly the usability issues that I wanted you to look for. In this case it wasn't so much the big issue...

Yes, I was getting annoyed because it was taking so long. And that's what often happens on the Internet. If it takes that long I just say bugger this and just close it and leave it.

In ending, I just wanted to ask you how you found these sort of exercises as a way to learn the system?

Oh. Really good actually. You know because you are actually reading it and physically having to go and do it. And so you can see ... It says, for example, just try the F1 button and have a look. And often I did just have a look to see what was in there. Yes, it was easier to do it. And even though the content of the course meant absolutely nothing to me it still didn't matter because you still got the general idea of what you were supposed to be doing and how it was supposed to work.

Yes. The main test of the functionality of the system is: Can you learn with it? But we can't really test that in an hour on the machine. So we were really trying to find out: Can you find your way about? Can you use this system? Picture yourself. You are sitting at your place. You get a CD-ROM in the mail, a video and something like this...

The best thing is these three things here [on the Handy Hints sheet].

..the FI, the Menu and the Course Explorer?

You can refer to them at any time. And I am a great one for using the Help, because I think: There must be an easier way to do this. Even just those three things there. Once you're in you can pretty much do it with these. They were a good help.

Let me ask you one other thing. You went into the learning support system, and again the content of it is no more meaningful than the course. What did you think of that side? Imagine you are sitting there at home, and you are struggling with something. How did you find this integrated approach? You know, ask your group? Ask the tutor? Linked together. Did that appeal?

Yes it did. Because the help desk I have got on my one is pretty basic and often you can't find what you need anyway. Because if one doesn't know the wording... There's a lot of concepts that I don't understand and so I think what the hell's that. In here it said "section one". What's "section"? so clicked on the old Help button - oh I understand that now...

<End of first interview with User 3>

First interview with User 2, 14/10/03

The idea of the interview is that I will go through a few points and try to get a feel for how you found using the system. And the first thing I want to do is just go through each of these little exercises/scenarios and ask how you did. So "Initialise", were you able to complete that?

Yes.

Did you have any difficulties?

No. That was very easy, step-by-step, very basic.

OK. Any other comment on it?

No. I found it very easy.

The second one was on starting up and browsing the study guide. Were you able complete that?

Yes.

Did you get caught up anywhere?

No.

You didn't have any major problem with that, that you can recall?

No.

OK. Those two were the ones that we went through together and had a couple of goes at.

The main thing for me was remembering to right-click and bring up the menu. Once you have got that, once you remember every time to right-click and bring up the Desktop menu then it is good.

Well, we can come back to some of those points. Now the third one was working on the assignment. Were you able to complete it?

Yes!

Did you have a few hiccups? I know you had a technical problem with the printer [which won't work unless sheets of paper are manually fed into it]...

It was more my reading the question, the instructions It was instruction 5 that got me. I sat on that for quite a while.

This was the practise assignment?

To search the topics in section one, for the assignment. Where the option was to open it but I didn't know what I was supposed to do with it. I didn't realise I was supposed to open it, check it out, and close it again. But apart from that, the instructions are really clear, right down to the copying and pasting which I have never, ever done in my whole life. And that was probably the easiest part of the whole thing.

Yes I think sometimes with these things, if you haven't learnt other ways of doing it then often it...

It's just so simple. I didn't realise how simple it was....

So that was the assignment one which was number 3, where you will also doing the communicating with the university, or simulating that. And that went all right?

Yep. That was fine. Apart from the problem that the welcome message that I had to delete had already been deleted [as the result of a glitch in re-setting the computer after the previous user had finished their scenarios].

So then we had access a lecture and ask for learning support. Were you able to complete that?

Yes. I was actually. Much to my surprise...

No problems with it?

No. It was surprisingly easy. I kept waiting for a ... I was very focussed on the instructions. Reading them through a few times before I tried it. And every time I tried it, it worked and I thought. Mmm... This is too easy. I can't be doing everything right... There must be something I am doing wrong. But it worked. It was really good.

Yes. I noticed that you ripped through that one quite fast actually.

Oh. Honestly... I understood what the instructions were and so I did it.

Did you find it got easier as you went along?

Yes. Because you.. I believe I got the feel of the menu. And when it says go to the Desktop Menu to Key Ideas, I knew.. by the time I got up to the fourth [scenario]. I knew exactly where it was. It wasn't a problem. But the Concept Map . that got me. I sat on that for a little bit. Because I couldn't quite get ..

What it was?

Yes. And I had to keep thinking, don't try to understand it, just do. But I was trying to understand what the Concept Map was and why it was there and why I had pulled it up. But I found it quite good actually. Surprisingly easy. I've probably done it wrong. Surely I've done it wrong.

Well. If you completed it you haven't done it wrong. With something like the Concept Map there's an aspect of that which requires.., if you don't understand the course and you are not actually doing it and]looking for help on something, then the idea of what you are doing by looking for related concepts.....

And it says, too, select the related item, "conceptual models" and I couldn't select it and it wouldn't work. So I went to the teacher, to the tutor [Help icon] and it said that if it is in blue you can't select it or something like that. And I thought mmm... that's not right. But when I came back I clicked it again and I selected it, I don't know how or why. I did something different obviously and it worked according to the instructions. But the first time I tried it, it didn't work, and I thought: This isn't right.

If you go back and look at the Help again there is a way to select it. It's where you select. But I won't tell you that. But you go back and look again when you have to. You have probably done a lot more with a computer today than you have done before..

Oh absolutely..

Or that you realise that you have done...

Ever. No I can't believe that I have copied and pasted, and sent emails. Cc'd emails, etc. I don't go down that road because it is too hard. I have no idea that it was so easy to get something off a computer and copy it and stick it onto something that you are going to email. It's simple. But I didn't know that before..

Well it's not always so simple..

Yes. I felt quite clever doing that.

You actually set that program up. You ran a whole lot of different programs, you found files that would run with those programs and attached them to it, as you select things through the explorer. You've been to web pages. You've been word processing, printing, composing emails, manipulated multimedia files. And you even updated a database and talked with the software when you were in the Help system. It's actually quite a lot of things. When you were doing it though, how aware were you that you were doing those things?

Which part?

I guess what I am asking you is how conscious were you?

Well. When I did this one here [Discuss The Assignment In Your Group in Scenario 3]. When I copied and pasted into it. I did it without realising exactly what I had done until I had done it. But when I got to this page and I got to the extramural support, I couldn't believe that I was doing it. Like I knew I was doing it but I thought I have got to be doing something wrong here because this is really easy. The little question thing where you can ask a question and then ask again and things like that... Well I got the gist of that and I thought it was pretty cool. But. I have to admit that third one I pretty much flew through that without realising what I had done until the end. And I thought. ooh goodness me I have just done some really major stuff that I have never done before. So yes...

So quite a bit of this you say you have never done before?

Probably 70%. I would be your best guinea pig ever.

So I was going to ask you whether you found it harder or easier? But if you haven't done it before...

I have tried - copying and pasting, and things like that. And I just can't do it. And going in and out of the little boxes and things like that. I find it extremely hard. Like I said to you before, all of a sudden the computer goes "Whoa! That's enough. I can't handle that." And I know it's not the computer. It's me being too busy, trying to go here and do this and do that and not closing the boxes that need to be closed and running things on top of other things and stuff like that.

So did you find this way of managing that for you...?

Brilliant. Fantastic. I think all computers should be like this. I really did. Half the time I get frustrated when I have to work with my computer because it is not as simple as it should be. And this is as simple as it should be. And that's great. If I was to be studying with Massey and there was a program like that, that would probably be one of the main reasons why I would study online. If I had to use that for my studying that would be fantastic because it is so easy to use. As far as using the computer goes. So how much is it? Is it for sale?

It's only a prototype at the moment...

How long do we have to wait. What a great idea.

So what else struck you that liked about this system?

I liked how the right-click would bring up the menu. I used that more than I used the Help. Because I knew that I could go into that and I could go wherever I wanted and I am not going to get stuck anywhere. I'm not going to open... apart from when I went into the library which was trying to get onto the Internet that was off the menu. I don't know how I did that. But just being able to go into the menu like that and surf around and see where everything is. Yep that's what I want, rather than going to Help and trying to look for whatever key word I need. I found that real simple to use.

If this was set up for real then when you visited the library it would mean nothing because you could just go there and then come out again. It's because with this trial we are having to have two connections, one to my place and one to the Internet. Because what would normally happen is you would go to the library and say that is not where I want to be, so you just go again. You don't stop the system by doing that... Did you find anything that frustrated you?

There were a couple of questions that I thought could have been explained a little better. But that was only "lecturers talk" instead of.. there was one that got me and I think I said to you why don't you just say "Close it". I can't remember where it was now ... There were just a couple of things. Yes, this one here. Use the Course Explorer to search the topics You had to pretty much read between the lines. I think that it says just check it out, have a look at it [in the Explorer],and then move on to the next question. But apart from that, as far as the actual program goes... No I thought it was great.

So part of the way that the scenarios were structured was that they start by telling you everything and then the later ones are not telling you everything to see if you can bridge the gaps. You said that after you had completed this part you felt that you understood how..?

Oh for sure. At the beginning, it was pretty basic really. And when you get to the end of what I have done so far it's definitely harder. But from the beginning you are pretty much weaned into it.

So how did you find the Handy Hints?

To be honest the only ones I used were copy and paste

So you kept these things in your mind?

Yes they were in my mind, probably after the second [scenario] I had it in my head that if I right-click the menu is going to come up, and if I can't find anything on there then I am going to press F1 and Help will come up. But it was for the first two scenarios that I still kept thinking what do I do what do I do and I kept looking here [Handy Hints]. But really I only used it for copy and paste ...

Which is something that you weren't previously familiar with?

Yes.

Anything else you would like to see added , not to the scenarios but to the system?

No. I don't think so. I'm really surprised at how easy it was to use for me personally, and I'm no whiz on the computer, Sure I had instructions, But towards the end they were maybe a bit vague. But I did it. I still did it. I completed a task which had something to do with something I had no idea about. But I followed the instructions and it was just so basic. And even when I went into that Library where I shouldn't have been it was very easy for me to get out and start again. I just found it so easy, whereas normally, if I was on my computer and something like that happened, I would end up turning it off, the off button, and going back to it another day because I'd get frustrated. Whereas with this, because I had that right-click, I could go back. I kept going back to it. That was my safety rope the whole time.

<End of first interview with User 2>

Second interview with User 3, 14/10/03

So what you went through today- you monitored your messages; you updated your course material; you explored other materials, basically through the Internet; and then you accessed an online tutorial. Now were there any of those you weren't able to complete?

No. Easy.

You didn't have any difficulties?

No. Not a one.

Did you think it was easier than yesterday?

Yes, it was a bit. Because I thought I was familiar with the layout and everything. So I knew Key Notes were down here and when it comes up I was familiar with it so it was a lot easier.

Actually you did some quite complicated things. Have you ever updated materials on your computer like that before?

No

You actually went to a remote computer..

No I have never done that.

And also you synchronised the data on your computer with that on another one, you searched a database for information on a particular topic, you did a similar search for web sites, and again you worked with an interactive program, only this time it wasn't on your machine it was on a another machine. How conscious were you of doing these things?

Look I was just doing them. I didn't know what was involved or anything so, to me I, I had no idea of what was going on in the background, the work that was going on just to get me to bring up my references and everything else. The other thing was because it was so quick. The information came very quickly, especially on the internet, to what I am used to at home, you have to wait around for the page to come up

you know - 8 items remaining - and ahhh... and then waiting , waiting but that was very quick. Press a button and it all came up fast.

Yes well that is part of the problem of the rural Internet isn't it? Where it's very frustrating trying to work on it. Have you done searches on the Internet before?

Yes.

So you are used to doing that. Well this was a different kind of search in that you were doing it indirectly

Yes. When you do it at home you have to think of your own thing to type in whereas because you have a specific topic and you went through it that way, it brought up relevant sites rather than having to muddle through and having to type in key words and things and do it that way.

So was it easier doing it this way?

Oh yes. A lot easier.

But do you think those restrictions help or hinder?

Well it's hard for me to say because the course material didn't mean anything to me. But in some ways I suppose you would think that it would be a help because it's connecting to your Massey or whatever. They're feeding you stuff that is relevant to what you are learning. So in that way I would expect it to be helping rather than hindering it. Because you expect them to know what you need to know.

Again I would just ask what did you like about doing it that way?

It was easy and it was fast. And that is my two main things at home. It's often quite hard to get your topic. Especially when you search the internet, you type in say 5 key words or 3 key words and then it comes up with 250,000 or a million possibilities and you think "Gee. I'm not going to go through all that". So yes, it was quick and it was easier. Because you didn't have to waste all that time in sorting through the information that you are given as to what's relevant and what's not.

Compared to yesterday you felt more confident.

Navigated myself around it a lot easier because ' it's got only so many main components to it. So you don't have to remember a lot of things. You've got your Explorer and there's all your things there. You've got your tutorials and that down the side, and then you have got your list of notes and self-assessment and that, and so it was all there- they are sort of your main ones and from that you can just branch off to what you need, rather than having to... If it wasn't that simplified then you'd be diving around trying to find things.

So do you think that you would be able to use it by yourself now?

Yes. I reckon that I could, quite easily.

Do you want to try?

Yes.

If you want to you can have a quick try when Mim is finished and see how it pans out. Any other comments on it, any other questions? Plus or minuses. Anything that more you like about it or don't like about it.

No. Like yesterday. There was nothing really to criticise about it, because it was easy to use, with your three things here. I felt confident enough to whiz around, because you've got all the help things and you are not thinking, you know like when using your own computer "OK I'll go for help.". And you see all these

things and you have no idea what they mean? It's all this irrelevant drivel and I'm going to go "That's just a waste of time anyway." At least here the content is relevant to what you are doing.

Do you find, when you are using your computer at home with Windows, do you find things are more complicated than they need to be?

Oh absolutely. Like I was trying to do something with a spreadsheet so I went through all the help options and the index and I searched for it and nothing came up. And it knew that I could do it, that it was possible to do it, but to actually find out how to do it was really hard. Especially when the help is designed to do that. It was really hard to find out what you were supposed to be doing. So I just leave it.

So you think that the approach of the extramural system which is minimalist rather than...

Yes. I think that people think that with computers you have to have all the bells and whistles and make it really complicated so that people are more impressed by it, but if its...the simpler it is, the more people are going to use it. Because it is easy to use and you don't have to worry about all the bells and whistles and everything else going on because it is all just there.

And you don't... If you were studying one of your courses using something like that you don't mind the options being limited the way it is.?

No I would rather have that because sometimes you can tend to just go off, you start one train of thought and you go off on that and two hours later you have gone way off track because there's a mountain, a river, of information out there and so you get trapped looking at all that stuff when all you needed was the first little bit. There is a tendency sometimes to get a bit of overkill on the information.

Well that's the story of computers... too much information..

Oh I'll get onto the Internet and start and then click onto another site and then another site and then ten sites later you are still none the wiser.

<End of second interview with User 3>

Second interview with User 2, 14/10/03

We are looking at the second part of the scenarios. What you did was you updated your course materials , you went and got your emails from the university, you searched the internet for related materials, you worked with some interactive software remotely as opposed to on your computer. Were you able to complete them all?

Yes. I did finish eventually. But I struggled.

So you had a couple of difficulties. So let's take the scenarios one by one. The first one was update course material?

That was fine. That took two seconds...a piece of cake

You had no problems?

No. As I was doing it I was thinking this is very easy.

So you updated your course material?

But then I got to update your messages. And I got to close all my new messages and then it had to find and open Assignment mode again. And I didn't know where Assignment mode was and I went through... I right-clicked to the menu. But I didn't go any further than that. I got into the Explorer and the topics but I didn't go any further so I ended up going back to working on the assignment , [scenario] number three. And I kept reading about using the course explorer to search the topics. I kept reading that and reading that and then I realised that assignment mode is in there, it's under the contents and I had to go through them all to find it and I found it, but it was..

So you got there...

I got there after taking a deep breath and calming down. I couldn't relate the assignment mode, the assignment thingy that I did earlier, to this part. I don't know why. I got it eventually. And once I got it I was away.

You were shifting your mind from one place, jumping to another..

One minute I'm looking at my messages and all of a sudden I've got to go back to my assignment mode and I'm thinking, "I can't do that. I'm in messages." And that's the thing, you can't do that generally on a computer because you lose something or you've got to close it all down completely and put it away before you open something else up, I find. Well, at least, that's how I do things, whereas this time I didn't have to do that. I could to and fro and I think I struggled to adjust to that.

To understand that you could do it?

Yes.

Of course the other difficult point I this was that you were in one part of the course but it was in another. And so you had to ...

..to look for it. And that's what I couldn't understand. Yes you're right. I was in 1.3 and assignment mode was in 1.1 I think...

1.4...

1.4. And I couldn't... And I'm thinking I can't go into there because that's not part of what I'm doing at the moment. And so it was just following the instructions really.

So you were able to complete it but you had to think at one point how to find that assignment mode when it wasn't immediately there. OK. So if we now look at exploring a topic. You explored some learning resources on the Internet. Were you able to complete this one?

Yes.

Did you have any difficulties?

Mmm... I did. #10 where it says ask "Where do I find more?". Well I just asked "explain usability?". And it didn't work. And I thought. And I did it twice and I thought well there should be more information. I read the instruction and I retried until there was no more information available and I thought, well there's not any information anyway. So obviously I am doing something wrong so I went back #9 and that was when I saw it said [on the screen]: "Where do I find more on "and so I clicked that and I was away.

So you were able to find your way through..?

Yes. Once I focussed I was fine. All it was, was the fact that I didn't read the instruction, I just read, and that's one of my traits, "ask about it", "find out about it " so...

So you just did what you had done last time that you used the support system? When you went there the previous time you asked a different question.

Yes. Well actually I did read the question and down on the bottom where it says ASK, I actually clicked that. And I thought, no, because that's sending an email to the tutor and I don't want to do that. Because it doesn't say that. So I thought, how do I ask. So I clicked on the usability adoption and I thought maybe I am supposed to wipe it and type in: "where do I find more?". No. That doesn't work. And then I went back again because I thought that isn't right, and that's when I saw it, right underneath "explain" [in the list], "where do I find". So it was just a matter of looking.

So. You didn't go to the Help?

No. With the ask where do I find?

mmm...

No. Because I knew that it was there on that page... But I did use the help this time around a hell of a lot more than I have before. Because I was having...it was # 13 locate Bad Designs. Well I hadn't located it because it wouldn't connect and I kept going to Help thinking that I had done something

Yes. There was a problem with the computer connecting to the Internet and that is why we had to restart it...

Yes. Once you did that then I picked it up straight away.

So the difficulty you had here was that you weren't asking the right question, you were asking it to explain and it took a bit to catch on...?

Yes I wasn't focussing. It was by no means that the instruction was vague. The instruction was there in black and white. But it didn't, I couldn't relate that to...

No. No. That's always one of these things... So we'll look at the interactive tutorial. Did you have any problem there?

No. I got smiling faces there.

OK. So you completed that one without..

Yes. Piece of cake.

Again if you sat back and looked there's quite a complex range of things that you were doing and one of them was you were going to another computer across the internet and downloading files from there onto your machine. Have you done that before?

No. No. No. No. Never, ever.

And you were synchronising data on your computer with another one which is when you were updating your computer.... OK we won't pursue that. But some other things .. searching. You searched a database for information on a particular subject, which was the explain, and you also searched for and located a website on a particular subject. Have you ever done those sorts of things before. Have you searched on the Internet?

Yes. But I've never downloaded it and sent it to my friends or anything like that, because I don't know how to do it and also downloading is very painful out here. Do you know what I mean? I don't have the knowledge to do it. It's too hard.

But if you are looking for something on a particular subject, you have been able to search on that subject and find something?

Yes. And if I've got something I want to access I'll print it off and I've got it. I've got this thing off the Internet. Not I've emailed that thing to you. Because it's too hard, I can't do it. I don't know how to do it. But I've just done it ... I've got no idea how to do it but I've done it. It's just so basic. The instructions are so basic.

So, in one sense, you weren't quite aware of what you were doing while you were doing it?

I was aware. But I find it really odd that I'm doing this and it's so easy. If you know what I mean. I just find that really odd. Like updating your messages and things like that, and taking stuff off one computer and putting it on this computer and then sending it off to another to me is just way over my head. But obviously it's not over my head it's how ... if someone gives you instruction and it's easy to follow you can pretty much do anything, can't you? You don't have to be extremely intelligent to be able to do things like this.

One of the ways that this extramural system has been designed is to try to make those processes like that simpler. More accessible, even invisible....

So how did you find doing these things? I presume that you are saying that doing these things through the extramural system was easier for you, or possible for you.

It was.. If I could do my study using this type of program it would be so easy. The instruction... I know that there were things that I got confused on but overall it was very clear what I was doing to the point where I thought: this is too clear, and too easy. I cannot surely I'm not ...

Doing these things?

--copying and pasting this and putting it on my thingy here and I'm going to email it to Joe Bloggs. So that to me, blows me away I find it I think it's great. Very, very easy. And I'm not the most computer... I like the computer and things but there are a lot of things I just give up. I just won't do it because I can't, I can't. You've got to be a brain box to use it.

So part of the problem for you if you were trying to study using your computer that you would be worried about not being able to complete things?

Yeah. Because I mean.. It's easier to do the basic stuff, do the hard yards, like go to the library and things like that and photocopy pages out of a library book, or ...Because its just easier. But it should be easier on a computer but I find that it is not. But if I had this sort of system it would be very easy. See what I mean?

I know exactly what you mean. Don't you worry. Travelling 70 km to the library because its easier than finding it on the Internet...

When you were going through these exercises your mind was probably on completing rather than on... Were there anything that you would have liked to see done better, things that you ran into. I mean putting aside the question of this could have been explained better in the scenarios but on the system itself?

I don't think so.

Well. Let me put it around another way. What was it that you liked about it...

Like I said earlier, I thought a couple of instructions were a bit vague..

I was interested because some of them were deliberately vague to see whether you had gained enough of an understanding of the basics of how it works that you could find your way out of the hole again?

I felt that I did. I don't know. I guess that you would know more than I would. But I felt that overall like there were times, there were two times when I thought I can't do this. I just can't do it. And the second time was when the computer stopped, like it crashed, or died or froze or whatever, but it wasn't my fault. And then the other time was the assignment one. Find or open your assignment. I couldn't get my head around that . But I think that my head was in a different framework of what was possible... But there were times that I thought, well , the question was vague, and I thought you've said this but I'm going to try this, I'm going to try and do something else and I did it. And I thought, "Ah yes. You do it like I have done it anyway". Once I had done it was obvious.

So you've spent a couple of hours on it, at most,. Do you think that you would be able to find your way around it by yourself?

I think so.

There was one thing that you said that I was interested in, the problems of the Internet out here. What do you mean by that?

Like when you open you emails, to check your emails and it takes half an hour to receive two emails. Like when we have this. I don't know if you have it at your house, but the phone overloads and if you are not on the Internet it rings every now and then. So if you are online then it just disconnects. And downloading something...

Like a web page or something?

Yeah, can take 31 hours to download. It's ridiculous but it's true. It's crazy.

I'm asking that because I believe that this is one of the problems that is least recognised in the towns, just how difficult it is...

You know how your computer works at I think its 27000 ...

...kilobits per sec

Well I have seen it at 1. I know that it should be 26,000 or 24,000 and it is one, literally just one. You can get a connection. You can dialup and get online But you can't get anything. Not even a home page. And you're paying your \$2.50 per hour for nothing.

<End of second interview with User 2>

Second interview with User 1, 14/10/03

This time you were attempting the last three of the scenarios -updating your course material and messages, exploring some resources mainly through the Internet and the interactive tutorial? Starting with the first one. Did you have any problem?

Just with the computer still being connected to the Internet [from the previous user] instead of the university to start off with. Other than that, no.

You found it straightforward?

Pretty straightforward, yeah.

The second one. When you explored resources on the web? Any difficulties there?

No.

And the interactive tutorial and updating self-assessments? Any difficulties there?

The last one? No. Interesting though. I quite like the Barney and the other bits and pieces I had a look at. Where it said to have a squizz at them.

Oh. Barney?

The interactive part. Barney and the telephone [in the tutorial scenario].

In that short time you actually do quite a few tasks, one of which is transferring data files from a remote computer and loading them onto your computer. Have you done that with your own computer?

No.

Not consciously.

No

OK. And another thing that you did was you were synchronising the data on your machine with that on another. Were you conscious of doing anything like that?

I haven't done anything like that at all. No.

And were you aware of what was going on?

Yes.

Now . You also, when you were accessing the extramural support you were using that in two ways. One way you were searching for information on a particular topic. Secondly you were searching for relevant web sites. Have you done that sort of a search for information before?

Yes.

In what way?

Was I using a computer to do it. And what was I looking for?

Yes.

Basically topics of study for school. So. Looking up relevant web sites, etc for say American Indians. Anything related have a look at. If it's worthwhile tag it. If it isn't leave it. So basically topics that the kids study at school.

OK. Obviously with the extramural system the search you make is more constrained because it is aimed at particular topics and things. Did you find it easier to do it this way compared to what you have done before? or harder?

Easier, basically. A whole lot of the time you get a whole lot of stuff. Say if I'm getting it off the Internet, I get 300 sites of which 275 of them are useless. It's a lot slower on the internet the other way than your way, because I have to physically open it and have a look at it to see if its relevant or not, and then go on to the next one and it's a long process. Whereas here, it's telling you what, obviously it's trimming it down

to what's relevant and what's not. So I presume for web sites that I have been given there, that they will be useful for what I want, not may be useful, will be useful. Even with the one that I had to delete because it was not useful. That was obviously just for this case scenario. But for me, from what I saw there, I would think that it is useful, whereas go onto the Internet then....

So you think there are some advantages to using something where it is already constrained in that way.

Yes. I think so with some.. It all depends with the study. I would suggest though that if you're lining it up... It's hard really if they're a student, like a university student, I can see if I'm reading this right, you don't want to have it all there for them. I mean there is obviously some research involved for them. But I do think with this extramural thing yes, if those addresses are there. I mean they still have to choose what's relevant and what's not don't they. So if they click onto your web sites that you've got listed, then they still make the choice whether they use that information or not.

I guess for the purposes of this, the most important thing was whether it was easy to use.

Very easy.

You also worked with an interactive program again. Well, more than one. The support system - your were interacting through the buttons with the system. Then you went to the web site and there was interactive software on a another computer, a tutorial. Overall, thinking about what you have done today, or since you have been looking at it, what aspects did you like compared to doing things with your own computer?

The limited ..buttons, shall we say, that you can use. I liked that. That there were only those three buttons to link to get round your software to the different areas. That it's very transparent that you can .., it's all very clear what you use to do what you want it to do. There's no hidden features. Like if you talk about my own computer there's so many hidden things that you stumble across. But here it's very straightforward. And I think for that, if you get yourself in trouble, it's very easy to get yourself out of trouble. Whereas, sometimes on my own computer I find you've got alt+ctrl+del. You've got to get out to start again, because you have no idea where you have gone and what you've done and the computer won't tell you what you've done, and the computer... Sometimes you can click on Help. Well, if you click on help on my computer, quite often it's not the scenario where you have got yourself, if you know what I mean. It'll say dah dah dah, when you know you haven't done that, you have done something different. Whereas here, if I click on Help I have found every time what I need to do and its been able to solve it for me. So I think that it is really user-friendly compared to my computer.

Clearly in part it's because it is special purpose. It's only doing one job. So you think that in the context of a learning scenario you think this is useful?

I think especially for someone who has purchased a computer to start their study and this is the first time they have had to come to grips with a computer as well as their new course. It's not intimidating at all. Like someone within half an hour will have the grasp that they can't do any damage, which a lot of people I think feel when they first get on their computer. And that it's very easy to find your way around the whole thing with those three keys. To me that's the most appealing thing about it. Like I think that my mother could even do it.

Is that a compliment or..?

Compliment. Once I had taught her how to turn the computer on I think she would be able to do it.

Were there things that irritated or irked you, or just bothered you a little?

No. Just that computer. Nothing about the program. Even when I had to download a paragraph of that information into my notes. Well the first time I went to try that I hadn't gone and clicked onto one of those topics. So all I got was the web address in my notes. So I thought, OK, I haven't gone far enough into that. So I just clicked on something and then again, all my notes came up. And again it was one of those things where you could very easily see OK I haven't gone far enough here. Just keep going. And you're not doing any damage.

So when you found you were somewhere where you shouldn't be, you were confident you could find your way out?

Yes. And to say again. If in doubt, press the help.

One of the trickiest points in these scenarios is when you are updating your messages. You are in one part of the course and you have to go to another part to find the assignment. Did you have to think about that?

No. I did think about what's going to happen here. How's this going to go? But I thought I would do it and watch and see what happens. But it went very easily, to go from one are to another, and very quick as well. So that's another advantage with that, I think, Russell. It's quick.

Have you got in mind doing something on the Internet? And it takes forever?

Well yeah I suppose...

What have you got in mind then?

I'm just thinking of computer programs in general, basically. Some, when you are jumping from one field to another...Oh I have got to shut that down, load that up. And not with changing disks , but just in general. But here, going from that one to the other was very quick to get from here to there. It was a matter of within 5 seconds I was gone from one to the other. There's no sitting and please wait while it does this and that...

From the perspective of being user-friendly, is there anything that you would like to see added?

Just from what I have seen today? I think that for me was easier today. Like #3 I found probably the most time consuming or thinking. But again it's just... I wondered today whether I would remember, without using my Handy Hints. I wanted to see if I could remember and I didn't have to use that guide. I looked at it once but no I remembered everything. And I think that, no I couldn't see anything that was missing. That message part was confusing ["Fetching messages" message remaining in status bar], but I didn't take any notice of it., because I knew that it was a trial and unless you had sent me any messages there wouldn't be any anyway and again it was pretty straightforward...Wait until it has finished it's thing before you go anywhere else. I did wonder when it said it was fetching and there were 8 new messages. These messages were coming up, but when the cursor was ready to go, I thought no and so I carried on.

That is a real problem, only a small one, and an easily fixable one...

Some people could sit there for ages waiting for these messages to come up, because when you come to that screen it says no messages anyway and when you update my messages, I was waiting for my messages to wipe that no messages out.

Any other little things like that?

No. that was the only one

Did you any problem asking a different kind of question to the support system? You ask them to explain and then you ask them where do I find?

No. That was pretty straightforward. Again I thought I sorted that out yesterday with the ticks.

You are not currently studying are you? Do you think that something like that interactive help would be quite useful if you were?

Definitely. I would be quite keen on doing it that way. Even when I was internally a student, getting hold of a lecturer was quite difficult, and leaving messages for them on the door, or notes and things like that was still days... It could be two days before you got to touch base or a message was left in my pigeonhole. So here for me, it was instant. Well if I have sent the message, to me it's going to be updated. ...well 1) I feel that I have got in contact with the lecturer or my tutor and, 2) that I think there is a lot more assistance there in that program than having to go and track down my lecturer and my tutor, etc. And again, with the books and things like that. It's all there, at your beck and call. To me in some ways you are at an advantage that way than if you are an internal student, because you have got limited resources on the shelves and its first in first served whereas here I mean its all... a great big cobweb, which you can work your way around.

So how confident are you now that you could go back and use that system by yourself?

Very. Without anything. I think that the Handy Hints would be a good thing to be distributed with it. But I think that if you came tomorrow and asked me to try do this, I definitely, I am 90% sure I could do what you want me to do.

How useful do you think those scenarios, or set of exercises are as a way of teaching you to learn the system.

Good. A couple of times there I was thinking ooh... am I going to be able to do this. But no. I think reading it. If I sat there reading it without doing it, I'd probably be panicking, but as you go through it, they build your confidence, if anything. To me it built my confidence more and more that I can do this. They may be a little bit too much because I found I may have jumped a couple of times, being a clever male... But I think that if that gives you each scenario that you may come across, then if now you can whip back here and do this or whip back there and do that, then I think that it is really good practice. Because on those later ones it is just assumed that you knew how to get there to do that. It wasn't telling you, it was telling you go and get the notes in 1.1, go back to 1.1, it wasn't telling you how to do it. So it was just putting into practice what you have already learnt.

So you think you now how a much better feel for how to use this thing than when you started?

Yes.

In terms of overall usability, 1-10. How would you rate it?

I would put it at about an 8.

Thank you.

<End of second interview with User 1>
