

1998

Software flexibility in a web environment

Michael Layng
Edith Cowan University

Follow this and additional works at: https://ro.ecu.edu.au/theses_hons



Part of the [Software Engineering Commons](#)

Recommended Citation

Layng, M. (1998). *Software flexibility in a web environment*. https://ro.ecu.edu.au/theses_hons/744

This Thesis is posted at Research Online.
https://ro.ecu.edu.au/theses_hons/744

Software Flexibility in a Web Environment

by

Michael Layng BSc (Computer Science)

**A Thesis Submitted in Partial Fulfilment of the
Requirements for the Award of**

Bachelor of Science Honours (Computer Science)

**Faculty of Communications, Health and Science
Edith Cowan University**

Date of Submission: 11 December 1998

Abstract

Flexible software systems are designed to be able to adapt to changes in their environment without the need for programmer intervention. This allows companies and institutions to avoid the cost of hiring a programmer every time a business rule changes or a data structure is modified.

The techniques of designing and implementing flexible 3GL software systems have been proven as a means of reducing the high cost of software maintenance caused by the changing of systems requirements. Such systems have been in operation in the European financial sector since the early 1980's. Edith Cowan University is participating in current research investigating the effectiveness of porting these techniques into the realm of modern 4GL environments such as Oracle.

The explosion in popularity of the World Wide Web in recent years means that software developers will now be looking at the Web as the primary means of deployment for many of their applications. Web application developers will face the same high maintenance costs that have always plagued software developers.


This study investigates the porting of techniques of software flexibility from 3GL and 4GL environments to the World Wide Web environment, and any benefits that this may bring to users and administrators of World Wide Web systems. This investigation involved the development of a flexible, Web-enabled system to allow unit coordinators at Edith Cowan University to manage access to Web-enabled unit materials. University staff were invited to comment on the usefulness of the flexibility of the system.

Declaration

I certify that this thesis does not, to the best of my knowledge and belief:

- (i) incorporate without acknowledgement any material previously submitted for a degree or diploma in any institution of higher education;
- (ii) contain any material previously published or written by another person except where due reference is made in the text; or
- (iii) contain any defamatory material.

Signed:


Michael Layng

Date:

8 2 2014

Acknowledgements

There are a number of people that must be acknowledged for their contributions to this thesis. First and foremost, very sincere thanks must go to my supervisor, Ms Jean Hall. Despite her heavy commitments Jean has made herself available to answer my many queries throughout the year. She has provided a friendly and supportive shoulder to lean on, and has taught me a great deal about the field of academic research. I have enjoyed this year immensely and look forward to working with Jean again in the future.

The participants in this research, Mr Justin Brown, Ms Marie Corrigan, Mr Bill Laidman, Mr Geoff Lourens, Ms Jan Ring, and Dr Timo Vuori all gave up valuable time and I thank them for this. I would also like to thank those staff members who expressed interest in the research but were unable to participate for various reasons.

Mr Lourens also provided a great deal of technical support and advice throughout the year. The Developer 2000 Server product that was used was new to both of us and Geoff's understanding of the various other Oracle products was invaluable. Ms Penny Cookson of SAGE Computing Services and Mr Glenn Nicholas, the president of the local Oracle Users Group, also provided technical advice throughout the year.

Finally, undertaking a thesis on top of full time employment obviously demands a lot of time and energy. Special thanks must go to my girlfriend, Miss Lynley Evans, for her support, encouragement and understanding throughout the year. My parents, Bill and Sue, and my sister, Melanie, also continued the support and encouragement that they have provided for my entire life.

List of Figures

| | |
|--|-----|
| Figure 1.1 Internet Domain Servers by Year..... | 4 |
| Figure 2.1 Categories of Maintenance..... | 11 |
| Figure 3.1 Web Application Server..... | 33 |
| Figure 4.1 Data Model for the Unit Material Management System..... | 45 |
| Figure 4.2 Example of the Security Table..... | 54 |
| Figure 4.3 OWS - Web Forms Structure..... | 55 |
| Figure 4.4 Accessing the World Wide Web Through a Firewall Server..... | 56 |
| Figure 4.5 Student View Form..... | 61 |
| Figure 4.6 Material Access Conditions Form..... | 62 |
| Figure 4.7 Default Restrictions Form..... | 63 |
| Figure 4.8 Student Pop-list..... | 64 |
| Figure 5.1 Oracle Security Manager..... | 72 |
| Figure A.1 Static HTML File..... | 107 |
| Figure B.1 Entity Relationship Diagram..... | 111 |
| Figure B.2 Unit Offerings Form..... | 113 |
| Figure B.3 Unit Coordinator Form..... | 114 |
| Figure B.4 Students Form..... | 115 |
| Figure B.5 Course Location Form..... | 116 |
| Figure B.6 Teaching Term Form..... | 117 |
| Figure B.7 Material Types Form..... | 118 |
| Figure B.8 Material Language Form..... | 119 |
| Figure B.9 Materials Form..... | 120 |
| Figure B.10 Assign Materials Form..... | 121 |
| Figure B.11 Assign Material Access Rights dialog..... | 123 |
| Figure B.12 Default Access Rights Form..... | 123 |
| Figure B.13 Enrolment Form..... | 124 |
| Figure B.14 IP Domain Form..... | 125 |
| Figure B.15 Time Window Form..... | 126 |
| Figure B.16 The Material View Console Form..... | 127 |
| Figure B.17 Code Listing for Hiding Application Menus..... | 129 |
| Figure B.18 POPULATE_UNIT_LIST Code Listing..... | 130 |
| Figure B.19 POPULATE_MATERIAL_LIST Code Listing..... | 133 |
| Figure B.20 UMS_LOGIN Package Listing..... | 137 |

List of Tables

| | |
|---|-----|
| Table 2.1 Dublin Core Metadata Elements..... | 25 |
| Table 4.1 Unit Material Management System Functions per Role..... | 50 |
| Table 4.2 CGI Environment Variables. | 66 |
| Table 5.1 Research Subjects. | 74 |
| Table 5.2 Responses to Question 1. | 76 |
| Table 5.3 Responses to Question 2. | 77 |
| Table 5.4 Responses to Question 3. | 78 |
| Table 5.5 Responses to Question 4. | 79 |
| Table 5.6 Responses to Question 5. | 80 |
| Table A.1 Web Server Listener 'Basic Configuration' Settings..... | 104 |
| Table A.2 Directory Mappings. | 105 |
| Table B.1 Menu Structure..... | 112 |

Table of Contents

| | |
|--|------------|
| ABSTRACT | II |
| DECLARATION | III |
| ACKNOWLEDGEMENTS | IV |
| LIST OF FIGURES | V |
| LIST OF TABLES | VI |
| TABLE OF CONTENTS | VII |
| CHAPTER 1: INTRODUCTION | 1 |
| 1.1 BACKGROUND..... | 1 |
| 1.2 SIGNIFICANCE..... | 2 |
| 1.3 PURPOSE..... | 6 |
| 1.4 RESEARCH QUESTIONS | 7 |
| 1.5 ORGANISATION OF THE THESIS | 7 |
| CHAPTER 2: REVIEW OF LITERATURE | 9 |
| 2.1 GENERAL LITERATURE..... | 9 |
| 2.1.1 <i>The Cost of Software Maintenance</i> | 9 |
| 2.1.2 <i>Static Software in a Dynamic World</i> | 12 |
| 2.1.3 <i>Flexible Systems</i> | 14 |
| 2.1.4 <i>Methods of Providing Software Flexibility</i> | 15 |
| 2.1.5 <i>The Importance of the World Wide Web</i> | 18 |
| 2.1.6 <i>Flexibility and the World Wide Web</i> | 18 |
| 2.1.7 <i>Implementation Tools</i> | 20 |
| 2.2 PREVIOUS FINDINGS | 22 |
| 2.3 RELATED STUDIES..... | 22 |
| 2.3.1 <i>Flexibility in 4GLs</i> | 22 |
| 2.3.2 <i>Metadata on the Web</i> | 23 |
| 2.4 RESEARCH METHODOLOGY | 28 |
| 2.5 SUMMARY | 29 |
| CHAPTER 3: METHOD | 30 |
| 3.1 INTRODUCTION..... | 30 |

| | |
|--|-----------|
| 3.2 DESIGN..... | 30 |
| 3.3 PRODUCT..... | 31 |
| 3.4 ENVIRONMENT..... | 32 |
| 3.4.1 Oracle Technology..... | 32 |
| 3.4.2 Development Environment..... | 35 |
| 3.5 SUBJECTS..... | 36 |
| 3.6 PROCEDURE..... | 37 |
| 3.7 DEMONSTRATION..... | 38 |
| 3.8 QUESTIONNAIRE..... | 39 |
| 3.9 DATA ANALYSIS..... | 39 |
| 3.10 RESEARCH LIMITATIONS..... | 40 |
| CHAPTER 4: THE UNIT MATERIAL MANAGEMENT SYSTEM..... | 43 |
| 4.1 INTRODUCTION..... | 43 |
| 4.2 SYSTEM DESIGN..... | 44 |
| 4.2.1 Requirements..... | 44 |
| 4.2.2 Data Model..... | 44 |
| 4.2.3 Functional Analysis..... | 49 |
| 4.3 IMPLEMENTATION ISSUES..... | 51 |
| 4.3.1 Web Forms Installation..... | 51 |
| 4.3.2 Obtaining the IP Address..... | 52 |
| 4.3.3 IP Address Issues..... | 55 |
| 4.3.4 User Roles..... | 57 |
| 4.4 FLEXIBILITY ISSUES..... | 58 |
| 4.4.1 Common Login Screen..... | 58 |
| 4.4.2 Hiding of Restricted Information/Functionality..... | 59 |
| 4.4.3 Graphical Control of Material Access Conditions..... | 61 |
| 4.4.4 Dynamic Pop-list Values..... | 63 |
| 4.5 SHORTCOMINGS OF DEVELOPER 2000 SERVER..... | 65 |
| 4.5.1 Access to CGI Environment Variables..... | 65 |
| 4.5.2 Web Browser Compatibility..... | 67 |
| 4.5.3 Performance..... | 68 |
| 4.6 SUMMARY..... | 69 |



| | |
|---|------------|
| CHAPTER 5: RESULTS | 70 |
| 5.1 RESEARCH QUESTION 1 | 70 |
| 5.1.1 Access Control..... | 70 |
| 5.1.2 User Enhanceability..... | 71 |
| 5.2 RESEARCH QUESTION 2 | 73 |
| 5.2.1 Subjects..... | 73 |
| 5.2.2 Questionnaire Responses | 75 |
| 5.2.3 Other Comments..... | 81 |
| CHAPTER 6: DISCUSSION | 83 |
| 6.1 INTRODUCTION | 83 |
| 6.2 RESEARCH QUESTION 1 | 83 |
| 6.3 RESEARCH QUESTION 2 | 83 |
| 6.3.1 Subjects..... | 83 |
| 6.3.2 Is There a Market? | 84 |
| 6.3.3 Benefits..... | 85 |
| 6.3.4 Useability | 87 |
| 6.3.5 Flexibility vs Control..... | 87 |
| 6.3.6 Hiding Forbidden Functionality | 89 |
| 6.3.7 Suggested Enhancements | 91 |
| 6.4 SUMMARY | 92 |
| CHAPTER 7: CONCLUSION..... | 93 |
| 7.1 FINDINGS | 93 |
| 7.2 FUTURE RESEARCH..... | 95 |
| 7.3 SUMMARY | 96 |
| DEFINITION OF TERMS..... | 97 |
| BIBLIOGRAPHY | 100 |
| APPENDIX A: ORACLE DEVELOPER 2000 SERVER CONFIGURATION.. | 103 |
| A.1 INTRODUCTION | 103 |
| A.2 SET UP THE WEB SERVER LISTENER | 104 |
| A.3 SET UP THE JAVA DEVELOPERS KIT | 106 |
| A.4 CREATE THE STATIC HTML PAGE | 107 |
| A.5 SET UP THE FORMS SERVER LISTENER | 108 |

| | |
|--|------------|
| APPENDIX B: UNIT MATERIAL MANAGEMENT SYSTEM SPECIFICATION | 109 |
| B.1 INTRODUCTION | 109 |
| B.2 SYSTEM REQUIREMENTS..... | 110 |
| B.3 ENTITY RELATIONSHIP DIAGRAM | 111 |
| B.4 MENU STRUCTURE..... | 112 |
| B.5 ADMIN MENU FUNCTIONS | 113 |
| <i>B.5.1 Unit Offerings</i> | <i>113</i> |
| <i>B.5.2 Unit Coordinators</i> | <i>114</i> |
| <i>B.5.3 Students</i> | <i>115</i> |
| <i>B.5.4 Course Locations</i> | <i>116</i> |
| <i>B.5.5 Teaching Terms</i> | <i>117</i> |
| <i>B.5.6 Material Types</i> | <i>118</i> |
| <i>B.5.7 Material Language</i> | <i>119</i> |
| B.6 MATERIAL MANAGEMENT MENU FUNCTIONS | 120 |
| <i>B.6.1 Materials</i> | <i>120</i> |
| <i>B.6.2 Assign Materials</i> | <i>121</i> |
| <i>B.6.3 Assign Material Access Rights</i> | <i>122</i> |
| B.7 UNIT MANAGEMENT MENU FUNCTIONS | 124 |
| <i>B.7.1 Enrolments</i> | <i>124</i> |
| <i>B.7.2 IP Domains</i> | <i>125</i> |
| <i>B.7.3 Time Windows</i> | <i>126</i> |
| B.8 STUDENT MENU FUNCTIONS..... | 127 |
| B.9 PROCEDURE LISTINGS..... | 128 |
| <i>B.9.1 Menu Hiding</i> | <i>128</i> |
| <i>B.9.2 Populating the 'Unit List Box' on Student Form</i> | <i>129</i> |
| <i>B.9.3 Populating the 'Material List Box' on Student Form</i> | <i>131</i> |
| B.10 UMS_LOGIN PACKAGE..... | 134 |
| APPENDIX C: DATA DEFINITION LANGUAGE | 138 |
| C.1 INTRODUCTION | 138 |
| C.2 MAIN DDL SCRIPT | 138 |
| C.3 DROP ALL OBJECTS | 139 |
| C.4 CREATE TABLES | 140 |
| C.5 CREATE INDICES | 142 |

| | |
|---------------------------------------|------------|
| C.6 CREATE CONSTRAINTS..... | 144 |
| C.7 CREATE SEQUENCES | 148 |
| C.8 CREATE PUBLIC SYNONYMS | 149 |
| C.9 CREATE USER ACCOUNTS..... | 150 |
| C.10 CREATE DATABASE ROLES | 151 |
| C.11 GRANT ACCESS PRIVILEGES | 152 |
| C.12 INSERT TEST DATA | 153 |
| APPENDIX D: QUESTIONNAIRE..... | 156 |

Chapter 1: Introduction

1.1 Background

Lehman's 'First Law of Software Evolution' states that "programs concerning real-world requirements either undergo continuous change, or become progressively less useful." (Lehman cited in Hall & Ligezinski, 1997a, p. 2). Whenever a change in requirements occurs, the software system must be modified, typically at the source code level, to reflect that change. Such changes contribute significantly to the maintenance budget of a software organisation.

Pressman (1992, p. 667) predicted that if appropriate steps were not taken, software maintenance could account for 80 percent of an organisation's budget by the mid-1990's. There are many different opinions on what these 'appropriate steps' are.

Many software professionals support the idea of 'component software'. This includes the object-oriented methodology as well as more recent developments such as the Component Object Model (COM) technologies from Microsoft Corporation. The component software approach is based on systems being made up of separate components, or objects, each of which is a self contained entity in its own right. The basis of this approach from a maintenance point of view is that a single component may be altered without affecting the rest of the system (Chapell, 1996).

Computer Aided Software Engineering, or CASE, is another popular method that seeks to reduce the cost of software maintenance. CASE tools allow a developer to define a

computer system graphically or by entering a number of properties or parameters. Some CASE tools also support reverse engineering - the ability to generate a representation of the design of a system from existing source code (Pressman, 1992). While CASE tools have no doubt contributed to the software development effort, technical knowledge of software design is still required to alter the behaviour of a system.

An alternative solution to the software maintenance challenge that has gained popularity in some sectors of the software development community is that of 'flexible software'. Johnson, Ligezinski & Woolfolk (1996) define a flexible system as "an automated system that can be re-synchronized with changes in the real world system through user-based data value modifications, without information restructuring or program code modification" (p. 482). The key to this definition is that the users of the system do not have to be trained information technology professionals to change the functionality of the software system. System functionality should be able to be changed by simply changing parameters stored in a lookup table or by manipulating some graphical objects within a graphical user interface. Flexible systems are user enhanceable.

1.2 Significance

The production of flexible systems is not a recent development. Blum (1993) discusses a system, implemented in 1980, that was developed using flexible software techniques.

Hall & Ligezinski describe how the concept of flexible software has been implemented effectively by large European banking and insurance organisations using third

generation languages (3GLs) such as COBOL. Research is currently being undertaken into the applicability of software flexibility techniques in fourth generation languages (4GLs). Ligezinski is primarily involved in the development of flexible systems for the financial sector in Vienna using 3GL languages, although he has experimented with 4GL environments such as Borland's Delphi. Edith Cowan University is currently involved in a research project that involves the implementation of a flexible system using Oracle's 4GL suite of products (Hall & Ligezinski, 1997a; Hall & Ligezinski, 1997b).

A new major area of software application development has recently emerged – Web-enabled applications. These are applications that can accept data entry from a World Wide Web site and/or report information back to a user via a World Wide Web site. A United States company, Network Wizards, published a survey in 1998 that showed the dramatic rise in the use of the World Wide Web. The survey calculated the number of domains, or sites, that were registered as being on the World Wide Web. The study showed that the number of sites almost doubled in the twelve months between July 1997 and July 1998 (Network Wizards, 1998).

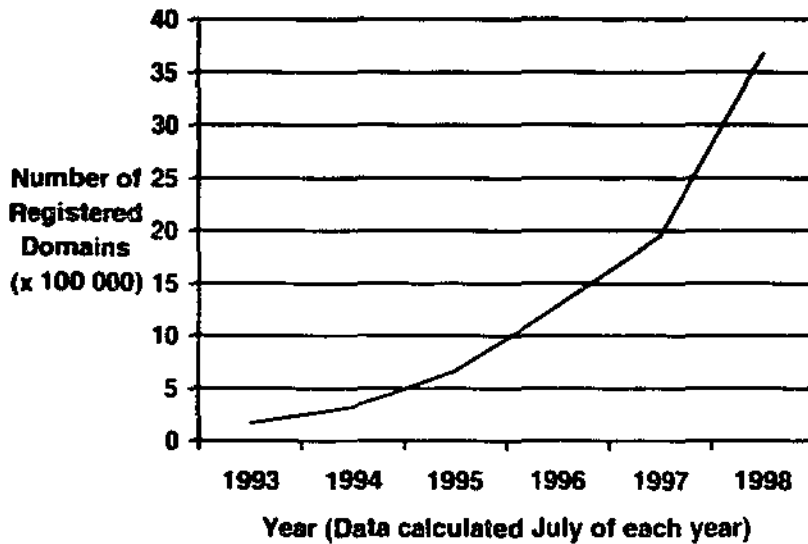


Figure 1.1 Internet Domain Servers by Year (Network Wizards, 1998).

In April 1998, the Australian Government released a report 'Electronic Commerce in Australia' (Department of Industry, Science and Tourism, 1998) which illustrates the incredible rate of growth of business on the Internet. The figures below relate to Australian business:

- In February 1998, 1.6 trillion dollars worth of transactions took place.
- 11% of all businesses have a Web presence.
- The volume of on-line purchasing by Australians has tripled in the last 18 months.
- By the end of 1998, all of the top 1000 Australian companies are expected to have a Web presence.

These statistics indicate that the World Wide Web has become recognised as a legitimate business tool within Australia.

Web-enabled applications will face the same maintenance problems as 3GL and 4GL applications have faced in the past. There is a need to determine whether the benefits that flexible software has provided in 3GL applications can also be achieved for Web-enabled applications.

The concept of an application that can be accessed from all over the world provides challenges in the area of access security. User access to an application may have to be controlled. What a particular user can see may depend on many factors, such as

- Their identity (i.e. login accounts and passwords).
- The geographical location that they are attempting to access the application from (i.e. their IP address).
- The current time and date. For example, a user may only have access to the application between 9am and 5pm, Monday to Friday.

As needs and policies change, an access controlled Web-deployed system should be flexible enough so that the access settings for any particular user, or a group of users, can be altered to reflect these changes without the intervention of a trained programmer, software engineer, or system administrator. As the person or persons in charge of administering the system may have roles that are unrelated to Web administration, it is highly possible that they do not work in the same office, city, or even country that the system is based in. Therefore it is important that such administration can be performed from a remote site.

A key reason why software flexibility is so applicable to the World Wide Web is the diversity and size of the audience. Systems can no longer be aimed at one particular

culture or one particular way of doing business. There is now almost global access to the Internet, and if the system does not cater for a certain culture then it may be excluding members of that culture from its user base. From a business point of view this can reduce the market base for products or services and result in a potential loss in revenue. One advantage of implementing software flexibility into a Web application is the ability to display Web pages in different languages, or with different culturally biased content, dependent on the identity of the user or the geographical location that they have accessed the systems from. Systems must be easily enhanceable so that different classes of users see different things.

The World Wide Web is an ideal medium for university lecturers to distribute information to a large number of students. Web-enabled unit materials can be accessed anytime from almost anywhere in the world making it especially attractive to students who study externally. Information is a valuable resource, and a university may not wish to have their information accessed by people who do not pay for the privilege. The ability for lecturers to define who may access their Web-enabled information, along with the ability to tailor this information for different groups of students should prove beneficial for universities.

1.3 Purpose

The purpose of this thesis is to determine whether the existing concepts of flexible software can be successfully applied to the realm of World Wide Web applications. This thesis also aims to determine whether the use of these techniques result in any benefit for users of the system in terms of administration and maintenance.

1.4 Research Questions

Research Question 1: Can the existing concepts of flexible software that exist in third generation language development be ported into the area of World Wide Web applications?

Research Question 2: Would the use of the concepts of flexible software in the development of Web sites benefit their users and administrators?

1.5 Organisation of the Thesis

Chapter 2 of this thesis reviews the literature that was found in relation to this research, ranging from different approaches to reducing the cost of software maintenance, through to literature about some of the development tools that can be used in a World Wide Web environment.

Chapter 3 outlines the method that was used to conduct this research. Chapter 4 provides the basic system design of a Web-enabled software application that was developed as part of this research. Chapter 4 also provides a discussion on many of the implementation issues that arose during the development.

The results gathered in response to the research questions for this thesis are stated in Chapter 5. Chapter 6 is a discussion of what these results may mean. Finally, Chapter 7 details the findings of this research and suggests future research ideas.

A section providing definitions of terms that are used in this thesis, a bibliography and several appendices provide background information that may assist in a greater understanding of this research.

Chapter 2: Review of Literature

2.1 General Literature

2.1.1 The Cost of Software Maintenance

Software maintenance accounts for a very significant portion of any typical software budget. Pressman (1992, p. 667) states that there has been a steady increase in the relative cost of maintenance from approximately thirty five percent of a typical software development budget in the 1970's to approximately sixty percent during the 1980's. He predicts the possibility of maintenance accounting for eighty percent of the software budget by the mid 1990's. Pressman (1992, p. 667) cites McCracken's (1980) description of a 'maintenance bound organisation' – an organisation that has to allocate so many resources to maintenance that they simply do not have the time to pursue any new development. Pressman's estimates are now six years old and the figure of eighty percent of the software budget may or may not be realistic. Hall & Ligezinski (1997a) state that the maintenance of a software system can cost as high as sixty or even seventy percent of the total software development budget for a given system. While the exact figure is unknown, it is generally agreed that "software maintenance accounts for more effort than any other software engineering activity ... the most costly phase of the software life cycle." (Pressman, 1992, p. 665).

Swanson (cited in Pressman, 1992, p. 664) identifies three categories of software maintenance:

1. Corrective maintenance – the removal of errors, or 'bugs' in an application

2. Adaptive maintenance – the modification of software to interface with a changing environment
3. Perfective maintenance – ongoing enhancements which are generally requested by users to improve the functionality of an application.

Booch disputes that adaptive and perfective maintenance come under the umbrella of software maintenance. “It is maintenance when we correct errors; it is evolution when we respond to changing requirements; it is preservation when we continue to use extraordinary means to keep an ancient and decaying piece of software in operation” (Booch, 1991, p. 5).

The difference in terminology is irrelevant. What is relevant is that these authors agree that there is more to the post implementation phase of a software system than fixing bugs. For the purpose of this thesis, Swanson's terminology will be used.

Figure 2.1 shows the portion of the total maintenance effort that each category of maintenance accounts for, as stated by Lientz & Swanson (1980) cited by Hall & Ligezinski (1997a).

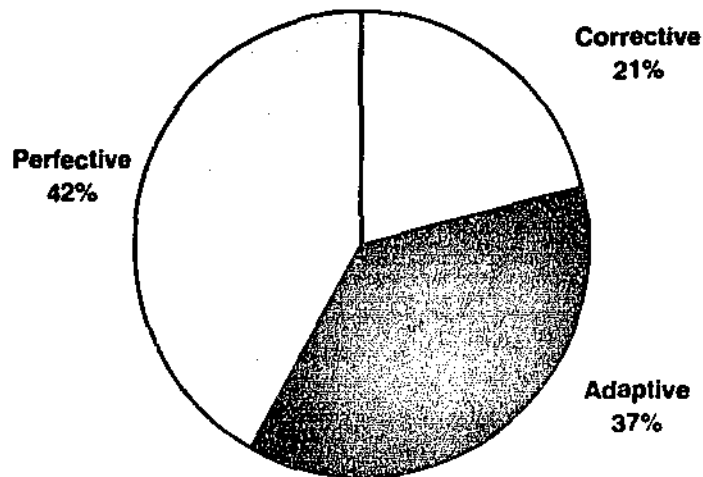


Figure 2.1 Categories of Maintenance (Lientz & Swanson (1980) cited by Hall & Ligezinski (1997a)).

Corrective maintenance is a fact of life for software developers. Design or implementation errors will always exist in all but the most trivial of applications and they must generally be corrected otherwise they threaten the applications integrity (Chapell, 1996).

The cost and impact of adaptive and perfective maintenance can be reduced if a system is designed with the knowledge that this type of maintenance will be required. As Parnas (1979, p. 129) states, the problems that software engineers face in terms of adaptive and perfective maintenance are no different to the issues that engineers of other domains are facing. Engineers of all domains must design their products in such a way that a change can be made to them with minimal impact on the system as a whole.

The major technique of designing for change that Parnas describes is that of breaking down an application into functional components. This should be performed in such a

way that each of these components is sufficiently self-contained so that altering the internal functionality of one component will not affect the rest of the system as a whole. Parnas advocates designing for 'program families' rather than specific applications. Components should be written as generalised as possible so that any application that requires the functionality expressed by a particular component is able to make use of that component (Parnas, 1979, p. 130).

This emphasis on code re-use and generalised components is very heavily mirrored in the object-oriented methodology as described by Booch (1991). This technique is particularly suited to combating the issue of adaptive maintenance. A component-based system should be designed so that if a factor external to the software system changes only one component of the system is affected. This component can then be modified and replaced without the need to alter the remainder of the system.

Perfective maintenance usually occurs as a result of a change in the requirements of the system. As users start interacting with a software system they may identify enhancements to the system that would improve the systems functionality. The delivered system may well satisfy the requirements of the users at the time of delivery but these requirements often change over time.

2.1.2 Static Software in a Dynamic World

"There is no magic technique for handling unanticipated changes" (Parnas, 1979, p. 129). It is impossible to foresee everything that can happen in the future, so software engineers must design for change, even though they may be unaware of what those changes will be.

Johnson, Ligezinski, & Woolfolk (1996) state, "many, if not most, of the real world requirements lie unknowable in the future at the time the system is designed" (p. 482). In some cases it is likely that the requirements of a system will change while the system is actually undergoing development. Hall & Ligezinski point out that changes in the requirements of a system are not predictable. "The assumption that if the analysis describes ALL variables, the requirements are complete, is false." (Hall & Ligezinski, 1997a, p. 2).

Blum identifies three categories of software system requirements:

1. Closed requirements - those requirements that are well defined at the time of development and are unlikely to change in the foreseeable future.
2. Abstract requirements - concepts that are not concrete, but rather just necessary products of the development. Primary examples of abstract requirements are 'security' and 'user friendliness'.
3. Open requirements - requirements that are poorly understood or are dynamic. These are requirements that are either not completely defined at the time of development, or those requirements that are likely to change either during development or after the product has been implemented. It is open requirements that are the basis for the majority of perfective maintenance.

"Few applications fall into just one category, and most complex products contain components with closed, abstract, and open requirements." (Blum, 1993, p. 725).

It is the challenges raised by open requirements that flexible software aims to manage.

2.1.3 Flexible Systems

Johnson, Ligezinski & Woolfolk (1996) define the categories into which all software systems fall into according to their level of flexibility:

1. Weak flexibility – if both the information structure and procedural (or source) code must be changed to modify the system.
2. Medium flexibility – if the information structure remains static, but procedural code must be altered to modify the system.
3. Strong flexibility – if the system may be modified by changing only some data values.

For the purpose of this thesis, the term 'flexible software' will refer to software systems that exhibit strong flexibility.

Johnson, Ligezinski & Woolfolk (1996) discuss the problems of static software systems in a dynamic environment and provide many examples where a change to the real world environment has resulted in considerable programming work in order for the affected software systems to adapt. These authors favour designing for flexibility and state that they have had "favorable and ongoing experience with techniques that ... allow automated [software] systems to flex with their real world counterparts" (p. 482).

Hall & Ligezinski (1997a) consider the maintenance of software systems as one of the largest areas of expenditure of both time and money and therefore an area that software developers should be targeting in an effort to increase the speed and decrease the cost of software development. The authors state that the methods that they have used to achieve software flexibility have been proven in practice in large computer systems in

the European banking sector. McNurlin & Sprague (1993, p. 315) cites the existence of a flexible software system in Switzerland. The system handles the trading of commodities such as oil, gas, currencies and precious metals. The authors state that this system had not had any programmer maintenance between 1982 and the publication of their book in 1993.

One surprising aspect of Hall & Ligezinski's findings is that the perception that it requires extra developmental effort to develop flexible software did not appear to eventuate in reality. The authors surmised that the reason that costs did not increase is that fewer applications had to be developed. Applications were written with such flexibility that an individual system could adapt to various situations, which in the past would have been addressed with two or more separate applications. (Hall & Ligezinski, 1997a).

Not all information technology professionals see flexible software as a feasible venture. David Ensor, an International Oracle consultant, states that developing systems that can adapt to changing requirements is fine in theory. He does not believe, however, that this is feasible, or even possible, given the development tools that are available today. He sums up his feelings on the matter by stating that "it is absurdly difficult to design a piece of code that will completely meet a totally unknown requirement" (Ensor & Stevenson, 1997, p. 511).

2.1.4 Methods of Providing Software Flexibility

There are a number of theories on how software flexibility should be implemented. Authors such as Mehandjiev & Bottaci (1996) believe that the concept of 'Visual

Programming' should be employed. They suggest that if an application can be written in a language that is purely graphical rather than text based then there is a huge potential for users to modify the structure of their program as they wish. This is based on the concept that humans can understand graphical interfaces much quicker than they understand text-based commands. The authors do concede that at this stage no one has produced a purely visual language of any note that is applicable to the general domain.

Blum (1993, p. 733) describes a framework for specifying open requirements by breaking down these requirements into smaller fragments of information. This allows the designer to specify exactly what is known about the requirements even if those requirements are not complete. These fragments can be made up of smaller fragments and so on. Blum compares this to the human memory that may allow a person to remember certain events and these events may combine to give a memory of a particular situation. As more events are remembered, the memory becomes more complete. Similarly as more fragments are added, larger chunks of the systems requirements unfold.

Blum (1993, p. 728) discusses a holistic computer system, TEDIUM, that has been developed to allow software engineers to design in this way. The TEDIUM application can be modified by the addition or removal of specification fragments. Unfortunately this type of development appears to be confined to the specialised TEDIUM system for the time being and is not accessible by the general software development community.

The approach to software flexibility that appears to be most widely used in the mainstream development community is that of 'parameterised' or 'data driven' systems.

This method involves the system being 'driven' by data values external to the system – usually stored in some sort of database. Johnson, Ligezinski and Woolfolk (1996) describe a flexible sales commission system - a system to calculate how much a salesperson should be paid for the sale of each type of product. The requirements of this system are that commission rates and discounts may change, as well as the formula by which the commission is calculated. Commission rates will also be determined by a number of rules. The authors describe a design that consists of the following:

- A database table (FILE1) that contains the commission rates and discounts for various products.
- A database table (FILE2) that contains the search key that should be applied to FILE1 in different situations. This controls the order that records in FILE1 are accessed, therefore allowing the commission to be calculated by a number of rules – each of which may have a higher precedence for a particular situation.
- A callable algorithm (PROG) which calculates the commission. By having this algorithm external to the system the user has complete control over how the figures are calculated without altering the source code of the system.

Hall and Ligezinski (1997b) describe a similar design method that was used to develop a research tool for the Health Department of Western Australia. Values and codes that have the possibility of changing or being added to are not hard coded in program logic. Instead, such codes are defined in database tables that can be altered by the user and then used immediately in the application.

It is the data-driven approach to software flexibility that will be the basis of the system developed to demonstrate software flexibility in this study.

2.1.5 The Importance of the World Wide Web

As the World Wide Web is a fast growing technology, most of the current literature referring to it is sourced from the Web itself. Among recent documents regarding the growth of the World Wide Web is a document released by the Commerce Department of the United States Government. (United States Commerce Department, 1998). It stresses that the World Wide Web is having, and will continue to have, a large effect on the United States and global economies. There are also a number of Internet Survey groups such as the Network Wizards who periodically publish figures that indicate the growth in Web sites and in the number of people using the Internet. This type of Web available information has a limited period of currency. Frequent investigation of articles published on the Internet is required to monitor the current situation.

As anyone can publish on the Internet, the content of these materials may not necessarily have been subject to peer review. Caution must be used when using information from such sources.

2.1.6 Flexibility and the World Wide Web

Hall and Ligezinski (1997b) discuss four issues of software flexibility that they encountered during their investigation of providing software flexibility using 4GLs. These issues appear equally relevant, if not more so, in the World Wide Web environment:

1. Access security – the information that is available from a Web site may depend on certain factors, such as the user's identity, their geographic location, and the time of day. Users should only see what they are permitted to use and should

not be aware of facilities that they do not have access to. These access rights may change on a regular basis.

2. Dynamic report formulation – in terms of Web sites, it is rare to actually create a physical report but the concept of dynamic report formulation translates directly to the appearance of the Web page. Not all users of a Web application need or wish to see the same information. A Web page should be able to have a different appearance based on the factors discussed in 'access security'. The appearance of the Web-deployed page may be different for different users.
3. Data entry processes – the ability to have dynamic data entry screens is very relevant to Web applications. Not all users should be required to enter the same information. Users should not even be prompted for information that is not relevant to them. Once again, the data that is relevant to certain users may change quite regularly. The language used for text labels should also be able to be changed depending on the native language spoken by the user.
4. Changing business rules – the problem of changing business rules is very similar in any software system, whether it is 3GL, 4GL or Web based. If the structure or values of information that a program uses change, then the program must be able to adapt. A common example is that of income tax rates. Every time a government changes these rates, payroll systems have to change to accommodate this.

Each of these four issues can be implemented without using flexible software techniques, but every time that a users profile or some environmental factor such as tax rates change, a specialist programmer would have to be called in to maintain the system.

These issues demonstrate clearly how relevant this area of research is to the World Wide Web.

Access security is an important issue on the World Wide Web. The World Wide Web is simply a network that can potentially be accessed by anyone with access to a computer, telephone line and an Internet service provider. Information that is not secured in some way can be accessed from any computer that can be connected to the Internet.

As a wide range of people from different cultures can potentially access a Web-enabled application, it is very likely that there will be differing requirements in terms of data entry and data reporting. Information may have a different relevance to different people in different cultures.

Finally, business rules and the real-world environment are continuously changing affecting all kinds of systems whether they are Web-enabled or not. However Web-enabled applications are still an immature concept and tend to undergo these sorts of changes more frequently than other environments as organisations strive to find the most effective way to harness the potential of the World Wide Web.

2.1.7 Implementation Tools

Many different methods of Web enabling software applications are currently in use.

Authors such as Michael Edwards (Edwards, 1997) and Nancy Cluts (Cluts, 1998), both employed by Microsoft Corporation, describe different techniques and technologies for implementing Web pages and getting input from users. Although both of these authors

have a declared bias towards their company of employment, they do discuss a number of new advances in the area of HTML and Java programming such as Dynamic HTML. Vanhelsuwe et al. (1996) provide a comprehensive reference of the Java (version 1.1) language. Although Lea's book (Lea, 1997) is directed at the issue of concurrency within Java, it still appears to be a useful language reference. Harmon (1996) describes Microsoft's implementation of Java, Visual J++, as well as the ActiveX technologies. This bias towards Microsoft reduces the appeal of this source as a reference but still poses many of the issues faced within the area of Web programming.

The implementation phase of this study will develop a Web-enabled system using an Oracle8 database and Developer 2000, one of Oracle's Web-enabled 'front end' development tools. Oracle also provides the flexibility to use third party Web development products such as Java and variations of the HTML technologies. There are a number of White Papers available from Oracle Corporation describing the use of Oracle8 and the associated Web Server applications. (Oracle Corporation, 1997a; (Oracle Corporation, 1997b). These papers document the capabilities of the various Oracle products that may be utilised in the development of Web-deployed systems.

The Web Application Server Handbook (Dynamic Information Systems LLC, 1998) provides in-depth information about Oracle Web Application Server version 3.0. The book details the architecture of a Web Application Server based system and specifies issues that must be addressed when developing a Web application as opposed to a standard client/server application. Issues such as network bandwidth and security apply to both environments, but they take on a new perspective in the context of a globally accessible application. This reference also provides comprehensive development

information in the major areas that the Web Application Server can be used, such as Java, HTML, and PL/SQL development.

2.2 Previous Findings

No research has been identified in the specific area of software flexibility in the realm of the World Wide Web. However, as discussed in the previous section, software flexibility has been implemented in real-world situations for the past two decades and authors such as Johnson, Ligezinski and Woolfolk (1996) state that such systems have reduced maintenance costs.

2.3 Related Studies

2.3.1 Flexibility in 4GLs

The majority of work and literature in terms of software flexibility has involved third generation environments. Hall and Ligezinski (1997b) recognise the potential benefit of porting these proven techniques into other domains. These authors specifically address the development of flexible software using 4GL products such as the Oracle Developer 2000 suite of products and Borland's Delphi programming environment. Although their studies are in the early stages they do re-iterate a number of benefits of employing flexible software techniques. They state that "[the] modern database environment with its supporting tools has many inherent facilities that can be harnessed towards the goal of flexible software" (Hall & Ligezinski, 1997b, p. 7). Most importantly, their research appears to show that lessons that have been learnt in one area of software development can subsequently be applied to other areas.

Stephen O'Connor, an Honours student at Edith Cowan University, is currently researching ways of implementing flexibility techniques using the Oracle environment (S. O'Connor, personal communication, September 10, 1998). This research is in progress and has not yet been published. It was described in his thesis proposal submitted to the Faculty of Science and Technology in 1997.

2.3.2 Metadata on the Web

The concept of metadata is tightly entwined with many Web applications. Hillmann (1998) defines metadata as "data about other data" (par. 1). Metadata is a set of attributes or properties that describe the data in question, such as the name of the person who collected the data, the date that the data was created or last modified, or which predefined category the data can be placed in. Hillmann states that although the term metadata has only recently come into vogue, it has been with us for some time. Librarians, in particular, have traditionally recorded information such as the author's name, date of publication, and key subject information for books and journals.

Metadata has a number of uses in the realm of the World Wide Web. The Web can be viewed as simply a large document database. Metadata plays a large role in allowing users to retrieve documents that contain information relevant to their needs. The inclusion of the '<META>' tag in the standard Hypertext Markup Language (HTML) specification reflects this. This tag allows Web authors to define a number of keywords that describe the document. Search engines then match these keywords with their search parameters to locate documents that are relevant to the user's needs. (Powers, 1998, p. 48).

Metadata can also be used to determine whether a person can view a particular Web document. There are already technologies that allow the filtering of Web content to meet a user's requirements. A technology known as the Platform for Internet Content Selection (PICS) allows Web documents to be screened via a rating system similar to the rating system that applies to the film industry (Miller & Resnick, 1996). A user may set up their Web browser so documents that are rated as violent or sexually explicit cannot be viewed.

The PICS system relies on a third party Web site that contains ratings (metadata) for a PICS assessed Web document. When a user attempts to load a PICS assessed Web document, the rating is retrieved from this third party Web site and compared with the settings in the user's Web browser. (Iannella & Ward, 1998).

A major characteristic of PICS is that access is completely controllable from the user's Web browser. So while PICS is ideal for parents who wish to restrict their children's access to certain Web documents, it provides no protection for owners of the documents. They have no influence over what a user's Web browser will allow them to see. (Iannella & Ward, 1998).

The proposed Dublin Core metadata standard aims to standardise the way documents are described to simplify the locating of documents on the World Wide Web that contain a certain characteristic. Different types of computer users have different needs in the area of Web document storage and retrieval and "users meet such needs today with a wide variety of metadata standards." (Hillmann, 1998, par. 1).

Hillmann (1998) identifies the following 15 elements of Dublin Core. This are listed in Table 2.1.

Table 2.1

Dublin Core Metadata Elements.

| Dublin Core Element | Description |
|----------------------------|--|
| Title | The name given to the document by the creator or publisher. |
| Creator | The person or organisation primarily responsible for creating the intellectual content of the document. |
| Subject | Keywords or phrases that describe the content of the document. |
| Description | A textual description of the document such as an abstract. |
| Publisher | The person or organisation responsible for the current presentation of the document. |
| Contributor | A person or organisation other than the 'Creator' that has made a significant intellectual contribution to the document. |
| Date | The date that the document was created or made available in its current format. |
| Type | A generalisation of what type of document we are describing. e.g. a novel, home page, essay, report, etc. There are some enumerated values for this element that are currently being discussed, but there is no standard at this time. |
| Format | The data format of the document. e.g. Postscript file, Microsoft Word document, JPEG image, etc. |

| | |
|------------|---|
| Identifier | A string or number used to uniquely identify the document, similar to International Standard Book Numbers (ISBN). |
| Source | Information about a document on which the current document is based. |
| Language | The language of the intellectual content of the document. e.g. English, French, etc. |
| Relation | The identifier of another document that is related to the current document in some way. |
| Coverage | Either the physical area to which the document applies (e.g. the document may be about Eastern Europe) or the era to which the document applies (e.g. the Shakespearian era). |
| Rights | A textual statement about the copyright details of the document. |

Note. From Hillman (1998).

Any of these elements can be repeated for a document. For example, a document will usually have more than one keyword that describes its content. In this case the document would contain multiple Subject elements.

The Resource Description Framework (RDF) takes a different approach to standardising the use of metadata. Instead of specifying a set of standard elements for everyone to follow, RDF specifies how a group of users should specify the elements that they require. “[RDF’s infrastructure] enables metadata interoperability through the design of mechanisms that support common conventions of semantics, syntax, and structure.”

(Miller, 1998, p. 1). RDF uses the features of the eXtensible Markup Language (XML) standard to specify an unambiguous way of specifying metadata elements.

The premise behind RDF is that different groups of computer users may require totally different sets of metadata. Dublin Core may be ideal for the searching and retrieving of text documents but may not suit other types of World Wide Web users. The aim of RDF is to impose a structured way of specifying metadata elements so that if two different users wish to convey the same information then there is a high probability that the structure of the two sets of metadata will be similar. RDF aims to achieve this without locking users into a rigid set of allowable metadata elements. RDF is being supported by the World Wide Web Consortium, but has not yet been ratified as a standard.

Hillmann (1998) states that the metadata for a document can either be stored within the document itself, or externally to the document. The HTML standard allows metadata to be stored within the document itself. Some other document formats do not have this ability and must therefore have their metadata stored in an external data source. This research involves storing metadata about a unit material in an external data source and using that information to determine whether a user has access to that material.

2.4 Research Methodology

The following two research questions were specified for this research:

1. Can the existing concepts of flexible software that exist in third generation language development be ported into the area of World Wide Web applications?
2. Would the use of the concepts of flexible software in the development of Web sites benefit their users and administrators?

Research Question 1 was addressed by designing and implementing a software application that incorporated techniques of software flexibility.

Research Question 2 was addressed by qualitative means. Malim and Birch (1997) describe a number of ways of gathering qualitative information from a test group of people, such as questionnaires and surveys. They also identify a number of ways to standardise the qualitative results that are gathered to ensure that findings are as meaningful as possible. However, this is not relevant due to the small sample size used for this study.

Drew, Hardman & Hart (1996) discuss the benefits and pitfalls of qualitative research. They also discuss the need to gain consent from any participants in the study that are to be identified in the thesis.

2.5 Summary

The issue of changing requirements has always been an integral part of software development. The ideas of Parnas (1979), Booch (1991), and other exponents of component-based software are often promoted as the blueprint of how to reduce the cost of maintenance. Other information technology practitioners favour an alternative approach and believe that software should be designed in such a way that functionality can be changed without altering any of the system's source code (Johnson, Ligezinski and Woolfolk, 1996).

The techniques of flexible software have been shown to be successful in 3GL environments over a period of at least two decades. Researchers are now beginning to investigate whether such techniques are applicable to 4GLs. It is a logical extension to take these techniques and apply them to the area of software development that is experiencing enormous growth at this time – the World Wide Web.

Chapter 3: Method

3.1 Introduction

The following two research questions were stated in Chapter 1:

1. Can the existing concepts of flexible software that exist in third generation development be ported into the area of World Wide Web applications?
2. Would the use of the concepts of flexible software in the development of Web sites benefit their users and administrators?

This chapter describes the methodology, resources and limitations that were related to this study.

3.2 Design

As there was no available data on efforts to implement the concepts of flexible software in a Web environment, Research Question 1 was addressed by the implementation of a system demonstrating these concepts. The management of academic, Web-enabled unit materials was the chosen application domain for two reasons:

1. There were a number of easily available subjects (Edith Cowan University staff) who were familiar with this domain and related issues;
2. It was known that there were a number of staff members within Edith Cowan University who were particularly interested in the management of Web-enabled unit materials.

Research Question 2 was addressed using qualitative methods.

A sample population was invited to view the Unit Material Management System and comment on the concepts of software flexibility that it demonstrated. Following the demonstration, these subjects were asked to complete a questionnaire about the applicability of the concepts that had been demonstrated. The findings for Research Question 2 were derived from responses collected during the demonstration and from the questionnaire.

3.3 Product

A major element of the research was the development of a software application that would allow unit coordinators to restrict access to their Web-enabled materials. The system was known as the Unit Material Management System.

The Unit Material Management System was developed as a vehicle to exhibit various concepts of software flexibility on the World Wide Web. It does not exhibit the full functionality and security that would be required for a production system.

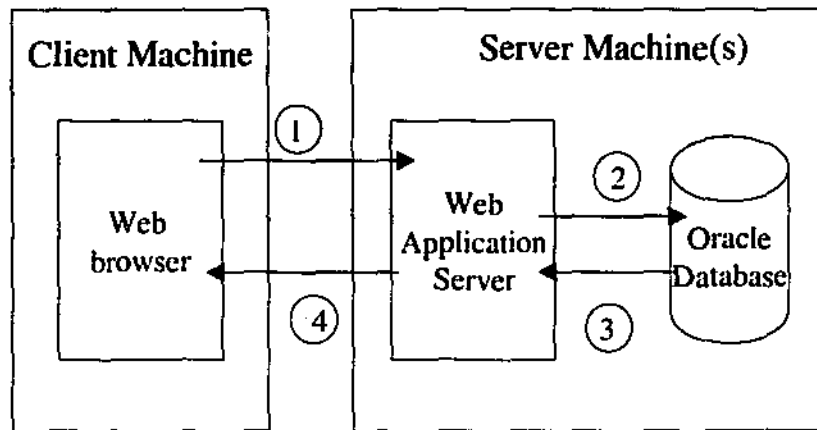
The development of the Unit Material Management System was an evolutionary process and many of the issues that arose during implementation were not foreseen during the initial design. Details of this application and issues that arose during its development are discussed in Chapter 4.

3.4 Environment

3.4.1 Oracle Technology

The Oracle suite of development tools was chosen as the platform on which to develop the Unit Material Management System. Oracle's recent development efforts appear to be directed at the Internet domain. Oracle Corporation markets the latest release of their database management system, Oracle Enterprise Edition Release 8.0, as "the database for network computing" (Oracle Corporation, 1998a). Oracle8 is based on Oracle's standard Network Computing Architecture (NCA) and claims to be able to support tens of thousands of users – a must when a system can be accessed from any World Wide Web-enabled computer. Other features such as support for a number of Java based technologies appear to make Oracle a serious contender for mainstream World Wide Web application development. This, as well as the availability of Oracle products through Edith Cowan University, made it a sensible choice of development platform.

Two methods of deploying a database application on the World Wide Web were investigated. The first option was to develop applications primarily using Oracle's Web Application Server (WAS) which allowed the deployment of static HTML pages populated with information from an Oracle database.



1. Web browser requests HTML page (also sends any parameters)
- 2 & 3. WAS retrieves data required for the page
4. WAS generates a static HTML page to display and sends it to the Web browser.

Figure 3.1 Web Application Server (Dynamic Information Systems LLC, 1998, p. 30).

The second Oracle Web deployment method investigated was to use an extension of the existing Oracle Developer 2000 products in conjunction with Oracle Web Server (or a third party Web server). Recent versions of the Developer 2000 product, commonly known as 'Web Forms', have included three additional products:

- Oracle Forms Server
- Oracle Reports Server
- Oracle Graphics Server

These applications are designed to work with Oracle Forms, Oracle Reports and Oracle Graphics respectively.

Oracle Forms Server is installed on the machine that is to act as the Web Server. When the user accesses a Web site containing the Oracle Forms Server, a small run-time

component called the Forms Client Applet is downloaded to the user's browser. The Forms Server Runtime Engine (located on the Web server) converts the developed form into a Java applet and downloads it to the Web browser. The Forms Client Applet then coordinates the displaying of the form. Reports Server and Graphics Server work in similar fashions. It must be noted that this technology still relies on the presence of a Web server product. This can either be Oracle's Web Application Server or a third party product, such as Apache by IBM.

In the second half of 1998, further Web development products such as JDeveloper have been released by Oracle Corporation. The JDeveloper tool allows the deployment of Oracle databases using applications written in Java. (Oracle Corporation, 1998b). JDeveloper was not considered as an option for the Unit Material Management System because it had not been released when the tools were being investigated.

The key advantage of using the Developer 2000 Server approach is that the development process is similar to that of a standard Developer 2000 application. Applications that have been written using Developer 2000 in the past can be Web-enabled with minimal effort once Developer 2000 Server is installed and configured correctly. Also, people who have used standard Oracle Forms applications will be reasonably familiar with the way the Developer 2000 Server user interface works.

The main disadvantage of using Developer 2000 Server as opposed to the Web Application Server product is that Developer 2000 Server has greater limitations when it comes to some World Wide Web issues. This stems from the origins of both products. Oracle Corporation describes its Web Application Server product as "the

centrepiece of our Internet Computing strategy, providing our customers cost-savings, easier application deployment, and the opportunity to build and deploy applications on the Internet.” (Oracle Corporation, 1998a, p. 3). The Developer 2000 Server product was originally designed for a client/server environment over a local area network. The product has since evolved to so that it can operate in the World Wide Web environment, but it does not make use of all of the features available to most Web applications. The limitations of this product are discussed in Chapter 4.

The Unit Material Management System primarily uses Developer 2000 Server, but it does make use of some of the features of the Web Application Server. This is detailed in Chapter 4.

3.4.2 Development Environment

The development and presentation of the Unit Material Management System was performed on two IBM compatible personal computers connected to the World Wide Web via a Local Area Network (LAN).

The first machine acted as the World Wide Web server and hosted the Unit Material Management System. This machine had a Pentium II processor running at 233MHz, 64Mb of memory, and an 8 Gb hard disk drive. Microsoft Windows NT 4.0 (build 1381, Service Pack 3) was used as the operating system.

The second machine acted as a World Wide Web client, typical of the sort of machine that a user of the Unit Material Management System would use. This machine also had

a Pentium II processor running at 233MHz, 64Mb of memory, and an 8 Gb hard disk drive, but used Microsoft Windows 95 as its operating system.

The following products were installed on the Web server machine:

- Personal Oracle Database version 8.0
- Oracle Web Application Server Release 3.0.1
- Oracle Developer 2000 Release 2.1
- Oracle Developer 2000 Server Release 2.0
- Sun Microsystems Java Developers Kit (JDK) version 1.1
- Oracle Designer 2000 Release 1.3

The client machine originally had Netscape Navigator 4.05 installed as its Web browser. During the initial investigation of the Oracle Developer 2000 Server product, it was discovered that there was a Java compatibility problem between the Oracle products and Netscape Navigator 4.05. As a result, Netscape Navigator version 4.06 was installed as the client machines Web browser.

3.5 Subjects

There was not enough time or potential subjects to be able to assemble a sample group of sufficient size or diversity to perform a statistical analysis. Research Question 2 was addressed by inviting a number of Edith Cowan University staff members with some experience in either unit material management or Web administration to a presentation of the Unit Material Management System.

The subjects that were chosen all had a background in information technology. Ideally the sample population should have been considerably larger and consisting of users with a varying range of experience in the information technology field. The major drawback with using subjects with little or no information technology experience is that they may struggle to comprehend the issues that flexible software tries to address. Subjects with no experience in information technology may not be aware of the severity of the issue of software maintenance, or how hard it is to programmatically control access to World Wide Web sites. It was for this reason that experienced information technology professionals were selected.

This issue is a limitation of the research and will be discussed further in the 'Research Limitations' section of this chapter.

3.6 Procedure

After submission of the thesis proposal in May 1998, investigation of the Oracle Developer 2000 Server product began. The initial data and functional analysis for the Unit Material Management System was also performed. There were a number of issues relating to the installation and configuration of the Developer 2000 Server product. These issues are discussed in Chapter 4.

Once the development tools were installed and configured, the majority of the time was spent designing and implementing the Unit Material Management System. This involved addressing a number of issues that were not foreseen before the

implementation began. These issues and the actions taken to resolve them are discussed in Chapter 4.

During the second half of August a list of possible subjects was compiled. These potential subjects were sent a letter of invitation explaining the basic concepts of the research and inviting them to attend a demonstration of the Unit Material Management System either in late September or October. Nine of the eleven invitees responded to the invitation, all saying that they were interested in participating.

Demonstrations were eventually scheduled for Monday 26th October and Thursday 5th November. Four potential subjects attended the first demonstration and two attended the second. Unfortunately, two lecturers that had expressed interest in attending were not available at these times. The term 'subjects' refers to those who were actually able to attend a demonstration.

In the days following the demonstrations the people who were able to attend were asked to complete a questionnaire regarding their opinions of the applicability of the concepts of software flexibility that had been demonstrated. The results of the questionnaires were collated in an Oracle database table along with the names and job descriptions of the subjects.

3.7 Demonstration

Both demonstrations of the Unit Material Management System lasted approximately 90 minutes and were tape recorded with the permission of all present. The Subjects were

briefed on the concepts of software flexibility and the various issues that it has addressed in 3GL application systems. This was followed by a demonstration of the Unit Material Management System.

The structure of the demonstrations was informal. Subjects were encouraged to ask questions or make comments at any time during the presentation. Discussions about the applicability of the concepts of software flexibility followed each demonstration.

3.8 Questionnaire

Every subject was asked to complete a questionnaire once they had attended a demonstration. The questionnaire asked for each subject's name and job description within the University. They were also asked whether they gave their consent for their names and comments to appear in this thesis.

The questionnaire contained five questions relating to how the concepts of software flexibility, that had been outlined in the demonstration, applied to unit material management on the World Wide Web. An example of the questionnaire is available in Appendix D.

3.9 Data Analysis

No data analysis was required for Research Question 1. The implementation of the Unit Material Management System provided a 'proof of concept'.

The data used to address Research Question 2 was derived from both comments made during the demonstrations of the Unit Material Management System and the responses from the subsequent questionnaires. This data was tabulated and analysed qualitatively. Common threads and issues were derived and discussed.

The limited size and non-randomness of the sample population would have rendered any sort of statistical analysis useless.

3.10 Research Limitations

Research question one simply required proof that it was possible to implement concepts of software flexibility in a Web environment. There were some limitations in the tools that were used to develop the Unit Material Management System, but these did not stop the implementation of the concepts of software flexibility. These limitations are discussed in Chapter 4.

There were some limitations in addressing Research Question 2.

Resource constraints meant that the research was only able to address one application domain. It is possible that findings that are derived from the management of academic, Web-enabled unit materials may not be an accurate reflection of the applicability of flexible software techniques to the World Wide Web in general. Ideally a number of prominent application domains would have been tested, but resource constraints did not allow this.

There were also a number of limitations in the size and make up of the sample population. It was decided that all of the subjects that were invited would have a strong background in the area of information technology. Responses would not be as meaningful if subjects were not aware of the problems that software flexibility aims to counter. If the subjects were not aware of the difficulties in controlling access to a World Wide Web site programmatically, or the continuing problem of the software maintenance backlog, then they may not have been able to make a valid judgement of the benefits of flexible software. Ideally, the sample population would have consisted of subjects with varying degrees of information technology experience.

A sample population of six subjects is very small as a representative group. This limitation was again due to time constraints and the availability of possible subjects. It was decided to focus on a small select group of people that would hopefully have an interest in the area of flexible unit material management and offer valuable comment. The drawback of having such a small sample group is that it is impossible to reliably detect any statistical trends in their responses. This was one of the major reasons that qualitative methods were employed.

The concepts of software flexibility were implemented and tested in one domain – the management of Web-enabled unit materials in academia. This resulted in responses from the sample population being strongly biased towards issues relating directly to this domain. Although it was stressed to the subjects that the Unit Material Management System was merely a vehicle to demonstrate software flexibility, many of the responses were related to functionality or policy issues that would need to be addressed if such a system was developed for mainstream use. This is an issue that was foreseen, however

it was decided that subjects would be able to comment more freely if the system addressed a domain that they were familiar with.

Chapter 4: The Unit Material Management System

4.1 Introduction

The Unit Material Management System was chosen as a suitable Web-enabled application to demonstrate that the concepts of software flexibility were applicable in a World Wide Web environment. This chapter discusses the design and implementation issues of the system that allow it to exhibit flexibility. A specification of the system is available in Appendix B.

The system was never intended to be any more than a vehicle to assist in this research. The requirements and analysis of the Unit Material Management System were conceived in discussions between my supervisor and myself. Therefore, the Unit Material Management System may not accurately reflect the policies or the functionality that would be required if such a system were to be developed for general use.

It must also be noted that the scope of the Unit Material Management System is restricted to the management of access to Web-enabled materials. There are a number of security issues that would need to have been addressed if such a system were to be developed for general use.

4.2 System Design

4.2.1 Requirements

The Unit Material Management System must control access to Web-enabled unit materials that are assigned to a unit. Access to each Web-enabled unit material for each unit may be restricted by the following factors:

- **Who** - The identity of the user. A user must be enrolled in an academic unit to access materials assigned to that unit.
- **Where** - The IP address that the user is attempting to access the materials from.
- **When** - The current time and date. This may consist of either:
 1. a periodic time window. For example, 9am to 1pm every Monday and Wednesday.
 2. a fixed time period. For example, between 1st of March 1998 and the 31st of July 1998.

These access conditions must be able to be controlled by the coordinator of each unit through a graphical, Web-enabled interface. It is essential that the system is user friendly enough to be used by unit coordinators with minimal information technology experience.

4.2.2 Data Model

Figure 4.1 shows the underlying data model of the Unit Material Management System.

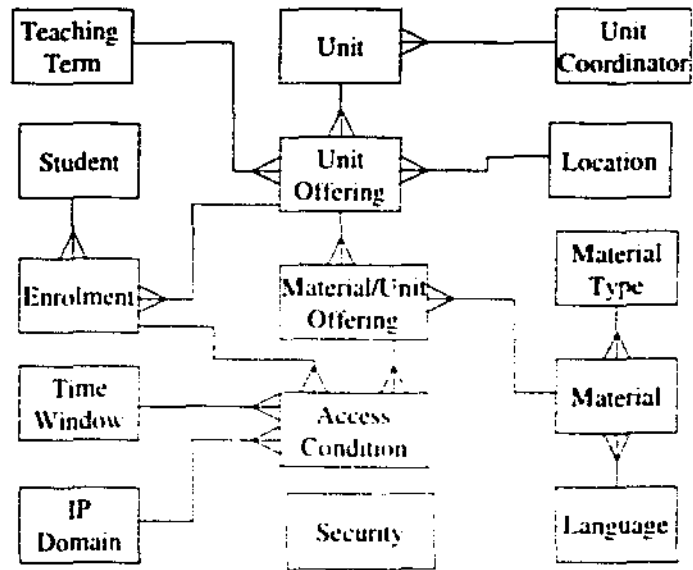


Figure 4.1 Data Model for the Unit Material Management System.

As the Unit Material Management System was developed in isolation from other systems there are a number of tables that store information that would normally be retrieved from other systems. This applies to the following tables:

- *Unit* – stores information such as the unit code, unit title, and the current unit coordinator.
- *Unit Offering* – stores information about a particular offering of a unit. A unit offering can be thought of as a single occurrence of that unit. For example, Unit CSP1143 may be offered at the Joondalup campus in semester 2, 1998. This is a different offering to CSP1143 offered at Mt Lawley in semester 2, 1998.
- *Student* – stores information such as the student’s identification number, name and address.
- *Enrolment* – stores which students are enrolled in a particular unit offering.

- *Location* – stores information about a teaching location. This can be a physical location such as ‘Mt Lawley Campus’ or a virtual location such as ‘External Studies’.
- *Unit Coordinator* – stores the unit coordinators login name as well as their full name and items such as their phone extension.
- *Teaching Term* – stores information about a period over which a unit is offered. This may be a period such as ‘Semester 2, 1998’, ‘Summer School 1999’, or ‘1st-14th of June 1999’.

There are two tables which stores lookup values for the Dublin Core metadata elements ‘Type’ and ‘Language’. These tables are used to populate pop-lists within the Unit Material Management System:

- The *Material Type* table stores lookup values for the ‘Type’ metadata element.
- The *Language* tables stores lookup values for the ‘Language’ metadata element.

The remaining tables make up the core of the material management functionality:

- The *Material* table stores information about each Web-enabled material that can be used within the system. This includes information such as the materials title, version number and the Web address of the document. The *Material* table also includes the 15 Dublin Core Metadata fields described in Chapter 2. This would allow an extension to the Unit Material Management System allowing documents to be searched and classified via these standard elements.
- The *Material/Unit Offering* table defines which materials apply to which unit offering. This allows different versions of materials to be assigned to different

offerings of the same material. This is essential in situations where different campuses are running different versions of a software package. This may require an adjustment to workshop or assignment materials. There may also be cultural considerations for units offered outside Australia.

- The *IP Domain* table represents valid IP addresses that a user may be permitted to access a material from. IP addresses are made up of four elements with each element being a number between 1 and 999 inclusive. If a full, four-element address is specified then access is restricted to a single machine. This would rarely be required or practical. Alternatively the unit coordinator can specify just the first part of the address. Typically, the first three elements of an IP address represent a particular network or domain. For example, the IP domain '139.13.169' may represent the student network at the Mt. Lawley campus of Edith Cowan University while '139.13.169.164' would represent a specific computer. Unit coordinators can enter the IP address to whatever granularity they wish, but three elements would seem to be the most practical. There are some issues with the use of IP addresses that will be discussed later in this chapter.
- The *Time Window* table represents a periodic time window for which a user may be able to view a material. A time window is defined by the following four fields:
 1. Label – a descriptive label that the unit coordinator can use to identify the period. For example, 'Monday 9am to 11am' or 'Weekends'.

2. Day – this can be one of nine values, a day of the week (Sunday to Saturday), 'Weekends' or 'Weekdays'.
 3. Start Hour – the hour of the day that defines the start of the window (1 – 24)
 4. End Hour – the hour of the day that defines the end of the window (1 – 24)
- The *Material Access Condition* table manages the restrictions on each material, per unit offering, per enrolled student. The table contains five fields that are used to restrict access to materials:
 1. Time Window – foreign key representing a link to a record in the Time Window table.
 2. Start Date – defines the start of a period between two calendar dates that a material can be viewed. This can be used to restrict access to the current semester.
 3. End Date – defines the end of a period between two calendar dates that a material can be viewed. This can be used to restrict access to the current semester.
 4. IP Address – foreign key to a record in the IP Domain table.
 5. Enrolment ID – a foreign key to a record in the Enrolment table that effectively specifies the student that these restrictions apply to.

Any of these fields except for the Enrolment ID may be omitted. Blank fields are ignored. For example, if the IP Address was left blank then the student could access the materials from any machine so long as they met the other restriction criteria.

The *Security* table stores information about users that access the system. Its purpose is detailed later in this chapter.

4.2.3 Functional Analysis

There are three main roles that users of the Unit Material Management System can be assigned.

1. **Administrators** – these users have the ability to add users and units to the system. There would usually only be one or two users who are deemed to be administrators. Their primary purpose is to set up the system. These users should have very little interaction with the system except for the start of semesters when new unit offerings and new students must be entered into the system. In a production system, much of the data that is currently entered by an Administrator may be obtained from other University systems such as a student enrolment database.
2. **Unit Coordinator** – it is the responsibility of the Unit Coordinator to manage all issues related to materials for any units that they coordinate. Unit Coordinators have the ability to add materials to the system, select which materials are applicable to a certain unit, and to control the access conditions for those materials.
3. **Students** – these users only have the ability to view units for which they have been granted access.

Table 4.1 shows the functions that are available to each role of user.

Table 4.1

Unit Material Management System Functions per Role.

| Administrator | Unit Coordinator | Student |
|-------------------------|---|---------------------|
| Add units to the system | Add unit materials | View unit materials |
| Assign unit offerings | Assign a unit material to a unit offering | |
| Add locations | Assign access conditions for a student to a unit material | |
| Add teaching terms | Add IP domains | |
| Add students | Add periodic time windows | |
| Add a unit coordinator | Enrol students in a unit offering | |
| Add material types | | |
| Add material languages | | |

A particular user can be assigned more than one role. For example, it is possible that a unit coordinator may also be enrolled in other units as a student. In this case the user would have access to all functions available to the 'Unit Coordinator' and the 'Student' roles.

4.3 Implementation Issues

There were a number of issues in the implementation of the Unit Material Management System that were beyond the scope of a typical Oracle Developer 2000 development. This section explains these difficulties and how they were overcome.

4.3.1 Web Forms Installation

Getting the Developer 2000 Server product installed and fully operating was probably the most frustrating aspect of the research. Many of the initial problems were due to a differing in terminology between Edith Cowan University and Oracle Corporation. Both my supervisor and myself were aware that we required a product known as 'Web Forms'. It was not realised that the name 'Web Forms' was a term coined by some members of the Oracle development community to describe the Developer 2000 Server product. This hindered early attempts to find information about the product.

The identification of the product was also hindered by the fact that Oracle Corporation were promoting a different product as their key World Wide Web solution, making it difficult to find information about Web Forms on Oracle Corporations World Wide Web site or from their sales staff.

An e-mail was sent to the president of the Western Australian branch of the Oracle Users Group, Mr Glenn Nicholas, outlining these problems. Mr Nicholas's reply made it clear that the Developer 2000 Server product was the product commonly referred to as Web Forms (G. Nicholas, personal communication, August 26, 1998).

Edith Cowan University had a license for Developer 2000 Server version 1.6. After attempting to install this version it was apparent that it was not compatible with the other Oracle development tools that were being used. After discussions with my supervisor, Oracle Corporation gave Edith Cowan University a special license to the newer version of Developer 2000 Server (version 2.0) for the duration of my research.

The actual installation of the Developer 2000 Server product was straightforward. Difficulties arose while getting Developer 2000 Server to communicate with the Web Application Server product. As mentioned in Chapter 3, Developer 2000 Server requires access to a Web server product to enable it to operate in a Web environment. There were a number of issues that had to be resolved such as which version of the Java Developers Kit should be used, and which Web browsers were compatible with the selected version of Java. An e-mail from Ms Penny Cookson of SAGE Computing Services assisted with some of these issues (P. Cookson, personal communication, September 7, 1998). Appendix A outlines the steps used to successfully set up the Developer 2000 Server product with the Web Application Server.

4.3.2 Obtaining the IP Address

One of the requirements of the Unit Material Management System was that access to unit materials could be restricted according to the user's IP address. A common way of obtaining information about the client machine in a World Wide Web environment is to use the Common Gateway Interface (CGI) environment variables. (Powers, 1998, p. 532). In particular, the REMOTE_ADDR CGI environment variable returns the IP address of the client machine.

As mentioned in Chapter 3, the Developer 2000 Server product was initially designed for the client/server environment rather than the World Wide Web. As such it does not have the access to the CGI environment variables that most Web development products do.

The Oracle databases internal data dictionary can be configured to store information about every connection to the database, including the IP address of the machine that the connection request originates from. Unfortunately, in a Developer 2000 Server application, database connections are initiated by the Forms Server Runtime Engine which is located on the Web server. As a result, the Oracle database believes that the connection request has originated from the Web server, and therefore it is the IP address of the server machine that is stored in the database. The Oracle database is effectively insulated from the client machine.

After correspondence with a number of people through the various Oracle newsgroups available on the Internet, it was determined that gaining access to the CGI environment variables was the only reliable way to get the IP address of the client machine.

Oracle's Web Application Server was used to resolve this problem. As mentioned in Chapter 3, Oracle Web Application Server has been designed 'from the ground up' as a World Wide Web application development tool and therefore does have the ability to access such features of the Internet as CGI environment variables.

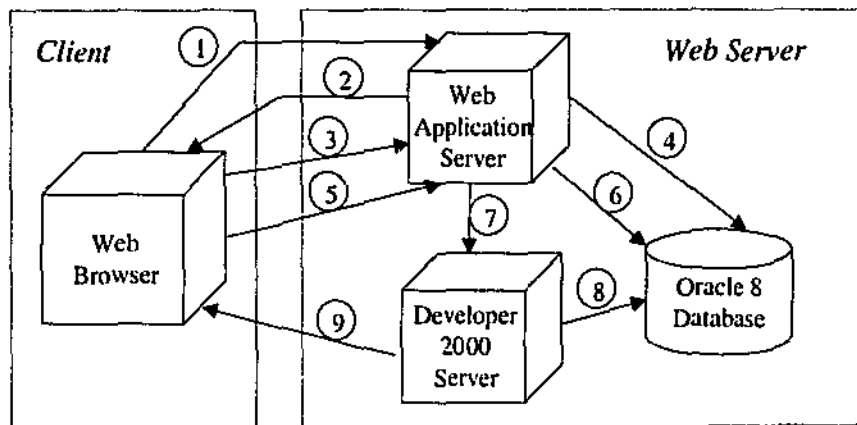
The Web Application Server product was used to provide the initial login screen for the Unit Material Management System. After the user's name and password were verified

within the Oracle database, the client machine's IP address was retrieved by making a call to the CGI environment variable 'REMOTE_ADDR'. A record was then inserted into a table named Security, containing the users' login name and the client machines' IP address. The primary key of the table was the user's login name and any previous record containing this login name was removed from the table prior to insertion. This meant that the Security table always contained the most recent IP address that a user had accessed the system from.

| login_name | ip_address |
|------------|----------------|
| drobbins | 139.154.16.234 |
| levans | 139.154.16.124 |
| bjones | 203.11.69.14 |
| esmith | 203.11.69.251 |
| mshaw | 139.69.14.54 |
| pcowan | 135.62.45.102 |
| njackson | 203.11.45.124 |

Figure 4.2 Example of the Security Table.

Developer Server 2000 Server applications are activated via a HTML page that contains information such as the form that must be loaded and the location of the relevant Java class files. This file may also be used to pass parameters to the application. In a typical Developer 2000 Server application this is simply a static HTML page that is placed on the Web server and accessed directly as the Web address of the application.



1. Browser requests web address
2. Web Application Server displays login screen
3. Browser return login information
4. Login information is verified. If invalid, return to step 2.
5. Retrieve IP address of client
6. Update Security table with user name and IP address
7. Activate Developer2000 Server - pass user name and password as parameters
8. Log user into database - retrieve user roles.
9. Display the initial form.

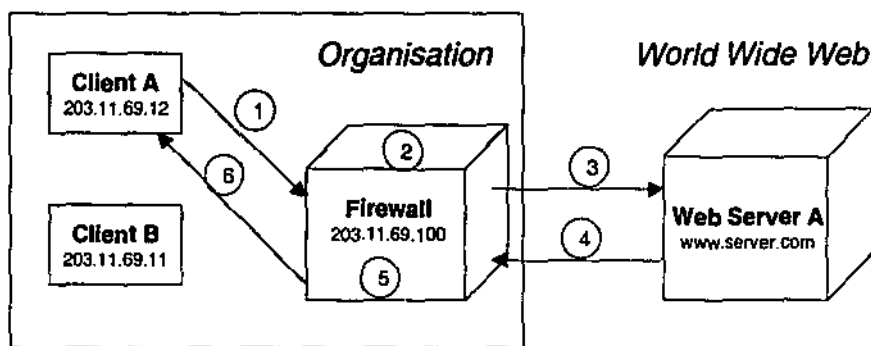
Figure 4.3 OWS - Web Forms Structure

Once the insertion of the Security record had been completed, the HTML page for the Developer 2000 Server application was dynamically generated with the user's login name and password passed as parameters to the application. Once the Developer 2000 Server component of the application was loaded the Security table was queried using the login name that had been passed as a parameter to determine the IP address of the client machine. The Web Application Server was used to initialise the security table and then pass control to the Developer 2000 Server part of the application.

4.3.3 IP Address Issues

There are two major issues that diminish the effectiveness of using CGI environment variables to get the IP address of the client machine.

The first of these issues occurs when a firewall is installed. Firewalls are servers that provide a gateway between an organisation's network and the rest of the Internet. All traffic between a client machine within the organisation's network and the rest of the Internet must pass through the firewall. To machines outside of an organisation's network, it appears that all requests are coming from the firewall server rather than the individual machine. This means that when firewalls are involved it is their IP address that is returned through the CGI environment variables instead of the IP address of the client machine within the organisation.



Client A wishes to access www.server.com

1. Client A passes the required address to the firewall server
2. Firewall determines whether Client A is allowed to access Web Server A. If not, request is terminated.
3. Firewall sends request to Web Server A - Web Server A thinks that request is coming from the Firewall! so CGI variable is set to 203.11.69.100
4. HTML page sent from Web Server A to Firewall.
5. Firewall determines whether HTML page contains any malicious code or other unauthorised data. If it does then the request is terminated.
6. Firewall forwards valid HTML page to Client A.

Figure 4.4 Accessing the World Wide Web Through a Firewall Server.

The consequence of this is that the Unit Material Management System cannot control access down to the level of individual client machines, but rather is restricted to limiting access to particular networks or sub-networks within organisations.

The second issue is that of roaming or floating IP addresses. When an organisation sets up their network they have two options with regard to their IP addresses. They may decide to have a configuration where each machine has a fixed IP address. This means that whenever a particular machine is logged on to the network it will have exactly the same IP address as it had last time it was logged on.

The other option available to organisations is to have floating or roaming IP addresses. Under this configuration there is a list of valid IP addresses that the organisation may use. When a computer is logged on to the network it is assigned the first IP address from the list that is not currently being used. This makes it unlikely that a particular machine will retain the same IP address every time it is logged on to the network.

If an organisation uses floating IP addresses then it is not practical to limit access to the Unit Material System to a particular IP address within that organisation. In this situation access could still be restricted to the first three components of the organisations IP address.

4.3.4 User Roles

There are three types of roles that a user may have within the Unit Material Management System – Administrator, Unit Coordinator, and Student. These were implemented using three corresponding Oracle Database roles – ADMIN, UCOORD, and STUDENT respectively.

These three database roles were assigned the appropriate privileges to access tables and other objects related to the Unit Material Management System. Each user of the system

was allocated one or more of these roles. The Unit Material Management System determined what roles a particular user had been allocated by querying the USER_ROLE_PRIVS view provided by the internal data dictionary of the Oracle database. The database administrator could change the behaviour of the application for a user simply by granting and revoking roles at the database level.

4.4 Flexibility Issues

This section describes a number of features of the Unit Material Management System that exhibit some of the concepts of software flexibility.

4.4.1 Common Login Screen

All users, whether they be administrators, unit coordinators, students, or some combination of the three, access the system by going to the same Web address. All of these users are presented with the same login screen. The options that a user sees after they login to the system are dependent on the roles that the particular user has been assigned.

Having a common login screen for all users has two major benefits. Firstly, only one Web address needs to be circulated and maintained. In a University situation it is possible that users with different roles may access the system from the same machine. If users of different types have to access different Web addresses then there would effectively be three applications to manage, instead of one. This also makes it easier if the role of a particular user changes. If a unit coordinator decides to further their own studies and therefore requires access to the student functions of the system, then they

would not have to change the way that they access the system. They would still access and login to the system as they had in the past, but once they have entered the system they would see an additional menu containing the student functionality.

The second benefit of having a common login screen is that users with more than one role can access functions from any of their roles in one instance of the application. If a user has the roles of Administrator and Student, then they do not have to log in to two separate applications. This makes the system easier to use and manage.

4.4.2 Hiding of Restricted Information/Functionality

When a user logs in to the Unit Material Management System, the system is able to determine what functionality that the user is allowed to use. The user interface of the system is altered so that functionality that a user is not allowed to access is hidden. The user may not even be aware of the existence of functionality that they do not have access to.

Application menus are only visible if the current user has access to the functions within that menu. If the person who is logged on to the system is a unit coordinator then they will see the Unit Management and Material Management menus. If they log out and an administrator logs in, the Unit Management and Material Management menus will disappear and the Administrator menu will be displayed. If the person who logs in to the system is a student and has no other roles within the system, then no menus are visible and the Student dialog box is loaded automatically.

There is no need for unit coordinators to have access to the access conditions that other unit coordinators have set up for their unit materials. In fact, the ability to see information about units other than those that the current user is responsible for may be unacceptable from a security aspect. The Unit Material Management System is able to identify which units the current user is responsible for and filter the information displayed accordingly.

The information that students see is also restricted to the units that they are currently enrolled in and the materials that they currently have access to. Figure 4.5 shows the dialog that students are presented with. The list of materials (in the bottom half of the dialog) only displays titles and versions of materials that the student has access to from their current machine at the current time and date. Selecting a material and then clicking the 'Display' push button results in the selected material being displayed in the user's Web browser.

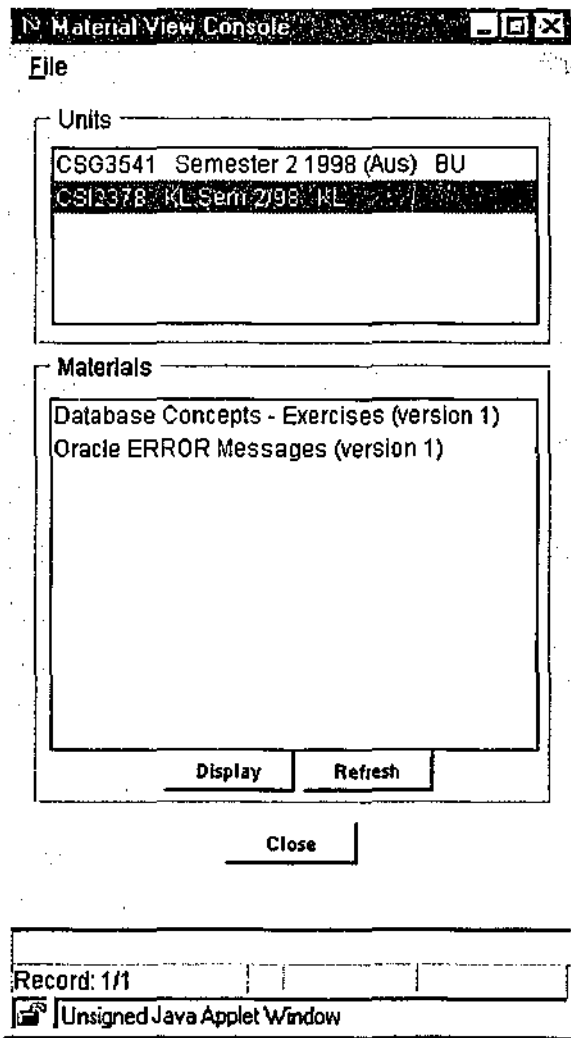


Figure 4.5 Student View Form.

4.4.3 Graphical Control of Material Access Conditions

A key software flexibility concept that was demonstrated within the Unit Material Management System was the ability of non-IT professionals, the unit coordinators in this case, to control access to unit materials. This was addressed by providing an intuitive graphical interface to the material management functions of the system.

The Material Access Conditions function of the Unit Material Management System allows unit coordinators to define restrictions to materials that have been assigned to units that they manage.

Materials For Unit Offering

Unit Code: 051232 Title: Introduction to Database Admin Location: KL Term: 1st Sem 2008

Material ID: 224 Title: Oracle Forms Release Notes Version: 4.5

Access Details

| Student ID | Name | IP Number | Time Period | Start Date | End Date |
|------------|----------|-----------------|-------------------|------------|----------|
| 0853295 | SMITH | | Monday 9am - 1pm | | |
| 0823426 | ROBBINS | 139.230.164.175 | Weekdays 11 - 5 | | |
| 0883623 | BONES | | Monday 9am - 1pm | | |
| 0833426 | ANDREW E | | Monday 9am - 1pm | | |
| 0831383 | MILLET | | Monday 9am - 1pm | | |
| 0823426 | | | Tuesday 1pm - 5pm | | |
| 0853295 | SMITH | | Tuesday 1pm - 5pm | | |
| 0883623 | BONES | | Tuesday 1pm - 5pm | | |
| 0833426 | ANDREW E | | Tuesday 1pm - 5pm | | |
| 0831383 | MILLET | | Tuesday 1pm - 5pm | | |

Add All Enrolled Students

<< < > >>

Query Mode Execute Query

New Delete Clear Save Close

Record: 6/10

Unsigned Java Applet Window

Figure 4.6 Material Access Conditions Form.

The Material Access Conditions form gives unit coordinators the facility to specify different access conditions for every student. In most cases however the unit coordinator would want to assign the same access conditions for all of the students enrolled in a particular unit delivery. The 'Add Enrolled Students' button at the bottom of the Material Access Conditions form (Figure 4.6) presents the unit coordinator with the form shown in Figure 4.7. The unit coordinator is able to specify conditions that

will be applied to all students enrolled in the current unit delivery. If the unit coordinator does not want to apply any restrictions to enrolled students they would simply leave all of the fields blank and press the OK button. This functionality allows unit coordinators to manage access to any level of granularity that they require.

N DEFAULTS

File Unit Management

Default Access Restrictions

IP Domain 139.230.184 ...

Time Period Weekdays 9 - 12 ...

Start Date

End Date

OK Cancel

Record: 1/1

[Unsigned Java Applet Window]

Figure 4.7 Default Restrictions Form.

4.4.4 Dynamic Pop-list Values

Preece (1993) states that “we can recognize material from a display far more easily than we can recall it” (p. 28). This theory has promoted the use of pop-lists, or lists of values, in many graphical user interfaces. For example, a unit coordinator may be able to recognise a unit code when presented with a list of possible unit codes. It is less likely that a unit coordinator would remember the code for each unit that they must deal with.

The Unit Material Management System provides pop-lists, or lists of values, where possible, to allow the user to select from existing values. An example of this is the use of student identification numbers. Wherever a user has to enter a student id number they have the option of being able to click a button and select from a list that shows all of the available student identification numbers and the corresponding name.

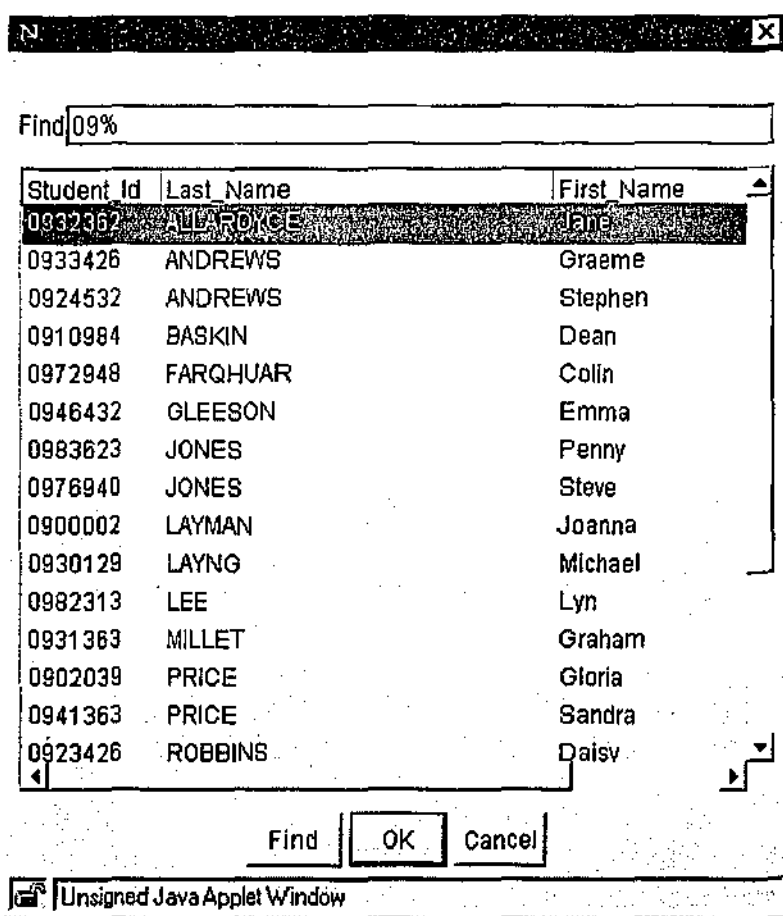


Figure 4.8 Student Pop-list.

The Unit Material Management System is flexible as the majority of these lists are created dynamically from table-stored values that the user can modify. An example of this is the pop-list for the 'type' metadata field. This pop-list is populated dynamically from the 'material_type' table in the database. Users may add, alter, or delete values

from this table (using the Material Types form) and see those modifications next time they display the list. The only pop-list in the whole application that is not dynamically created is the pop-list that allows the user to select a day of the week. These values are highly unlikely to change during the lifetime of the system.

4.5 Shortcomings of Developer 2000 Server

Although the implementation of the Unit Material Management System was successful, a number of shortcomings of the Developer 2000 Server product were encountered during the development process.

4.5.1 Access to CGI Environment Variables

This issue has already been discussed with respect to getting the IP address of the client machine. The Oracle Web Application Server provided a work-around for that particular problem. The lack of access to CGI environment variables, however, is a severe shortcoming of any Web application development tool. The CGI environment variables provide a large amount of information about the client machine that may be very useful in a number of applications. Table 4.2 shows the types of information that is accessible through CGI.

Table 4.2

CGI Environment Variables.

| Environment Variable | Meaning |
|-----------------------------|---|
| AUTH_TYPE | Specifies the authentication method, such as username/password, used by the Web browser, if any. |
| CONTENT_LENGTH | Contains the length, in characters, of the user-supplied data, if any. |
| CONTENT_TYPE | Specifies the MIME type of the user-supplied data, if any. |
| GATEWAY_INTERFACE | Designates the version of the CGI specification being used. The current version is 1.1. |
| PATH_INFO | Contains any additional path information appended to the requesting URL. |
| PATH_TRANSLATED | Contains the Web server's translation of the virtual path information, appended to the URL, to the actual path on the server machine. |
| QUERY_STRING | Contains any information appended to the URL with a question mark. |
| REMOTE_ADDR | Contains the IP address of the client machine. |
| REMOTE_HOST | Contains the domain name, if available, of the client machine. |
| REMOTE_IDENT | Contains the user's login name, if one was used for authentication with the Web server. |

| | |
|-----------------|---|
| REMOTE_USER | Contains the remote username, as supplied to the Web server. |
| REQUEST_METHOD | Specifies the request method used by the browser (GET or POST). |
| SCRIPT_NAME | Contains the virtual path and file name of the CGI script. |
| SERVER_NAME | Contains either the domain name or IP address of the Web server machine. |
| SERVER_PORT | Contains the port being used by the Web server. |
| SERVER_PROTOCOL | Specifies the protocol being used between the Web server and Web browser, typically HTTP. |
| SERVER_SOFTWARE | Contains the name and version of the Web server software. |

Note. Adapted from Powers (1998, p. 532).

4.5.2 Web Browser Compatibility

The major strength of the Java programming language is that it is platform independent. This is achieved by the use of an interpreter known as the Java Virtual Machine (JVM). Java programs are compiled into bytecode, which is independent of any native machine code. Every Web browser that supports Java has a JVM that translates the Java bytecode into the native machine code of the machine that it is being run on. (Vanhelsuwe et al., 1996).

Oracle Developer 2000 version 2.0 requires a JVM that is compatible with version 1.1 of the Java Development Kit (JDK). Unfortunately the JVMs of the two most popular Web browsers, Netscape Navigator 4.05 and Microsoft Internet Explorer 4.01, only supported JDK 1.04 as of July 1998. Fortunately an updated version of Netscape Navigator (version 4.06) was released in the second half of 1998. While this solved the compatibility problem during development, it did mean that anyone who wishes to access the Unit Material Management System via the World Wide Web would have to use Netscape Navigator 4.06 or later.

Oracle released a product to solve this problem in the second half of 1998. The product, known as JInitiator, is Oracle's implementation of the Java Virtual Machine. (Oracle Corporation, 1998c). Applications such as the Unit Material Management System can be configured so that they operate within the JInitiator product rather than the client Web browsers own JVM. The JInitiator product is compatible with both Netscape Navigator 3.0 and above, and Microsoft Internet Explorer 4.0 and above. JInitiator was not incorporated within the Unit Material Management System as its existence was not known of until September 1998.

4.5.3 Performance

One of the main concerns about using applications over the World Wide Web is that of performance. Every time a user executes an operation which requires data to be written to or queried from the Oracle database, a series of network packets must journey from the client machine to the server and back again. The performance of such a system is dependent on many factors relating to the available bandwidth and speed of the many networks that may make up the path between the client and the Web server.

The Unit Material Management System suffers from these performance issues. While the system is useable, there is a noticeable delay whenever the database is accessed.

4.6 Summary

The Unit Material Management System demonstrates concepts of software flexibility in a World Wide Web environment using Oracle's Developer 2000 Server and Web Application Server products. Successful implementation was achieved despite some issues relating to the attainment of a user's IP address.

Chapter 5: Results

5.1 Research Question 1

Research Question 1 was addressed by implementing the Unit Material Management System, a system that demonstrated some concepts of software flexibility in a World Wide Web environment.

There were two concepts of software flexibility in a World Wide Web environment that were to be implemented:

1. Access Control - The ability to control what functionality of the system that users can see based on factors such as the user's identity, the current time and date, and the IP address that the user has logged on from.
2. User Enhanceability - The ability to alter the behaviour of the application without having to alter the underlying source code.

The various issues involved in the implementation of the Unit Material Management System were discussed in detail in Chapter 4.

5.1.1 Access Control

The Unit Material Management System has the ability to restrict access to Web-enabled materials based on any combination of the student's identity, IP address, or the current time and date.

As discussed in Chapter 4, there were some issues involving the IP address of client machines that access the World Wide Web through a firewall server, or are part of a network that utilises floating IP addresses. The consequence of this is that it is only possible to restrict user access to a particular network within an organisation instead of individual client machines. While this represents a failure to completely fulfil the requirements that were specified for the Unit Material Management System, it is not an issue that would render such a system unusable. There was a consensus amongst the subjects that participated in this research that it would be very rare for unit coordinators to want to restrict access to the system down to the granularity of a particular machine.

Unit coordinators are also restricted in what they see. They are only permitted to see details for units that they are currently responsible for.

The Unit Material Management System demonstrates that access control can be implemented in a Web environment.

5.1.2 User Enhanceability

The Unit Material Management System exhibits user enhanceability in number of ways.

Firstly, the screen that a user sees when they log in to the system is totally dependent on the Oracle database roles that the user has been assigned to. As mentioned in Chapter 4, there are a number of different menus that are either visible or invisible depending on whether a particular user has been assigned to a database role. By altering the roles of a user, the way that the Unit Material Management System looks and functions for that particular user can be changed. An administrator with DBA privileges can perform the

granting and revoking of database roles using Oracle's graphical database management tools. With these tools it is simply a case of selecting a user and the role that is required to be granted or revoked.

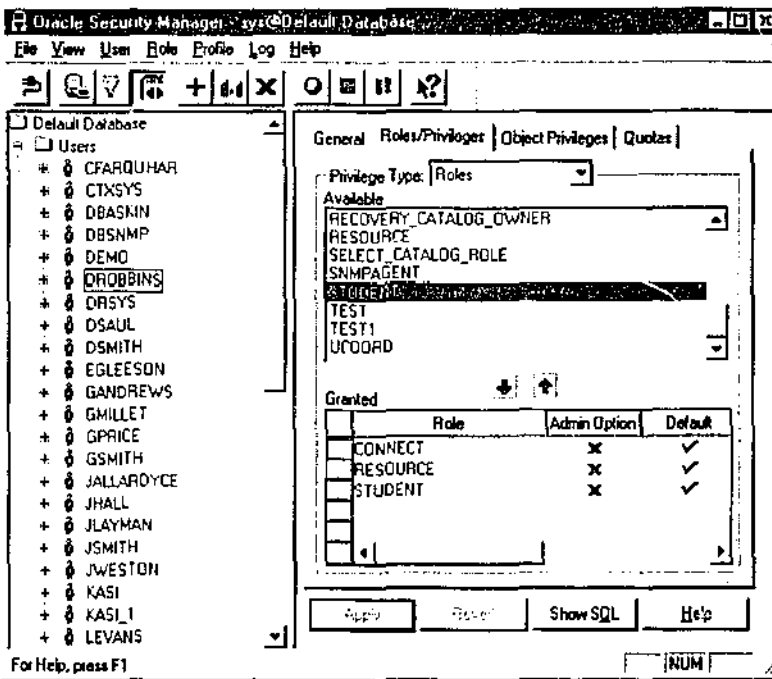


Figure 5.1 Oracle Security Manager.

The second concept of user enhanceability that is exhibited by the Unit Material Management System is the ability to change the values that are displayed in pop-lists. As indicated in Chapter 4, every pop-list within the Unit Material Management System, except for the 'day of the week' pop-list, are dynamically populated from database tables. The values stored in these tables, and therefore the values displayed in the pop-lists, can be altered by a unit coordinator using the appropriate form in the Unit Material Management System. From the user's perspective, they are able to alter what they see in a pop-list by altering values in a graphical form.

The final concept of user enhanceability demonstrated is the management of the factors by which access to Web-enabled materials is controlled. The Unit Material Management System allows a user to specify during what times, and from which IP addresses each user may access a particular material from. This is achieved using a simple graphical form. As an alternative, there are automated functions that allow a user to specify default restrictions for a group of users rather than setting restrictions individually.

Each of these three issues of user enhanceability was successfully implemented in the Unit Material Management System.

5.2 Research Question 2

5.2.1 Subjects

A total of six subjects participated in this research. Each subject gave his or her consent to be identified as part of this thesis.

Table 5.1 identifies the subjects and their job title within the University. The three columns titled 'A', 'L', and 'T' represent whether the subject's job involves, or has involved, administration, lecturing, or technical duties respectively. This information was obtained from the questionnaire.

A subject was determined to have had administration duties if they have had experience in the coordination of units or courses within the University.

Lecturing duties were specified for those subjects who have had experience with the actual presentation of units. These subjects may have been responsible for distributing materials to students that attend their lecture, but not necessarily to all of the students that take that unit at various campuses.

Subjects were considered to have had technical duties if they have had experience in the administration or setting up of a Web site.

Table 5.1

Research Subjects.

| Subject | Job Title | A | L | T |
|-------------------|--------------------------------------|----------|----------|----------|
| Mr Justin Brown | Sessional Lecturer | | √ | √ |
| Ms Marie Corrigan | Manager Virtual Campus | √ | √ | √ |
| Mr Bill Laidman | Course Coordinator | √ | √ | |
| Mr Geoff Lourens | Database Administrator | | | √ |
| Ms Jan Ring | Associate Dean (Teaching & Learning) | √ | √ | √ |
| Dr Timo Vuori | Lecturer | | √ | |

Mr Justin Brown is a PhD student as well as a sessional lecturer. He manages his own Web server for the purposes of his PhD research. He also writes all of his unit materials directly to the World Wide Web.

Ms Marie Corrigan is the manager of the University's Virtual Campus. One of the functions of the Virtual campus is to provide external students a means of accessing unit

materials. As such, Ms Corrigan has experience in both the administration of a Web server and the issues involved in the management of unit materials. Ms Corrigan has also been employed as a lecturer in the past.

Mr Bill Laidman has had minimal experience in the management of Web sites. As a senior lecturer and course coordinator, he has had a large amount of experience in the formulation and management of unit materials.

Mr Geoff Lourens administers the undergraduate student Web server at the Mt Lawley campus of Edith Cowan University. He has no lecturing experience.

Ms Jan Ring was heavily involved in the evolution of Edith Cowan University's Virtual Campus that she managed for a number of years. She held the position of Cyberspace Coordinator for the Faculty of Science and Technology for the year of 1997. Ms Ring has also had experience as a lecturer. She is currently the Associate Dean for Teaching and Learning.

Dr Timo Vuori maintains his own Web site and provides materials for three units that he lectures in. He lectures in the area of computer and Internet security.

5.2.2 Questionnaire Responses

The questionnaire included five questions directly related to the concepts of flexible software. Tables 5.2 to 5.6 list the answers given to each of the five questions by the six subjects. The question is printed in italic text before each table.

Question 1: Would unit coordinators generally like to have control over the administration of their academic materials or would they prefer that a Webmaster handle this?

Table 5.2

Responses to Question 1.

| Subject | Response |
|-------------------|--|
| Mr Justin Brown | Definitely my own control so long as the process is as automated as possible. |
| Ms Marie Corrigan | If it is easy then many of them would like to do it themselves or at least get an admin person. |
| Mr Bill Laidman | I would not like to speak for others but I would like to have some involvement in the administration of my own unit materials. |
| Mr Geoff Lourens | Unit coordinators in most cases. |
| Ms Jan Ring | Mixed – depends on their skill base and technical confidence – should give both options. |
| Dr Timo Vuori | The standard material should be managed by a Web master, but the lecturer needs to have an area to provide additional materials when needed. |

Question 2: Are there specific circumstances where unit coordinators would benefit from having such control?

Table 5.3

Responses to Question 2.

| Subject | Response |
|-------------------|---|
| Mr Justin Brown | The rapid updating and customisation of Web materials in direct response to students needs/queries. |
| Ms Marie Corrigan | Yes, it would give them more flexibility in designing their units. |
| Mr Bill Laidman | Easy preparation/modification (for a coming semester) or late modification (for an imminent semester) could be done without possible delay. |
| Mr Geoff Lourens | Yes – Exam questions and answers. |
| Ms Jan Ring | Yes – in areas where content changes rapidly – computing, multimedia, etc. |
| Dr Timo Vuori | Not sure if there are any. |

Question 3: Should all unit coordinators (from any faculty) be able to use such a system?

Table 5.4

Responses to Question 3.

| Subject | Response |
|-------------------|--|
| Mr Justin Brown | Yes, as long as adequate training was provided. |
| Ms Marie Corrigan | It should certainly be available to them. Whether they are willing or able is another matter. |
| Mr Bill Laidman | I would hope so! |
| Mr Geoff Lourens | Yes. |
| Ms Jan Ring | That means you have to train them all – huge overhead. However it should be the end goal over a 5 year period. |
| Dr Timo Vuori | Yes. |

Question 4: Do you see any problems associated with unit coordinators having this level of control?

Table 5.5

Responses to Question 4.

| Subject | Response |
|-------------------|---|
| Mr Justin Brown | No, as long as a set of rules and procedures were in place to ensure lecturers do not go beyond their units. |
| Ms Marie Corrigan | No provided it is simple (intuitive) and secure. |
| Mr Bill Laidman | I would like to see unit materials locked in place for the entire semester as the current version and only be changed with the OK from an overall controller. |
| Mr Geoff Lourens | Yes – lack of training – abuse by lecturers. |
| Ms Jan Ring | They may make changes that require University level approvals without authority – they would forget this step! Can later cause problems with appeals. |
| Dr Timo Vuori | Yes, most of them lack the required skills. |

Question 5: How important is it to hide forbidden functionality (e.g. only showing students material that they currently have access to)?

Table 5.6

Responses to Question 5.

| Subject | Response |
|-------------------|---|
| Mr Justin Brown | Important in two regards – stop students getting everything in a unit at once and therefore not attending lectures – ensure students follow course materials in a linear manner. |
| Ms Marie Corrigan | Political dynamite this question! There are arguments for it (personal id concept) but students may want to see units that they completed or units that they are thinking of doing. |
| Mr Bill Laidman | Very important. |
| Mr Geoff Lourens | Very – short option list (hundreds of units to choose from). |
| Ms Jan Ring | Depends on why it is hidden – if it is shown as unavailable you may encourage hackers or cries of ‘unfair’ – out of sight may be out of mind. It may sometimes be useful for students to see other materials in a look only mode for future planning. |
| Dr Timo Vuori | This is very important. |

5.2.3 Other Comments

There were a number of comments and suggestions provided by the subjects that were not directly addressed by any of the five questions above. These comments were made either during one of the demonstrations, or in a space provided on the questionnaire. Many of the comments regarded improvements to the Unit Material Management System functionality that did not relate directly to the issue of software flexibility, the primary research area of this thesis.

Mr Lourens, Mr Brown, and Dr Vuori all made comments regarding the issue of auditing. Mr Lourens suggested that the same user should not be able to log in to the system simultaneously from different machines. Mr Brown and Dr Vuori were interested in being able to keep track of which students were using which materials. They suggested that this may help lecturers when grievances occur over a student's marks. They could discern whether the student had spent an appropriate amount of time looking at the relevant materials.

Many subjects were concerned about ownership and security of the unit materials. Mr Laidman expressed concerns that lecturers may make changes to unit materials midway through a course, which could cause confusion among students. Dr Vuori was also concerned at the fact that all unit coordinators had complete access to unit materials for all units. This is a very valid concern that would have to be addressed in any production application.

Dr Vuori also suggested that not all materials should be subject to the level of access restriction that the Unit Material Management System provided. He suggested that materials should be assigned one of three levels of access control:

1. Global – anyone with access to the World Wide Web can view the material at any time.
2. University – anyone with a valid account for the Unit Material Management System can view the material at any time.
3. Unit – this is the level of access control currently provided by the Unit Material Management System.

This would allow the University to place materials such as information for prospective students on the Web.

Ms Corrigan and Mr Laidman commented on the granularity of control that the system gave unit coordinators. Both could see extreme situations where it would be necessary to assign different access conditions for individual students, but both agreed the majority of lecturers would assign all of their students unrestricted access. Ms Corrigan stressed that many unit coordinators, especially from non-computing based faculties, would avoid the system at first, but would embrace it once they were aware that default unit restrictions could be set by the click of a button.

Ms Corrigan recognised the usefulness of the Unit Material Management System in managing unit materials that were culturally significant. For example, different versions of a unit material could be displayed in Malaysia and Australia.

Chapter 6: Discussion

6.1 Introduction

This chapter discusses the issues that arose from the data presented in Chapter 5.

Research Question 1 was addressed by the implementation of a Web-enabled software system that exhibited techniques of software flexibility. Research Question 2 was addressed qualitatively by discussing common threads and issues that were derived from responses given by the sample population of this research.

6.2 Research Question 1

The proof of concept of implementing techniques of software flexibility in a World Wide Web environment was successfully demonstrated by the Unit Material Management System. This was achieved using Oracle's Developer 2000 Server product as the primary development tool. Issues that arose during the implementation of the system have been discussed in Chapter 4.

6.3 Research Question 2

6.3.1 Subjects

As mentioned in previous chapters, every person that was invited to be a participant in this research had a strong information technology background. It was considered that using subjects who were aware of the issues that the techniques of software flexibility

aim to address would be the optimal way of getting meaningful responses considering the limited resources available for this study.

The six people who participated in the research provided a balanced mix in terms of their areas of experience. The identity and vocational duties of the subjects were detailed in Chapter 5. Three of the subjects were considered to have extensive unit administration experience, five had performed lecturing duties, and four of the subjects were considered to have a high level of technical competence in Web site administration. This gave the study a good balance between the people involved with Web site administration, who would lose some level of control, and unit material managers, who would gain some level of administrative control.

6.3.2 Is There a Market?

Every subject indicated that there was a market for a Web-enabled system that could be administered by unit coordinators, although not all were convinced that the role of Webmaster would become any less significant than it is at present. The majority of the subjects indicated that unit coordinators would like the option of being able to administer access to their own materials, but they would not like it to be the only option. Dr Vuori went as far as to say that the standard materials for a unit should be managed by a Web administrator and that unit coordinators should just have the ability to 'fine tune' their unit by adding additional materials.

Ms Corrigan and Mr Laidman commented that many unit coordinators would not utilise all of the functionality of the Unit Material Management System. Many unit coordinators would just like to place their materials on the Web and give all of their

students unrestricted access. Other unit coordinators, however, would have a need to assign access restrictions to individual materials. The system needs to be as automated as possible to cater for unit coordinators that do not have a need to deal with the lower granularity of access restriction.

Ms Corrigan also stressed that a system such as the Unit Material Management System would have to be marketed correctly. She was concerned that many unit coordinators, especially those without an information technology background, would be apprehensive about using such a system. Ease of use and functionality such as the ability to assign default access restriction to all students enrolled in a unit (see Figure 4.7) are essential if such a system is to gain acceptance.

6.3.3 Benefits

A number of existing circumstances were identified, within the University, where the Unit Material Management System could be of benefit.

Ms Corrigan identified the internationalisation of units as one of these areas. A system such as the Unit Material Management System would allow unit coordinators to provide different versions of a material to cater for students from different cultures. For example, students undertaking a unit in Malaysia may be able to view course materials in a different language, or format, than students in Australia.

Mr Lourens suggested that the Unit Material Management System could be utilised for on-line exams. Students could be assigned a particular machine and time period for which they can view an exam. While this is a valid suggestion, there would be a

number of privacy and security issues that would have to be resolved. For example, if students were not supervised then situations could arise where they are getting help from another student. In some situations a different student may actually sit the exam.

Another area that the Unit Material Management System could be utilised is that of pay-per-view materials or courses. The system would allow the University to offer training courses to external organisations over the World Wide Web. The organisation would have access to the relevant Web-enabled materials for a set period of time, such as 9am – 5pm on weekdays between 1st and 14th of September. Their access could also be restricted so that materials could only be viewed from within the organisation's network.

Ms Ring and Mr Brown made the comment that there would be a benefit in any situation where materials or access conditions change rapidly. The Unit Material Management System would allow unit coordinators to place a new version of a material on the Web and assign access to those students that it relates to. Any students that it does not relate to would not be aware of the existence of the new version. An example of where this is applicable could be programming units where different versions of a programming language are used at different campuses.

Mr Laidman stated that the Unit Material Management System would allow unit coordinators to perform urgent modifications for their materials without having to wait for a Webmaster to assist them.

6.3.4 Useability

A key concept of software flexibility is the ability for users without a strong information technology background to be able to manage the software system. (Hall & Ligezinski, 1997b).

There was a consensus amongst the subjects that all unit coordinators should be able to use the Unit Material Management System. Approximately half of the subjects stated that some basic degree of training would have to be provided.

It is relevant that Mr Lourens was of the opinion that all unit coordinators would be able to use the system. A large portion of Mr Lourens' duties involve providing technical support for students and lecturers. He is in a position to judge the level of computer literacy among non-information technology unit coordinators.

There is no doubt that some degree of basic training would be required for all users, as there would be for most systems. This does not indicate a failure of the Unit Material Management System to exhibit software flexibility. Very few software applications are intuitive enough that users are able to use it effectively without any training. The concept of software flexibility is based around the fact that these users can modify the behaviour of the system without having to alter the source code of the application. The consensus was that this could be achieved with training.

6.3.5 Flexibility vs Control

Software flexibility allows the users of a system to easily change its behaviour. This reduces the level of control that a system administrator or Webmaster has over the

system. Ms Ring stated that if unit coordinators are able to make changes to unit materials without authorisation then there could be problems if students appeal against their grades.

Mr Laidman was also concerned about the versions of materials changing during a course. He believed that versions of materials should be locked in place at the beginning of a course or semester and only changed with the proper authorisation. This demonstrates the basic issue of balancing control and flexibility. By locking materials and only allowing them to be changed by some sort of administrative figure the system loses some of its flexibility. If the materials were not locked at the beginning of a course, unit coordinators have the ability to change what a student sees whenever they want. This could lead to a lot of confusion amongst students and, as stated by Ms Ring, problems for the University if a student should appeal against their grade.

Mr Brown stated that this issue should be addressed by ensuring that University policies and rules are enforced within the Unit Material Management System. The Unit Material Management System did not attempt to enforce University rules and policies. A production version of the Unit Material Management System would be available to a large number of unit coordinators, many of whom may only have very basic information technology skills. It is important that such a system does not give unit coordinators the ability to breach University rules and policies.

Such an extension to the Unit Material Management System would address the concerns stated by Dr Vuori and Mr Lourens, who felt that a lack of training on the part of unit coordinators could lead to abuse of the system.

Every subject inferred that the loss of control was an issue that must be addressed (see Table 5.4). When one or two system administrators are responsible for administering a system it is relatively easy to ensure that they are aware of the policies and rules that apply to a particular system. The flexibility of the Unit Material Management System effectively makes every unit coordinator within Edith Cowan University a system administrator. It is a much larger task to ensure that all of these unit coordinators are fully aware of the policies and rules that must be followed when using the Unit Material Management System. The only effective way to solve this problem is to make it impossible for rules and procedures to be broken.

6.3.6 Hiding Forbidden Functionality

As Ms Corrigan stated, this issue is 'political dynamite'. There are three areas where this is an issue in the Unit Material Management System:

1. Students may only see the titles of unit materials that they have access to.
2. Unit coordinators can only see information about units that they are responsible for.
3. Users of the Unit Material Management System can only see menu items for functions that they are allowed to access. For example, a unit coordinator can not see the menu that contains functionality that a student would access.

Subjects were asked to consider the first of these points. They had mixed views on this issue, and for a variety of reasons.

Ms Ring suggested that allowing all users to see all of the functionality of the system would invite hackers or cause users without full functionality to question why they have

restricted access to the system. If a user is not aware of the functionality that they do not have access for, then they will not miss it. Similarly, if a hacker is not aware of the existence of certain functionality then there will be less temptation to gain unauthorised access to that functionality.

This may be adequate for inexperienced users of the system. However if a user under certain conditions has access to certain functionality, and that functionality later disappears due to a change in these conditions, the user may become confused and frustrated. They may think that it is a mistake on their part and spend time trying to 'find' the missing functionality. For example, a student may have access to a particular material between the hours of 9am to 1pm. If a student accesses the system at 10am, then all is well. If that same student then accesses the system at 2pm and can not find the material that they had accessed 4 hours earlier, they may assume that either they or the system has made an error. Considerable frustration may result. This is an issue that may be solved somewhat by training on how the system works.

Ms Ring and Ms Corrigan were concerned that students would not be able to review materials for units that they have already completed, or read materials for units that they are considering doing in the future. While this is definitely a disadvantage of restricting access to materials, it is contradictory to the concept of access restriction. One way to address this issue would be to implement Dr Vuori's suggestion of having more than one level of security for Web-enabled materials. He suggested that the Unit Material Management System could be configured so that materials can be assigned one of three levels of security:

1. **Global** – anyone with access to the World Wide Web can view the material at any time.
2. **University** – anyone with a valid account for the Unit Material Management System can view the material at any time.
3. **Unit** – this is the level of access control currently provided by the Unit Material Management System.

This type of system would allow lecturers to give materials that give an overview of a unit a 'University' or 'Global' security level. Whether or not a student has access to materials for units that they have already completed is a University policy issue. While this is not currently an option in the Unit Material Management System, it could be implemented if required.

6.3.7 Suggested Enhancements

Many of the subjects suggested enhancements and extensions to the Unit Material Management System that would be required if the system was ever to go into production. It should be noted that the Unit Material Management System was never intended to be a fully functional system, but simply a vehicle to demonstrate concepts of software flexibility.

One major issue was that of material access and control amongst unit coordinators. Many subjects were concerned about the security of the materials on the system. They felt that there should be some concept of material ownership, and that it should be the responsibility of the unit coordinator that owns a particular material to decide whether that material should be accessed and assigned to units by other unit coordinators. There

were also concerns that unit coordinators had the ability to change materials midway through a semester. There were fears that this could conflict with University policies. It was suggested that access to materials be locked at the beginning of a semester, and could only be changed with approval from some senior person within the University.

The other major enhancement that arose was that of auditing. Many subjects felt that it would be useful to know which students have accessed which materials. This may help unit coordinators determine what areas students are having difficulties with, or whether or not students are actually making use of the resources available to them.

6.4 Summary

The Unit Material Management System demonstrated that the concepts of software flexibility could be implemented in a World Wide Web environment. The six subjects that viewed the system could see benefits in the applicability of software flexibility techniques, but a number of issues were raised. These were mainly University policy issues related to unit coordinators gaining more control over the management of their Web-enabled materials and a greater ability to alter what students can access.

Chapter 7: Conclusion

7.1 Findings

The Unit Material Management System, a Web-enabled application that allowed unit coordinators to manage student access to unit materials, demonstrated that the concepts of software flexibility that have been proven in third generation language environments could be implemented in a World Wide Web environment. The University staff that participated in this research viewed the Unit Material Management System and felt that there were benefits to be gained from the use of these concepts, but they did raise a number of interesting issues.

The majority of these issues were related to the enforcement of University policies and rules and relate directly to the concept of flexibility versus control. A key advantage of a flexible software application is that it gives more control over the management of that application to its users. However, it also diminishes the level of control that traditional system administrators have. This creates a situation where a large number of people, with varying degrees of information technology experience, can have a significant impact on the operation of a software system. With a World Wide Web application, these people may be distributed all over the world. One example that arose from the study was the ability of unit coordinators to change unit materials mid-way through a semester. It was felt that this could lead to confusion and complaints from students.

Giving users greater control over the management of a system requires tighter enforcement of policies within that system. If a small team of administrators manages a

system, then it may not be necessary to make it completely foolproof. Educating the administrator and making them accountable for any breaches of these policies can often be enough. When the system can be managed by many different people, of different skill levels, in different areas of the World then education and accountability is not enough. Systems must be secure enough so that it is impossible for breaches to occur.

Although each of the participants in this study could see benefits in the implementation of software flexibility concepts, it was felt that not all of the potential users of the system would wish to use it. While many unit coordinators would make use of the benefits afforded to them by the Unit Material Management System, there must still be the facility to cater for those unit coordinators that do not wish to manage the system themselves. Some unit coordinators simply may not want the responsibility of managing their unit materials themselves.

The Unit Material Management System specifically targeted the domain of academia, although the issues raised during this research would conceivably be a problem in any application domain. Whenever there is a situation where users of a varying information technology skill level have the ability to alter the functionality of a system, there is the potential for problems to occur. The greater flexibility a system has, the greater the potential for error.

7.2 Future Research

There is very little known research involving software flexibility in environments other than third generation languages. No other research has been discovered that investigates software flexibility in a World Wide Web environment. As such, there is plenty of scope for future research based on, and relating to, the findings of this study.

This thesis investigates the applicability of software flexibility to the domain of Web-enabled unit materials in academia. The findings of this thesis are that there is a benefit to be gained from the application of flexible software techniques in this domain. Further research could test whether this remains true for other domains.

This study has determined that there are benefits to be gained from implementing the techniques of flexible software in a World Wide Web environment. A major issue that has not been investigated is what the cost of implementing a flexible system is in comparison with the cost of implementing a non-flexible system that addresses a similar problem. This would be applicable to either of the 3GL, 4GL or World Wide Web environments. This would allow conclusions to be drawn as to whether the benefits provided by software flexibility justify the cost of implementing it.

7.3 Summary

Software flexibility can be implemented in a World Wide Web environment. Doing so offers a number of benefits to the users of the system while reducing the workload of traditional system administrators and Webmasters. Whether or not the cost of implementing software flexibility in a World Wide Web environment outweighs the benefits that it provides is unknown. This factor must be investigated before recommendations are made to adopt these techniques in the development of World Wide Web systems.

Definition of Terms

| | |
|---|--|
| Computer Aided Software Engineering (CASE) | A computer-based system that assists in one or more phases of the development of a software application. CASE systems are often able to provide a graphical representation of a system's design. |
| Common Gateway Interface (CGI) | A standard interface for communication between a Web server and other server programs. |
| Database Role | A set of privileges to various database objects such as table and views. Roles can be assigned to users, automatically giving users the privileges assigned to that role. |
| Database Administrator (DBA) | An individual responsible for the administration of a database. |
| E-mail | Messages sent electronically from one computer to another over the Internet. |
| Firewall | A combination of hardware and software that provides a security system, usually to prevent unauthorised access from outside to an internal network or intranet. |
| Flexible Software | Software that can be adapted to cater for a change in its requirements without altering the underlying source code. Software whose functionality can be changed without the intervention of a programmer or software engineer. Also known as 'user enhanceable software' or 'zero maintenance software'. |
| Fourth Generation Language (4GL) | A non-procedural, high level language. e.g. The Oracle Developer 2000 environment. |
| Hacker | An individual who attempts to gain unauthorised access to a computing system. |
| Hypertext Transfer Protocol (HTTP) | The protocol used by the World Wide Web for transmitting and receiving Web documents. |
| Internet | A global network of computers connected by the TCP/IP protocol. |
| Intranet | An organisation based on Internet technology, but with access restricted to a particular group or organisation. |

| | |
|---|---|
| Internet Protocol (IP) Address | The numeric address assigned by the Network Information Center (NIC) that uniquely identifies each computer on the network that uses TCP/IP. The address is made up of four groups of numbers, each separated by a period, such as 123.32.154.12. |
| Java Bytecode | Machine independent instructions produced by a Java compiler that can be executed by a Java Virtual Machine. |
| Java Virtual Machine (JVM) | A system that loads, verifies and executes Java bytecode. The interface between the machine-independent Java bytecode and the underlying hardware. |
| Lists of Values (LOV) | A graphical list that provides a user with a list of valid values for a particular field. The term 'List of Values' is Oracle terminology for a 'pop-list'. |
| Metadata | Attributes or properties about data. e.g. the creation date of a document. |
| Pop-list | A graphical list that provides a user with a list of valid values for a particular field. |
| Production System | A software system that has been released for general use in the domain that it was developed for. |
| Proxy Server | A machine providing a cache of recently accessed Web documents available on other servers that are slower or more expensive to access. |
| Software Maintenance | The alteration of an existing software system to correct a difference between what the system is required to do and what the system actually does. |
| Structured Query Language (SQL) | A natural English-based query language that allows information to be retrieved from an Oracle database. |
| Third Generation Language (3GL) | A procedural programming language, generally text based. e.g. COBOL or C++. |
| Transmission Control Protocol/Internet Protocol (TCP/IP) | A standard protocol through which computers can communicate via the Internet. |
| Web Browser | A viewer program used by a client machine to display Web pages. |
| Webmaster | An individual who administers a World Wide Web site. |

WWW

World Wide Web – a global collection of inter-connected HTML documents on the Internet.

Bibliography

- Blum, B. I. (1993). Representing Open Requirements with a Fragment-based Specification. Man and Cybernetics, 23 (3), 724-736.
- Booch, G. (1991). Object Oriented Design with Applications. Redwood City, CA: The Benjamin/Cummings Publishing Company, Inc.
- Bottaci, L., & Mehandjiev, N. (1996). User-enhanceability for organisational information systems through visual programming. Proceedings, Lecture Notes in Computer Science 1080, 432-456.
- Chapell, D. (1996). Understanding ActiveX and OLE. Redmond, WA: Microsoft Press.
- Cluts, N.W. (1998, February). DHTML? Applets? Controls? Which To Use? [on-line]. Available WWW: <http://www.microsoft.com/sitebuilder/features/dhtmlvsctrls.asp> [1998, May 2].
- Department of Industry, Science and Tourism. (1998, April). Electronic Commerce in Australia [on-line]. Available WWW: <http://www.dist.gov.au> [1998, June 1].
- Dynamic Information Systems LLC. (1998). Oracle Web Application Server Handbook. New York: McGraw Hill.
- Edwards, M. (1997, October). 1001 Ways to Get Input from Web Users [on-line]. Available WWW: <http://www.microsoft.com/sitebuilder.microsoft.com/products/sitebuilder/works hop/author/script/1001ways.asp> [1998, April 27].
- Ensor, D., & Stevenson, J. (1997). Oracle Design. Sebastopol, CA: O'Reilly & Associates, Inc.
- Hall, M.J.J., Ligezinski, P. (1997a). Designing flexible software to accommodate dynamic user requirements: An alternative solution to a continuing IS problem. In Proceedings of World Conference on Systemics, Cybernetics and Informatics ISAS '97 Vol. 1. Caracas, Venezuela: International Institute of Informatics and Systemics, Universidad Simon Bolivar.
- Hall, M. J. J., & Ligezinski, P. (1997b). Developing flexible software with Oracle tools. In Proceedings of Oracle Openworld 1997 Conference – Step Into the Future Today. Melbourne, Vic: ANZORA.
- Harmon, T. (1996). Visual J++ and ActiveX. Scottsdale, AZ: Coriolis Group Books.
- Hilimann, D. (Ed.). (1998, July). A User Guide For Simple Dublin Core [on-line]. Available WWW: http://purl.oclc.org/dc/documents/working_drafts/wd-guide-current.htm [1998, October 5].

- Iannella, R., & Ward, N. (1998, May). A Day in the Life of MetaData. Presented by Distributed Systems Technology Centre, Raddison Observation City, Perth, WA.
- Johnson, B., Ligezinski, P. & Woolfolk, W. W. (1996). The problem of the dynamic organization and the static system: Principles and techniques for achieving flexibility. Proceedings of the 29th Hawaii International Conference on System Sciences, IEEE Computer Society Press Volume 3, 482-491.
- Lea, D. (1997). Concurrent Programming In Java. Reading, MA: Addison-Wesley Publishing Company.
- Malim, T., & Birch, A. (1997). Research Methods and Statistics. London: MacMillan Press Ltd.
- McNurlin, B. C., & Sprague, R. H. Jr. (Eds.). (1993). Information Systems Management in Practice. (3rd ed.). Englewood Cliffs, NJ: Prentice Hall International.
- Miller, E. (1998, May). An introduction to the Resource Description Framework. D-Lib Magazine [on-line]. Available WWW: <http://www.dlib.org/dlib/may98/miller/05miller.html> [1998, July 8].
- Miller, J., & Resnick, P. (1996). PICS: Internet Access Controls Without Censorship [on-line]. Available WWW: <http://www.bilkent.edu.tr/pub/WWW/PICS/iacwc.htm> [1998, July 7].
- Network Wizards. (1997, January). Internet Domain Survey [on-line]. Available WWW: <http://www.nw.com/zone/WWW-9701/report.html> [1998, May 1].
- Oracle Corporation. (1997a, June). Oracle 8 Enterprise Edition Release 8. Redwood Shores, CA: Oracle Corporation.
- Oracle Corporation. (1997b, July). Oracle Developer/2000 For the Web. (Revision 1.1). Redwood Shores, CA: Oracle Corporation.
- Oracle Corporation. (1998a, August). Oracle First to Deliver Enterprise Application Server with Newest Release [on-line]. Available WWW: <http://www.oracle.com/cgi-bin/press/printpr.cgi?file=980825.11763.html&mode=owpr&td=1&tm=9&fd=1&fm=08&status=Search&ty=98&limit=50&fy=98> [1998, October 7]
- Oracle Corporation. (1998b, August). Oracle JDeveloper [on-line]. Available WWW: <http://www.oracle.ru/products/tools1/jdeveloper/index.html> [1998, November 25].
- Oracle Corporation. (1998c, August). Oracle JInitiator [on-line]. Available WWW: <http://www.oracle.com/products/tools/dev2k/dn.html> [1998, November 25].
- Parnas, D. L. (1979). Designing software for ease of extension and contraction. IEEE Transactions on Software Engineering, 5 (2), 128-137.

- Powers, S. (Ed.). (1998). Dynamic Web Publishing (2nd ed.). Indianapolis, IN: Sams.net Publishing.
- Preccc, J. (Ed.). (1993). A Guide to Usability: Human Factors in Computing. Reading, MA: Addison-Wesley Publishing Company.
- Pressman, R. S. (1992). Software Engineering – A Practitioners Approach (3rd ed.). New York: McGraw Hill Inc.
- Vanhelsuwe, L., Phillips, I., Hsu, G.T., Sankar, K., Ries, E., Rohaly, T. & Zukowski, J. (1996). Mastering Java. San Francisco: Sybex.

Appendix A: Oracle Developer 2000 Server Configuration

A.1 Introduction

This appendix outlines the steps that were taken to get the Oracle Developer 2000 Server product working on the Web server that hosted the Unit Material Management System (see the 'Environment' section in Chapter 3).

This appendix assumes that Oracle Web Application Server (version 3.01) is already installed on the Web server machine and has a Web address of 'http://hd19696.fste.ac.cowan.edu.au'. This address, as well as port numbers and physical directory paths will probably change for every installation of the Developer 2000 Server and the Web Application Server. This appendix also assumes that the reader has some knowledge about the workings of the Oracle Web Application Server.

All Oracle products were installed under the 'D:\ORANT' directory on the Web server machine.

A.2 Set Up The Web Server Listener

The first step is to create a Web server listener for the Developer 2000 Server application. The Web server listener for the Unit Material Management was named 'wfrms' and configured to operate on port 9000. The values for the name and the port number can be altered to meet the requirements of a particular installation.

1. Load the Oracle Web Application Server Administration page ('http://hd19696.fste.ac.cowan.edu.au:8888').
2. Click the 'Web Application Server' hyperlink.
3. Click the 'Oracle Web Listener' hyperlink.
4. Click the 'Create Listener...' button.

A Web page titled 'Basic Configuration' should appear in the Web browser. This page allows the developer to enter preliminary values for the creation of the Web server listener. Table A1 lists the settings that were used for the Unit Material Management System. Any settings not listed were left as the default value that was supplied by Oracle Web Application Server.

Table A.1

Web Server Listener 'Basic Configuration' Settings.

| Variable | Value |
|-----------------|------------------------------|
| Listener Name | wfrms |
| Port Number | 9000 |
| Host Name | hd19696.fste.ac.cowan.edu.au |
| Document Root | D:\ORANT\OWS\3.0\DOC\ |

After these settings are entered, the 'Advanced Configuration' button should be clicked. Confirmation that the Web server listener was successfully created should appear.

The 'Advanced Configuration' contains settings that may not be required for simple Web server listeners. The listener for the Unit Material Management System requires additions to the 'Directory Mappings' section. Directory mappings define the physical directory on the Web server machine that directories in the Web address (known as virtual directories) represent. For example, in the Web address 'www.server.com/data/index.html' there is a virtual directory called 'data'. This virtual directory must represent a physical directory, such as 'D:\WEBDATA'.

Table A2 specifies the directory mappings that were entered for the Unit Material Management System. Directory mappings that were already specified by the Oracle Web Application Server should not be altered or removed.

Table A.2

Directory Mappings.

| Physical Directory | Virtual Directory |
|---------------------------|--------------------------|
| D:\ORANT\FORMS50\JAVA\ | /web-code/ |
| D:\MLAYNG\WEBHTML\ | /web-html/ |
| D:\ORANT\FORMS50\JAVA\ | /web-jars/ |

The 'web-code' virtual directory identifies the directory that contains the Oracle Forms (.FMX) files and the Java class files. The 'web-jars' directory contains the Oracle JAR files. Files that have to be downloaded from the Web server to the client when a Web

page is accessed can all be bundled together into a single JAR file, reducing the load on the Web server. Oracle provides a JAR file that contains all of the class files that are required for the Developer 2000 Server product.

The 'web-html' virtual directory contains the HTML files that the application requires. In the case of the Unit Material Management System there is only one, 'STATIC.HTML' (see Figure A.1).

Once the directory mappings have been entered, the 'Modify Listener' button should be clicked to save these settings. Confirmation of a successful modification indicates that the setting up of the Web server listener is complete.

A.3 Set Up the Java Developers Kit

Version 1.1 of the Java Developers Kit (JDK) must be installed under the '/web-code/' virtual directory. For the Unit Material Management System this meant that the JDK directory was created under 'D:\ORANT\FORMS50JAVA'. There is a copy of JDK 1.1 on the Oracle Developer 2000 Server CD-ROM.

The Oracle Developer 2000 Server installation program should have placed some extra Java class files under the directory 'D:\ORANT\FORMS50JAVA\ORACLE'. If this directory is not present then the installation program should be run again.

A.4 Create the Static HTML Page

The next step is to create the HTML file that will coordinate the loading of the Unit Material Management System application. This is the initial HTML file that the Web browser calls when the application is accessed. This file specifies information that the Oracle Forms Server Listener (discussed in the next section) requires.

```
<HTML>
<!-- FILE: static.html -->
<!-- Oracle Static (Non-Cartridge) HTML File Template (Windows NT) -->
<!-- Rename, and modify tags and parameter values as needed -->

<HEAD><TITLE>Developer/2000 Server</TITLE></HEAD>

<BODY><BR>Please wait while the Forms Client class files download and run.
  <BR>This will take a second or two...
<P>

<!-- applet definition (start) -->
<APPLET CODEBASE="http://hd19696.fste.ac.cowan.edu.au:9000/web-code/"
  CODE="oracle.forms.uiClient.v1_4.engine.Main"
  ARCHIVE="http://hd19696.fste.ac.cowan.edu.au:9000/web-jars/f50all.jar"
  HEIGHT=400
  WIDTH=600>

<PARAM NAME="serverPort" VALUE="5555">

<PARAM NAME="serverArgs" VALUE="module=material_management_main.fmx">

</APPLET>
<!-- applet definition (end) -->

</BODY>
</HTML>
```

Figure A.1 Static HTML File.

The following 'APPLET' parameters control the initialisation of the Unit Material Management System:

- **CODEBASE** – specifies the virtual directory that contains binary Oracle Forms files.
- **CODE** – specifies the initial Java class and method to be called.
- **ARCHIVE** – specifies the JAR file that must be downloaded to the client machine when the Unit Material Management System is accessed.
- **serverPort** – The port number that the Forms Server Listener (see below) is operating on.
- **serverArgs** – Specifies the initial Oracle Forms binary file to load.

A.5 Set Up the Forms Server Listener

The Forms Server Listener coordinates the communication between the client forms and the database on the Web server. This listener must be running on the Web server machine for the Developer 2000 Server product to work. To start the listener you must enter the following at the command line: `D:\ORANT\BIN\F50SRV32.EXE port=5555`

The port number, '5555', must correspond with the 'serverPort' parameter in the static HTML file shown in Figure A.1.

Appendix B: Unit Material Management System Specification

B.1 Introduction

This appendix provides a basic specification for the Unit Material Management System. It specifies the algorithms and features that were important in the implementation of the concepts of software flexibility within the system. It does not specify standard trigger coding that is a part of any Oracle Forms development. The SQL scripts that specify the low-level structure of each database object, as well as the privileges assigned to each database role are available in Appendix C. It must be noted that the Unit Material Management System was simply a vehicle for research. The requirements and functionality may not completely reflect University policies.

All forms within the Unit Material Management System were designed to 'look and feel' like standard Oracle Developer 2000 forms. All of the standard querying and navigation buttons were included.

B.2 System Requirements

The Unit Material Management System must control access to Web-enabled unit materials that are assigned to a unit. Access to each Web-enabled unit material for each unit may be restricted by the following factors:

- **Who** - The identity of the user. A user must be enrolled in an academic unit to access materials assigned to that unit.
- **Where** - The IP address that the user is attempting to access the materials from.
- **When** - The current time and date. This may consist of either:
 6. a periodic time window. For example, 9am to 1pm every Monday and Wednesday.
 7. a fixed time period. For example, between 1st of March 1998 and the 31st of July 1998.

These access conditions must be able to be controlled by the coordinator of each unit through a graphical, Web-enabled interface. It is essential that the system is user friendly enough to be used by unit coordinators with minimal information technology experience.

B.3 Entity Relationship Diagram

Figure B.1 shows the data model for the Unit Material Management System. The SQL scripts that specify the structure of each table and their constraints can be found in Appendix C. The purpose of each table is discussed in Chapter 4.

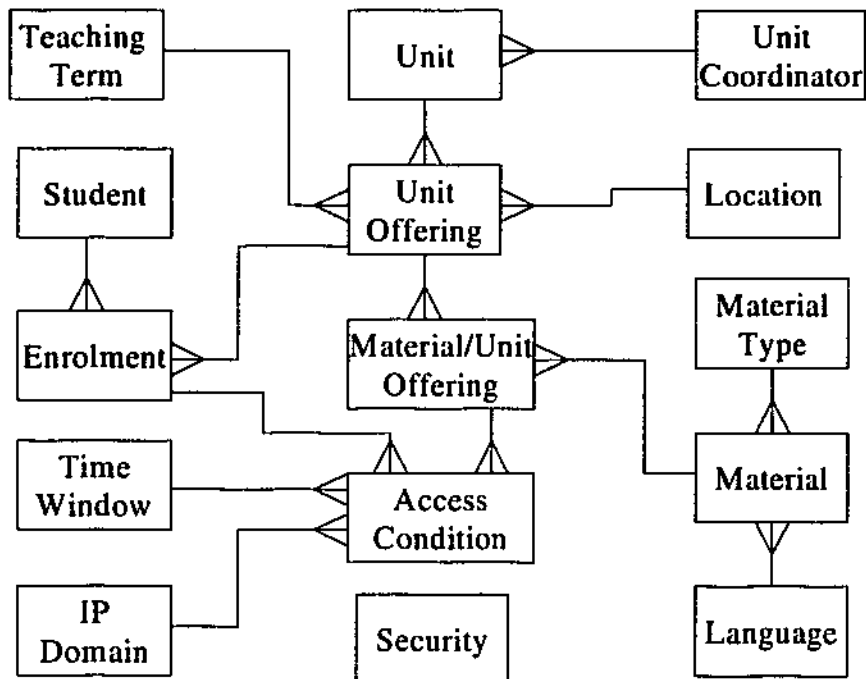


Figure B.1 Entity Relationship Diagram.

B.4 Menu Structure

There were four main menus in the Unit Material Management System. The 'Admin' menu could only be seen by system administrators, the 'Unit Management' and the 'Material Management' menus could only be seen by unit coordinators, and the 'Student' menu could only be seen by students.

Table B.1 specifies the functions available in each menu. The sections B.5 to B.8 specify the purpose of the functions that are accessed via each menu.

Table B.1

Menu Structure.

| Admin | Unit Management | Material Management | Student |
|-------------------|------------------------|-------------------------------|-----------------------|
| Unit Offerings | IP Domains | Materials | Material View Console |
| Unit Coordinators | Time Windows | Assign Materials | |
| Students | Enrolments | Assign Material Access Rights | |
| Course Locations | | | |
| Teaching Terms | | | |
| Material Types | | | |
| Material Language | | | |

B.5 Admin Menu Functions

B.5.1 Unit Offerings

This function allows a system administrator to add/edit unit offerings. An offering of a unit must have a location ('Where'), a course interval ('When'), and a unit coordinator specified. The pop-lists for these values are populated dynamically from the database.

UNIT'S

File Admin Unit Management Material Management
Student

Unit

Unit Code

Unit Title Introduction to Database Admin

Unit Description Basic Oracle admin concepts

Unit Coordinator jhall ...

Offerings

| Where | When | Start Date | End Date |
|----------------------|-----------------------|----------------------|----------------------|
| ML ... | Semester 2 1998 (... | 01/07/1998 | 31/12/1998 |
| KL ... | KL Sem 2/98 | 01/09/1998 | 30/12/1998 |
| JO ... | Semester 1 1999 (... | 01/01/1999 | 30/06/1999 |
| <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> |

<< < > >>

Query Mode Execute Query

New Delete Clear Save Close

Record: 3/7

Unsigned Java Applet Window

Figure B.2 Unit Offerings Form.

B.5.2 Unit Coordinators

This function allows a system administrator to specify details about unit coordinators.

The login name must correspond to the unit coordinator's Oracle account login name.

If a unit coordinator does not exist in this table, then they will be unable to access the system.

Unit Coordinator

File Admin Unit Management Material Management Student

Coordinators

Login Name

Full Name

<< < > >>

Query Mode Execute Query

New Delete Clear Save Close

Record: 3/7

Unsigned Java Applet Window

Figure B.3 Unit Coordinator Form.

B.5.3 Students

This function allows a system administrator to add students to the system and edit the details of existing students. This function also allows a system administrator to view the unit offerings that a student is currently enrolled in. Actual enrolments into unit offerings are performed by the unit coordinator using the 'Enrolments' function.

Student Details

Student Id

Last Name

First Name

Initials

Address

Login Name

Enrolments

| Unit Code | Unit Title | When | Where |
|-----------|----------------------|-----------|-------------|
| CSG3541 | Software Engineering | 2004-2005 | Engineering |
| CSI2378 | Introduction to CS | 2004-2005 | Engineering |
| | | | |
| | | | |
| | | | |

<< < > >>

Query Mode Execute Query

New Delete Clear Save Close

Record: 1/?

Unsigned Java Applet Window

Figure B.4 Students Form.

B.5.4 Course Locations

This function allows system administrators to specify valid locations that a unit may be offered. The location may be a physical place, such as 'Mt. Lawley Campus', or a virtual location such as 'External Studies'. The information entered here populates various pop-lists throughout the system.

The screenshot shows a Java applet window titled "Locations". The menu bar includes "File", "Admin", "Unit Management", and "Material Management". The main content area is titled "Locations" and contains the following fields and controls:

- Code:** A text box containing "MTE".
- Name:** A text box containing "Mt Lawley".
- Country:** A text box containing "Australia".
- Address:** A text box containing "Bradford St".

Below the input fields are several control buttons:

- Navigation buttons: "<<", "<", ">", ">>".
- Buttons: "Query Mode", "Execute Query".
- Bottom row buttons: "New", "Delete", "Clear", "Save", "Close".

At the bottom of the window, there is a status bar showing "Record: 2/?" and "Unsigned Java Applet Window".

Figure B.5 Course Location Form.

B.5.5 Teaching Terms

This function allows a system administrator to specify valid terms that a unit may be offered for. A term is any period of time over which a unit is offered. It could be a standard University semester, a Summer School period, or a two week bridging course. The label that is given to the term appears in pop-lists throughout the system.

The screenshot shows a Java Applet window titled "Teaching Terms". The window has a menu bar with "File", "Admin", and "Unit Management". Below the menu bar is a form titled "Teaching Term" with three input fields: "Label" (containing "SEMESTER"), "Start Date" (containing "01/01/1998"), and "End Date" (containing "30/06/1998"). Below the form are navigation buttons: "<<", "<", ">", ">>"; "Query Mode" and "Execute Query"; and "New", "Delete", "Clear", "Save", and "Close". At the bottom of the window, it says "Record: 1/?" and "Unsigned Java Applet Window".

Figure B.6 Teaching Term Form.

B.5.6 Material Types

This function allows a unit coordinators to specify valid values for the Dublin Core metadata element, 'Type' that can be specified for unit materials within the Unit Material Management System. Values that are entered here will appear in the pop-list in the 'Materials' function.

The screenshot shows a Java applet window titled "Material Types". The window has a menu bar with "File", "Admin", "Unit Management", "Material Management", and "Student". Below the menu bar is a section titled "Material Types" containing a list of text input fields. The fields are labeled: "Slides", "Notes", "Exam", "Assignments", "Lectures", "Spl Exam", "Exercises", and "Other". There are several empty input fields below "Other". At the bottom of the window, there are two buttons: "Query Mode" and "Execute Query". Below the buttons, it says "Record: 9/9" and "Unsigned Java Applet Window".

Figure B.7 Material Types Form.

B.5.7 Material Language

This function allows a unit coordinators to specify valid values for the Dublin Core metadata element, 'Language' that can be specified for unit materials within the Unit Material Management System. Values that are entered here will appear in the pop-list in the 'Materials' function.

The screenshot shows a web application window titled "N Languages". The menu bar includes "File", "Admin", and "Unit Management". The main content area is titled "Languages" and contains a vertical list of text input fields. The first three fields are populated with "English", "Japanese", and "French". Below the list are several buttons: "Query Mode", "Execute Query", "Delete", "Save", and "Close". At the bottom of the window, it displays "Record: 4/4" and "Unsigned Java Applet Window".

Figure B.8 Material Language Form.

B.6 Material Management Menu Functions

B.6.1 Materials

This function allows unit coordinators to specify materials to be controlled by the Unit Material Management System. Each material is assigned an automatically generated identification number. The unit coordinator must specify a title, version, and current Web address for each material. They also have the ability to define any of the remaining Dublin Core metadata elements for that material but this is not compulsory.

The screenshot shows a web application window titled "Materials". The menu bar includes "File", "Unit Management", and "Material Management". The main content area is a form with the following fields and values:

| | |
|-------------------|--|
| ID | 165 |
| Version | 1.00 |
| Title | Database Concepts - Answers |
| Web Address | http://web-htm/database_concepts_answers.htm |
| Creator | Michael Layng |
| Subject | RDMS Concepts |
| Description | |
| Publisher | |
| Other Contributor | |
| Create Date | 02/04/1995 |
| Type | Exercises |
| Source Material | |
| Language | English |
| Relation | Answers to "Database Concepts - Exercises" |
| Coverage | |
| Rights | |

Navigation buttons: << < > >> Query Mode Execute Query New Delete Clear Save Close

Status bar: Record: 3/7 Unsigned Java Applet Window

Figure B.9 Materials Form.

B.6.2 Assign Materials

This function allows a unit coordinator to assign a material to a unit offering that they are responsible for. If a unit material is not assigned to a unit offering, then students in that unit cannot be granted access to it.

The screenshot shows a web application window titled "Assign Materials To Unit Offerings". The interface is divided into several sections:

- Unit Offering:** A form with three fields: "Unit" (dropdown menu showing "CS 302A"), "Term" (dropdown menu showing "4"), and "Location" (dropdown menu showing "ML"). To the right of these fields are text input boxes containing "Introduction to Database Admin", "Semester 2 1998 (old)", and "ML Lauder".
- Materials:** A table with columns for "ID", "Version", and a description. The first three rows are populated with data:

| ID | Version | Description |
|-----|---------|-------------------------------|
| 126 | 1 | Oracle ERROR Messages |
| 42 | 1 | Database Concepts - Exercises |
| 165 | 1 | Database Concepts - Answers |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Below the table are navigation buttons: "<<", "<", ">", ">>". Below these are two buttons: "Query Mode" and "Execute Query". At the bottom of the interface are five buttons: "New", "Delete", "Clear", "Save", and "Close".

At the very bottom of the window, there is a status bar that reads "Record: 2/?" and a security warning icon with the text "Unsigned Java Applet Window".

Figure B.10 Assign Materials Form.

B.6.3 Assign Material Access Rights

This function allows a unit coordinator to determine the restriction on materials for each student in their units. The 'Materials For Unit Offering' section of the form displays one material that is assigned to one of the unit coordinators units. The arrow buttons at the base of the form can be used to scroll through these materials.

The 'Access Details' section of the form controls who can see a material, where they can see it from, and when. If a student does not appear in this section, then they can not access a material. Any of the 'IP Number', 'Time Period', 'Start Date', or 'End Date' fields may be left blank, indicating that there is no restriction on that factor. For example, if the 'IP Number' field is blank, then a student may access a material from any computer. Students, IP numbers, and time periods can be selected from dynamically created pop-lists.

Often a unit coordinator will not want to specify restrictions fro each individual student. The 'Add All Enrolled Students' button will bring up the form shown in Figure B.12. The user may enter restrictions in to this form. If they select 'OK', these restrictions will be applied to all students that are enrolled in the current unit offering.

It is possible that a student will have more than one restriction entry for a material. They may be allowed to view a material on Mondays, 9am-11am, and on Thursdays, 4pm-6pm. A single can have as many entries as necessary. So long as one of these restriction entries is true when a student accesses the system, they will be able to view the material.

Assign Material Access Rights

File Unit Management Material Management

Materials For Unit Offering

Unit Code: 0703426 Title: Introduction to Database Admin Location: 111 Term: Fall Sem 2003

Material ID: Material Form Release Notes Version: 2.4

Access Details

| | | IP Number | Time Period | Start Date | End Date |
|---------|---------|-----------------|-------------------|------------|----------|
| 0023426 | ROBBINS | 139.230.164.175 | Weekdays 11 - 5 | | |
| 0083623 | JONES | | Monday 9am - 1pm | | |
| 0033426 | ANDREWS | | Monday 9am - 1pm | | |
| 0031363 | MULLEN | | Monday 9am - 1pm | | |
| 0023426 | ROBBINS | | Tuesday 1pm - 5pm | | |
| 0053205 | SMITH | | Tuesday 1pm - 5pm | | |
| 0083623 | JONES | | Tuesday 1pm - 5pm | | |
| 0033426 | ANDREWS | | Tuesday 1pm - 5pm | | |
| 0031363 | MULLEN | | Tuesday 1pm - 5pm | | |

Add All Enrolled Students

<< < > >>

Query Mode Execute Query

New Delete Clear Save Close

Record: 1/7

Unsigned Java Applet Window

Figure B.11 Assign Material Access Rights Form.

DEFAULTS

File Unit Management

Default Access Restrictions

IP Domain: 139.230.164

Time Period: Weekends 9 - 4

Start Date:

End Date:

OK Cancel

Record: 1/1

Unsigned Java Applet Window

Figure B.12 Default Access Rights Form.

B.7 Unit Management Menu Functions

B.7.1 Enrolments

This function allows a unit coordinator to enrol students into a particular unit offering. A student must be enrolled in an offering to be assigned access to a material. In a production system, this information would be gained from other University systems. As with all unit coordinator functions within the Unit Material Management System, the unit coordinator can only see offerings that they are responsible for.

Enrol Student in an Existing Unit

File Unit Management Material Management

Unit Offerings

| | | |
|-----------|--------|--------------------------------|
| Unit Code | 093076 | Introduction to Database Admin |
| Term | 4 | Semester 2 2006 (Aut) |
| Location | ML | ML Library |

Students

| Student ID | Name | Surname |
|------------|------|---------|
| 0930129 | ... | ... |
| 0952263 | ... | SMITH |
| 0929573 | ... | BOSTON |
| 0902039 | ... | PRICE |
| 0900002 | ... | OSMAN |
| 0946432 | ... | GLEESON |
| | | |
| | | |
| | | |

<< < > >>

Query Mode Execute Query

New Delete Clear Save Close

Record: 2/?

Unsigned Java Applet Window

Figure B.13 Enrolment Form.

B.7.2 IP Domains

This function allows a unit coordinator to specify IP domains that a student may access a material from. The numeric IP address specified can be a partial address to specify a particular network. For example, an IP address of '139.230.164' would identify all machines with an IP address beginning with '139.230.164'.

The screenshot shows a web-based form titled "IP Domains" within a browser window. The form has a menu bar with "File", "Unit Management", and "Material Management". The main form area contains three input fields: "Ip Number" with the value "139.230.164", "Ip Address" with the value "ECU Mt Lawley", and "Domain Name" with the value "General". Below these fields are navigation buttons: "<<", "<", ">", ">>"; "Query Mode" and "Execute Query"; and "New", "Delete", "Clear", "Save", and "Close". At the bottom, there is a status bar showing "Record: 4/4" and a security warning "Unsigned Java Applet Window".

Figure B.14 IP Domain Form

B.7.3 Time Windows

This function allows a unit coordinator to specify a particular window of time that a student can view a material for. The 'Time Period Id' field is simply a label that will appear in pop-lists throughout the system. Unit coordinators should make this label as meaningful as possible. For example, 'Mt Lawley Database Workshop'.

The remaining three fields must always contain a value. The 'Day of the Week' may either be Monday to Friday, or one of the two special values, 'Weekdays' (indicates Monday to Friday) or Weekends (Saturday and Sunday). The hour start and end must be a value between 1 and 24, representing the hour of the day that access may start or end. The end hour must be of a greater value than the start hour.

| Time Period Id | Day Of Week | Hour Start | Hour End |
|---------------------|-------------|------------|----------|
| database admin wshp | Monday | 10 | 12 |
| Monday 1pm - 4pm | Monday | 13 | 16 |
| Tuesday 9am - 1pm | Tuesday | 9 | 13 |
| Tuesday 1pm - 5pm | Tuesday | 13 | 17 |
| Tuesday 5pm - 9pm | Tuesday | 17 | 21 |
| Weekdays 11 - 5 | Weekdays | 11 | 17 |
| Weekends 9 - 4 | Weekends | 9 | 16 |
| Weekdays 5 - Midnqt | Weekdays | 17 | 24 |
| Weekdays 9 - 12 | Weekdays | 9 | 12 |

Figure B.15 Time Window Form

B.8 Student Menu Functions

The Material View Console is the only function that students have access to. This function allows a student to view materials that they currently have access to. The 'Units' list box displays the unit offerings that a student is currently enrolled in. When one of these unit offerings is selected, the available materials for that unit offering are displayed in the 'Materials' list. Only the materials that the user currently has access to are displayed.

Clicking the 'Display' button displays the material in the student's Web browser.

The screenshot shows a Java applet window titled "Material View Console". At the top, there are menu options: "File", "Admin", and "Unit Management". Below the menu is a section labeled "Units" containing a list box with one entry: "CSI2378 Semester 2 1998 (Aus) ML". Below the "Units" section is a section labeled "Materials" containing a list box with four entries: "Database Concepts - Exercises (version 1)", "Database Concepts - Answers (version 1)", "Assignment Submission Requirements (version 1)", and "Micromine Home Page (version 1)". At the bottom of the "Materials" section are two buttons: "Display" and "Refresh". Below the "Materials" section is a "Close" button. At the very bottom of the window, there is a status bar showing "Record: 1/1" and "Unsigned Java Applet Window".

Figure B.16 The Material View Console Form

B.9 Procedure Listings

This section provides listings of the most complex procedures from the Unit Material Management System.

B.9.1 Menu Hiding

Figure B.17 shows the 'WHEN-NEW-FORM-INSTANCE' trigger for the main application form of the Unit Material Management System. This is the form that is first called after a user has logged in to the system. The trigger queries the database's data dictionary to determine what roles the user has assigned to them. It then shows or hides each of the four menus (described previously) according to these roles.

```
DECLARE
    bAdmin BOOLEAN;      -- is user an ADMIN
    bUCoord BOOLEAN;    -- is user a UCOORD
    bStudent BOOLEAN;   -- is user a STUDENT
    nNumber NUMBER;
    mi_id MenuItem;     -- menu id
    mi_id2 MenuItem;    -- menu id

BEGIN

    -- check to see if user is admin
    select count (granted_role) into nNumber from user_role_privs where
    granted_role = 'ADMIN';

    IF nNumber > 0 THEN
        bAdmin := TRUE;
    ELSE
        bAdmin := FALSE;
    END IF;

    -- check to see if user is a unit coordinator
    select count (granted_role) into nNumber from user_role_privs where
    granted_role = 'UCOORD';

    IF nNumber > 0 THEN
        bUCoord := TRUE;
    ELSE
        bUCoord := FALSE;
    END IF;

    -- check to see if user is a student
    select count (granted_role) into nNumber from user_role_privs where
    granted_role = 'STUDENT';

    IF nNumber > 0 THEN
        bStudent := TRUE;
    ELSE
        bStudent := FALSE;
    END IF;
```

```

END IF;

IF (bStudent = TRUE) AND (bAdmin = FALSE) AND (bUCoord = FALSE) THEN
-- user only has student role - skip the main form and show the
-- student form
SET_MENU_ITEM_PROPERTY ('menul.student', VISIBLE, PROPERTY_FALSE);
SET_MENU_ITEM_PROPERTY ('menul.admin', VISIBLE, PROPERTY_FALSE);
SET_MENU_ITEM_PROPERTY ('menul.material_management', VISIBLE,
PROPERTY_FALSE);
SET_MENU_ITEM_PROPERTY ('menul.unit_management', VISIBLE,
PROPERTY_FALSE);
SET_WINDOW_PROPERTY ('MAIN_WINDOW', WINDOW_SIZE, 10, 10);
CALL_FORM ('STUDENT_VIEW');
EXIT_FORM;
ELSE
IF (bStudent = FALSE) AND (bAdmin = FALSE) AND (bUCoord = FALSE)
THEN
nNumber := SHOW_ALERT ('NOT_VALID_LOGIN');
EXIT_FORM;
ELSE
mi_id := Find_Menu_Item('menul.student');

IF bStudent = TRUE THEN
SET_MENU_ITEM_PROPERTY (mi_id, VISIBLE, PROPERTY_TRUE);
ELSE
SET_MENU_ITEM_PROPERTY (mi_id, VISIBLE, PROPERTY_FALSE);
END IF;

mi_id := Find_Menu_Item('menul.admin');
IF bAdmin = TRUE THEN
SET_MENU_ITEM_PROPERTY (mi_id, VISIBLE, PROPERTY_TRUE);
ELSE
SET_MENU_ITEM_PROPERTY (mi_id, VISIBLE, PROPERTY_FALSE);
END IF;

mi_id := Find_Menu_Item('menul.material_management');
mi_id2 := Find_Menu_Item('menul.UNIT_MANAGEMENT');
IF bUCoord = TRUE THEN
SET_MENU_ITEM_PROPERTY (mi_id, VISIBLE, PROPERTY_TRUE);
SET_MENU_ITEM_PROPERTY (mi_id2, VISIBLE, PROPERTY_TRUE);
ELSE
SET_MENU_ITEM_PROPERTY (mi_id, VISIBLE, PROPERTY_FALSE);
SET_MENU_ITEM_PROPERTY (mi_id2, VISIBLE, PROPERTY_FALSE);
END IF;
END IF;
END IF;
END;

```

Figure B.17 Code Listing for Hiding Application Menus

B.9.2 Populating the 'Unit List Box' on Student Form

The 'POPULATE_UNIT_LIST' procedure shown in Figure B.18 determines what units a student will see in the 'Units' list box of the 'Material View Console'. Students should only ever see units that they are currently enrolled in.

```

PROCEDURE POPULATE_UNIT_LIST IS

```

```

-- Create a database cursor to populate the list box
CURSOR c11 IS select distinct undel_id, unit_unit_code,
    loctns_location_code, interval_label from units,
    unit_deliveries, course_intervals, enrolments, students
where (student_id = enrolments.stu_student_id) AND (UPPER
(login_name) = UPPER(USER)) AND (interval_id =
coint_interval_id) AND (undel_undel_id = undel_id) order
by unit_unit_code, loctns_location_code;

nIndex NUMBER := 1; -- current list box index
sUnitCode VARCHAR2 (10);
sIntLabel VARCHAR2 (60);
sLocation VARCHAR2 (20);
sLabel VARCHAR2 (100);

BEGIN
    CLEAR_LIST ('unit_list');

    -- ensure that there is always an element in the list (even if it is
    -- blank) otherwise an exception is thrown when the list is selected
    ADD_LIST_ELEMENT ('unit_list', 1, ' ', 0);

    -- loop through the cursor
    FOR clrec IN c11 LOOP
        sUnitCode := clrec.unit_unit_code;
        sIntLabel := clrec.interval_label;
        sLocation := clrec.loctns_location_code;

        sLabel := sUnitCode || ' ' || sIntLabel || ' ' || sLocation;
        ADD_LIST_ELEMENT ('unit_list', nIndex, sLabel, clrec.undel_id);
        nIndex := nIndex + 1;
    END LOOP;

END;
```

Figure B.18 POPULATE_UNIT_LIST Code Listing

B.9.3 Populating the 'Material List Box' on Student Form

When a student selects a unit from the 'Units' list box on the 'Material View Console' form, procedure 'POPULATE_MATERIAL_LIST' determines which materials, related to that unit, the student currently has access to. The procedure relies on the Web server machines clock, the IP address of the client machine, and the identity of the student.

The procedure is listed in Figure B.19.

```
PROCEDURE POPULATE_MATERIAL_LIST IS

    CURSOR c1 IS SELECT material_title, mat_material_id, mat_version,
        mud_id, ipd_ip_number, enr_enr_id, tper_time_period_id,
        start_date, end_date FROM students, materials, enrolments,
        material_access_conditions, material_unit_deliveries WHERE
        (student_id = enrolments.stu_student_id) AND (material_id =
        mat_material_id) AND (version = mat_version) AND (mud_id =
        mud_mud_id) AND (material_unit_deliveries.undel_undel_id =
        :units.unit_list) AND (enr_id = enr_enr_id) AND
        (UPPER(login_name) = UPPER(USER)) ORDER BY mud_id;

    nIndex NUMBER := 1;
    sTitle VARCHAR2 (100);
    sVersion VARCHAR2 (100);
    sLabel VARCHAR2 (100);

    dCurrentDate DATE;
    dCurrentDateTime DATE;

    sCurrentIP VARCHAR2 (100);

    bContinue BOOLEAN;

    dow NUMBER; -- day of week
    hs NUMBER; -- hour start
    he NUMBER; -- hour end

    prevMudId NUMBER := -1;

    nCurrentDayNum NUMBER;

BEGIN

    CLEAR_LIST ('material_list');
    ADD_LIST_ELEMENT ('material_list', 1, ' ', 0);

    -- get current date from the system
    select SYSDATE into dCurrentDate from DUAL;
    select SYSDATE into dCurrentDateTime from DUAL;

    -- get user's IP address
    select ip_addr into sCurrentIP from security where UPPER
(login_name)
                                                    = UPPER(USER);

    -- loop through database cursor
    FOR c1rec IN c1 LOOP
```

```

-- check for start date constraints
IF (clrec.start_date is null) OR (TO_DATE (clrec.start_date) <=
dCurrentDate) THEN
-- passed the start date constraint
-- check end date constraint
IF (clrec.end_date is null) OR (TO_DATE (clrec.end_date) >=
dCurrentDate) THEN
-- passed the start date constraint

IF (clrec.ipd_ip_number is null) OR (sCurrentIP LIKE
(clrec.ipd_ip_number || '%')) THEN
-- passed ip constraint
-- check for day of week constraint

-- check time period constraints
IF (clrec.tper_time_period_id is null) THEN
bContinue := TRUE;
ELSE
select day_of_week, hour_start, hour_end into dow, hs,
he
from time_periods where time_period_id =
clrec.tper_time_period_id;

nCurrentDayNum := TO_NUMBER (TO_CHAR (dCurrentDateTime,
'D'));
IF (dow is null) OR (dow = nCurrentDayNum) OR
((dow = 8) AND ((nCurrentDayNum > 1) AND
(nCurrentDayNum < 7))) OR -- check for weekday
-- condition
((dow = 9) AND ((nCurrentDayNum = 1) OR
(nCurrentDayNum = 7))) -- check for weekend
condition
THEN
bContinue := TRUE;
ELSE
bContinue := FALSE;
END IF;

IF (bContinue = TRUE) AND ((hs is null) OR (hs <=
TO_NUMBER (TO_CHAR (dCurrentDateTime, 'HH24')))) THEN
bContinue := TRUE;
ELSE
bContinue := FALSE;
END IF;

IF (bContinue = TRUE) AND ((he is null) OR (he >
TO_NUMBER (TO_CHAR (dCurrentDateTime, 'HH24')))) THEN
bContinue := TRUE;
ELSE
bContinue := FALSE;
END IF;
END IF;

IF bContinue = TRUE THEN
-- passed all time period constraints

-- passed all constraints, add material to the list
sTitle := clrec.material_title;
sVersion := TO_CHAR (clrec.mat_version);

IF NOT (clrec.mud_id = prevMudID) THEN
sLabel := sTitle || ' (version ' || sVersion || ')';
ADD_LIST_ELEMENT ('material_list', nIndex, sLabel,
clrec.mud_id);
prevMudID := clrec.mud_id;
nIndex := nIndex + 1;
END IF;

END IF; -- time period constraints

```



```
        END IF; -- ip constraint
        END IF; -- end date constraint
    END IF; -- start date constraint
END LOOP;

END;
```

Figure B.19 POPULATE_MATERIAL_LIST Code Listing

B.10 UMS LOGIN Package

This package is called from the Oracle Web Application Server and coordinates the login procedure. The 'UMS_INIT' procedure is the first to be called. The first time it is called it displays Web page asking for the user's login name and password. Once the user has entered these details, they are verified with the Oracle database. If the verification fails then the same Web page comes up waiting for the user to re-enter their details.

If the verification is successful, the function is called again. This time the function writes the client machines IP address and the user's login name into the 'Security' table in the Oracle database. The 'Web Forms' procedure is then called.

Appendix A discussed a 'Static HTML Page' that was required for a Developer 2000 Server application (see Figure A.1). As the Unit Material Management System had to use the Web Application Server to retrieve the user's IP address, this HTML page is generated dynamically. This is performed by the 'Web Forms' procedure. The user's name and password are passed as parameters to Developer 2000 Server. The listing for this package is available in Figure B.20.

```

PACKAGE UMS_LOGIN IS
  PROCEDURE ums_init (p_id          VARCHAR2 DEFAULT NULL,
                     p_password IN VARCHAR2 DEFAULT NULL,
                     p_action   IN VARCHAR2 DEFAULT NULL);
END;

PACKAGE BODY UMS_LOGIN IS

  vdummy      VARCHAR2(1);
  verr_type   VARCHAR2(1) := 'E';
  verr_mess   VARCHAR2(500);
  v_sec_id    security.login_name%TYPE;
  vsession    security.ip_addr%TYPE;

  -- check that the user's login name and password are correct.
  FUNCTION check_valid_user (p_id VARCHAR2, p_password VARCHAR2) RETURN
  BOOLEAN IS
    vCount INTEGER;
    vReturn BOOLEAN;
  BEGIN
    RETURN TRUE;
  END check_valid_user;

  -- This function determines whether or not login id
  -- and password details have been correctly entered.
  -- If the details are correct it displays the initial
  -- startup page for the user.  If the details are
  -- incorrect it does not redirect the page and returns
  -- an error message indicating why.

  FUNCTION redirect_page (p_id IN VARCHAR2,
                        p_password IN VARCHAR2)
    RETURN BOOLEAN IS
    vreturn BOOLEAN := FALSE;
    vremote VARCHAR2(16);
    v_id VARCHAR2(20);
  BEGIN

    IF p_id IS NOT NULL AND p_password IS NOT NULL THEN
      vreturn := check_valid_user (p_id, p_password);
    END IF;

    RETURN(vreturn);

  exception when others then
    htp.p(SQLERRM);
    RETURN(vreturn);

  END redirect_page;

  -- this function coordinates the activation of the Developer 2000
  -- Server portion of the Unit Material Management System
  PROCEDURE Web_forms (p_id VARCHAR2, p_password VARCHAR2) is
    serv_arg VARCHAR2 (200);
  BEGIN
    serv_arg := 'module=INITIAL_FORM_HIDDEN';
    serv_arg := serv_arg || ' userid=' || p_id || '/' || p_password;
    htp.htmlOpen;

```

```
htp.headOpen;
htp.title('Unit Material System');
htp.headClose;
htp.bodyOpen;
htp.line;
htp.header(1,'Unit Material System');
htp.line;

htp.appletOpen ('oracle.forms.uiClient.v1_4.engine.Main', 20, 20,
  'CODEBASE="http://hd19697.fste.ac.cowan.edu.au:9000/Web-code/"
  ARCHIVE="http://hd19697.fste.ac.cowan.edu.au:9000/Web-
  jar/f50all.jar"');

htp.param ('serverPort', 5555);
htp.param ('serverArgs', serv_arg);

htp.appletClose;

htp.paragraph;

htp.p ('Please wait while the Unit Material System loads. This may
  take a few seconds ...<P>');

htp.bodyClose;
htp.htmlClose;

END Web_forms;
```

```

-- main control procedure
PROCEDURE UMS_INIT (p_id          VARCHAR2 DEFAULT NULL,
                   p_password IN VARCHAR2 DEFAULT NULL,
                   p_action     IN VARCHAR2 DEFAULT NULL) IS
  lv_seq_no number;
  vremote VARCHAR2 (16);

BEGIN

  IF NOT redirect_page(p_id,p_password) THEN

    -- display login screen
    http.headOpen;
    http.title('Unit Material System');
    http.headClose;
    http.bodyOpen;
    http.header(1,'Unit Material System');
    http.line;

    http.formopen('ums_login.ums_init',
                 'POST',cattributes=>'NAME="login"');

    http.p('<font size="3">Welcome to the Unit Material System.
           Please enter you user name and password to enter the
           system. Do not attempt to login unless you have been
           assigned a user name.<P>');

    http.tableopen;
    http.tablerowopen;
    http.p('<td width="45%" valign="top" valign="top">');
    http.bold('User name: ');
    http.formtext('p_id','8','8');
    http.p('<br>');
    http.bold('Password: &nbsp;');
    http.formpassword('p_password','8','8');
    http.p('<br>');
    http.formsubmit('p_action','Login');
    http.p('</td>');
    http.p('<td width="55%">');
    http.p('</td>');
    http.tablerowclose;
    http.tableclose;
    http.formclose;
  ELSE
    vremote := owa_util.get_cgi_env('REMOTE_ADDR');
    DELETE from security where login_name = p_id;
    insert into security values (p_id, vremote);
    Web_forms (p_id, p_password);
  END IF;

END UMS_INIT;
END;

```

Figure B.20 UMS_LOGIN Package Listing

Appendix C: Data Definition Language

C.1 Introduction

This appendix provides a listing of the Data Definition Language (DDL) that was used to create the various database objects that the Unit Material Management System was based on.

The execution of the various scripts was controlled by the file 'ums.sql'. To set up the database you must log in to the account that will own the database tables and execute the 'ums.sql' script. You will be prompted for the passwords to the 'sys' account at various stages throughout the script so that various database roles and synonyms can be created.

These scripts were originally generated from Oracle Designer 2000, but were modified manually as the system evolved.

C.2 Main DDL Script

```
/*
  UMS.SQL

  Author: Michael Layng
  Create Date: August 1998

  The main DDL creation script for the Unit Material Management
  System.
  This file calls all other Unit Material Management System scripts.

  User must be logged in to the user account that will own the
  application.
*/

REM Drop all existing Unit Material Management System database objects
@ums.drp

REM Create tables
```

```

@ums.tab

REM Create indices
@ums.ind

REM Create constraints
@ums.con

REM Create sequences
@ums.seq

REM Connect as DBA user SYS.
REM User will be prompted for the password
connect sys

REM Create public synonyms for created objects
@ums.syn

REM Create test user accounts for the system
@ums.usr

REM Create roles
@ums.rol

REM Reconnect as the application owner
REM You will be prompted for the password
connect mlayng

REM Grant access privileges to database objects
@ums.gra

REM Populate tables with test data
@ums.pop

REM END;

```

C.3 Drop All Objects

```

/*
  UMS.DRP

  Author: Michael Layng
  Create Date: August 1998

  This script removes all of the base tables and sequences for the Unit
  Material Management System.

  User must be logged in to the user account that will own the
  application.
*/

drop table material_access_condition;
drop table material_unit_delivery;
drop table material;
drop table ip_domain;
drop table time_period;
drop table enrolment;
drop table student;
drop table unit_delivery;

```

```
drop table unit;
drop table location;
drop table course_interval;
```

```
drop sequence enrolid;
drop sequence intervalid;
drop sequence undelid;
drop sequence matdelid;
drop sequence mataccessid;
drop sequence matid;
```

C.4 Create Tables

```
/*
  UMS.TAB

  Author: Michael Layng
  Create Date: August 1998

  This script creates the base table structure for the Unit
  Material Management System.

  User must be logged in to the user account that will own
  the application.
*/
```

```
PROMPT Creating Table COURSE_INTERVAL
CREATE TABLE course_interval(
  interval_id          NUMBER                NOT NULL,
  start_date          DATE                  NOT NULL,
  end_date            DATE                  NOT NULL,
  interval_label      VARCHAR2(240)        NOT NULL
)
TABLESPACE system;
```

```
PROMPT Creating Table ENROLMENT
CREATE TABLE enrolment(
  enr_id              NUMBER(10,0)         NOT NULL,
  stu_student_id     VARCHAR2(8)          NOT NULL,
  undel_undel_id     NUMBER(10,0)         NOT NULL
)
TABLESPACE system;
```

```
PROMPT Creating Table IP_DOMAIN
CREATE TABLE ip_domain(
  ip_number           VARCHAR2(20)         NOT NULL,
  ip_address          VARCHAR2(30)        NULL,
  domain_name         VARCHAR2(30)        NULL
)
TABLESPACE system;
```

```
PROMPT Creating Table LOCATION
CREATE TABLE location(
  location_code       VARCHAR2(5)         NOT NULL,
  location_name       VARCHAR2(20)        NOT NULL,
  location_country    VARCHAR2(20)        NULL,
  location_address    VARCHAR2(100)       NULL
)
TABLESPACE system;
```

```
PROMPT Creating Table MATERIAL
```



```

CREATE TABLE material(
  material_id          NUMBER          NOT NULL,
  version             NUMBER          NOT NULL,
  material_title      VARCHAR2(40)    NOT NULL,
  material_address    VARCHAR2(60)    NOT NULL,
  creator             VARCHAR2(30)    NULL,
  subject            VARCHAR2(30)    NULL,
  description         VARCHAR2(50)    NULL,
  publisher          VARCHAR2(30)    NULL,
  other_contributor  VARCHAR2(60)    NULL,
  create_date        DATE            NULL,
  material_type      VARCHAR2(10)    NULL,
  source_material    VARCHAR2(60)    NULL,
  language           VARCHAR2(30)    NULL,
  relation           VARCHAR2(60)    NULL,
  coverage           VARCHAR2(60)    NULL,
  rights            VARCHAR2(60)    NULL
)
TABLESPACE system;

```

PROMPT Creating Table MATERIAL_TYPE

```

CREATE TABLE material_type(
  material_type      VARCHAR2(30)    NOT NULL
)
TABLESPACE system;

```

PROMPT Creating Table MATERIAL_LANGUAGE

```

CREATE TABLE material_language(
  language_name     VARCHAR2(20)    NOT NULL
)
TABLESPACE system;

```

PROMPT Creating Table MATERIAL_ACCESS_CONDITION

```

CREATE TABLE material_access_condition(
  mac_id            NUMBER(10,0)    NOT NULL,
  enr_enr_id       NUMBER(10,0)    NULL,
  ipd_ip_number    VARCHAR2(20)    NULL,
  mud_mud_id       NUMBER(10,0)    NOT NULL,
  tper_time_period_id VARCHAR2(20)    NULL,
  start_date       DATE            NULL,
  end_date         DATE            NULL
)
TABLESPACE system;

```

PROMPT Creating Table MATERIAL_UNIT_DELIVERY

```

CREATE TABLE material_unit_delivery(
  mud_id           NUMBER(10,0)    NOT NULL,
  mat_version      NUMBER          NOT NULL,
  mat_material_id  NUMBER          NOT NULL,
  undel_undel_id  NUMBER(10,0)    NOT NULL
)
TABLESPACE system;

```

PROMPT Creating Table STUDENT

```

CREATE TABLE student(
  student_id       VARCHAR2(8)      NOT NULL,
  last_name        VARCHAR2(30)     NOT NULL,
  first_name       VARCHAR2(30)     NOT NULL,
  initials         VARCHAR2(10)     NULL,
  login_name       VARCHAR2(10)     NULL,
  address          VARCHAR2(100)    NULL
)

```

```

)
TABLESPACE system;

PROMPT Creating Table TIME_PERIOD
CREATE TABLE time_period(
  time_period_id          VARCHAR2(20)          NOT NULL,
  day_of_week             NUMBER                NULL,
  hour_start              NUMBER                NULL,
  hour_end                NUMBER                NULL
)
TABLESPACE system;

PROMPT Creating Table UNIT
CREATE TABLE unit(
  unit_code               VARCHAR2(7)          NOT NULL,
  unit_title              VARCHAR2(30)         NOT NULL,
  ucoord_login            VARCHAR2(10)         NOT NULL,
  unit_description        VARCHAR2(100)        NULL
)
TABLESPACE system;

PROMPT Creating Table UNIT_DELIVERY
CREATE TABLE unit_delivery(
  undel_id                NUMBER(10,0)         NOT NULL,
  coint_interval_id       NUMBER                NOT NULL,
  loctns_location_code    VARCHAR2(5)          NOT NULL,
  unit_unit_code          VARCHAR2(7)          NOT NULL
)
TABLESPACE system;

PROMPT Creating Table UNIT_COORDINATOR
CREATE TABLE unit_coordinator(
  login_name              VARCHAR2(20)         NOT NULL,
  full_name               VARCHAR2(50)         NOT NULL
)
TABLESPACE system;

PROMPT Creating Table SECURITY
CREATE TABLE security(
  login_name              VARCHAR2(20)         NOT NULL,
  ip_addr                 VARCHAR2(60)         NULL
)
TABLESPACE system;

```

C.5 Create Indices

```

/*
  UMS.IND

  Author: Michael Layng
  Create Date: August 1998

  This script creates the database indices for the Unit
  Material Management System.

  User must be logged in to the user account that will own
  the application.
*/

```

```

PROMPT Creating Index ENR_STU_FK_I on Table ENROLMENT
CREATE INDEX ENR_STU_FK_I ON ENROLMENT
( stu_student_id )
PCTFREE 10;

PROMPT Creating Index ENR_UNDEL_FK_I on Table ENROLMENTS
CREATE INDEX ENR_UNDEL_FK_I ON ENROLMENT
( undel_undel_id )
PCTFREE 10;

PROMPT Creating Index MAC_ENR_FK_I on Table MATERIAL_ACCESS_CONDITION
CREATE INDEX MAC_ENR_FK_I ON MATERIAL_ACCESS_CONDITION
( enr_enr_id )
PCTFREE 10;

PROMPT Creating Index MAC_IPD_FK_I on Table MATERIAL_ACCESS_CONDITIONS
CREATE INDEX MAC_IPD_FK_I ON MATERIAL_ACCESS_CONDITION
( ipd_ip_number )
PCTFREE 10;

PROMPT Creating Index MAC_MUD_FK_I on Table MATERIAL_ACCESS_CONDITIONS
CREATE INDEX MAC_MUD_FK_I ON MATERIAL_ACCESS_CONDITION
( mud_mud_id )
PCTFREE 10;

PROMPT Creating Index MAC_TPER_FK_I on Table
MATERIAL_ACCESS_CONDITIONS
CREATE INDEX MAC_TPER_FK_I ON MATERIAL_ACCESS_CONDITION
( tper_time_period_id )
PCTFREE 10;

PROMPT Creating Index MUD_MAT_FK_I on Table MATERIAL_UNIT_DELIVERIES
CREATE INDEX MUD_MAT_FK_I ON MATERIAL_UNIT_DELIVERY
( mat_version ,
  mat_material_id )
PCTFREE 10;

PROMPT Creating Index MUD_UNDEL_FK_I on Table MATERIAL_UNIT_DELIVERY
CREATE INDEX MUD_UNDEL_FK_I ON MATERIAL_UNIT_DELIVERY
( undel_undel_id )
PCTFREE 10;

PROMPT
PROMPT Creating Index UNDEL_COINT_FK_I on Table UNIT_DELIVERY
CREATE INDEX UNDEL_COINT_FK_I ON UNIT_DELIVERY
( coint_interval_id )
PCTFREE 10;

PROMPT
PROMPT Creating Index UNDEL_LOCTNS_FK_I on Table UNIT_DELIVERIES
CREATE INDEX UNDEL_LOCTNS_FK_I ON UNIT_DELIVERY
( loctns_location_code )
PCTFREE 10;

PROMPT
PROMPT Creating Index UNDEL_UNIT_FK_I on Table UNIT_DELIVERIES
CREATE INDEX UNDEL_UNIT_FK_I ON UNIT_DELIVERY
( unit_unit_code )
PCTFREE 10;

```

```

CREATE UNIQUE INDEX ensure_unique_unit_offerings on unit_delivery
(loctns_location_code, coint_interval_id, unit_unit_code);

CREATE UNIQUE INDEX ensure_unique_enrolment on enrolment
(undel_undel_id, stu_student_id);

CREATE UNIQUE INDEX ensure_unique_material_offer on
material_unit_delivery (undel_undel_id, mat_material_id, mat_version);

CREATE UNIQUE INDEX ensure_term_label on course_interval
(interval_label);

```

C.6 Create Constraints

```

/*
  UMS.CON

  Author: Michael Layng
  Create Date: August 1998

  This script creates the database constraints for the Unit
  Material Management System.

  User must be logged in to the user account that will own
  the application.
*/

PROMPT Adding PRIMARY Constraint To COURSE_INTERVALS Table

ALTER TABLE COURSE_INTERVAL ADD (
  CONSTRAINT COINT_PK
  PRIMARY KEY (INTERVAL_ID)
USING INDEX
PCTFREE 10)
/

PROMPT Adding PRIMARY Constraint To ENROLMENTS Table

ALTER TABLE ENROLMENT ADD (
  CONSTRAINT ENR_PK
  PRIMARY KEY (ENR_ID)
USING INDEX
PCTFREE 10)
/

PROMPT Adding PRIMARY Constraint To IP_DOMAINS Table

ALTER TABLE IP_DOMAIN ADD (
  CONSTRAINT IPD_PK
  PRIMARY KEY (IP_NUMBER)
USING INDEX
PCTFREE 10)
/

PROMPT Adding PRIMARY Constraint To LOCATION Table

```

```
ALTER TABLE LOCATION ADD (  
    CONSTRAINT LOCTNS_PK  
    PRIMARY KEY (LOCATION_CODE)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To MATERIALS Table

```
ALTER TABLE MATERIAL ADD (  
    CONSTRAINT MAT_PK  
    PRIMARY KEY (VERSION,  
                MATERIAL_ID)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To MATERIAL_TYPE Table

```
ALTER TABLE MATERIAL_TYPE ADD (  
    CONSTRAINT MATTYPE_PK  
    PRIMARY KEY (MATERIAL_TYPE)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To MATERIAL_LANGUAGE Table

```
ALTER TABLE MATERIAL_LANGUAGE ADD (  
    CONSTRAINT MATLAN_PK  
    PRIMARY KEY (LANGUAGE_NAME)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To MATERIAL_ACCESS_CONDITION Table

```
ALTER TABLE MATERIAL_ACCESS_CONDITION ADD (  
    CONSTRAINT MAC_PK  
    PRIMARY KEY (MAC_ID)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To MATERIAL_UNIT_DELIVERIES Table

```
ALTER TABLE MATERIAL_UNIT_DELIVERY ADD (  
    CONSTRAINT MUD_PK  
    PRIMARY KEY (MUD_ID)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To STUDENTS Table

```
ALTER TABLE STUDENT ADD (  
    CONSTRAINT STU_PK  
    PRIMARY KEY (STUDENT_ID)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To TIME_PERIODS Table

```
ALTER TABLE TIME_PERIOD ADD (  
    CONSTRAINT TPER_PK  
    PRIMARY KEY (TIME_PERIOD_ID)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To UNITS Table

```
ALTER TABLE UNIT ADD (  
    CONSTRAINT UNIT_PK  
    PRIMARY KEY (UNIT_CODE)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To UNIT_COORDINATOR Table

```
ALTER TABLE UNIT_COORDINATOR ADD (  
    CONSTRAINT UNIT_COORD_PK  
    PRIMARY KEY (login_name)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding PRIMARY Constraint To UNIT_DELIVERIES Table

```
ALTER TABLE UNIT_DELIVERY ADD (  
    CONSTRAINT UNDEL_PK  
    PRIMARY KEY (UNDEL_ID)  
USING INDEX  
PCTFREE 10)  
/
```

PROMPT Adding FOREIGN Constraint To ENROLMENTS Table

```
ALTER TABLE ENROLMENT ADD (  
    CONSTRAINT ENR_STU_FK  
    FOREIGN KEY (STU_STUDENT_ID)  
    REFERENCES STUDENT (STUDENT_ID)  
)  
/
```

PROMPT Adding FOREIGN Constraint To ENROLMENTS Table

```
ALTER TABLE ENROLMENT ADD (  
    CONSTRAINT ENR_UNDEL_FK  
    FOREIGN KEY (UNDEL_UNDEL_ID)  
    REFERENCES UNIT_DELIVERY (UNDEL_ID)  
)  
/
```

PROMPT Adding FOREIGN Constraint To MATERIAL_ACCESS_CONDITIONS Table

```
ALTER TABLE MATERIAL_ACCESS_CONDITION ADD (  
    CONSTRAINT MAC_ENR_FK  
    FOREIGN KEY (ENR_ENR_ID)  
    REFERENCES ENROLMENT (ENR_ID)
```

```

)
/

PROMPT Adding FOREIGN Constraint To MATERIAL_ACCESS_CONDITIONS Table

ALTER TABLE MATERIAL_ACCESS_CONDITION ADD (
    CONSTRAINT MAC_IPD_FK
    FOREIGN KEY (IPD_IP_NUMBER)
    REFERENCES IP_DOMAIN (IP_NUMBER)
)
/

PROMPT Adding FOREIGN Constraint To MATERIAL_ACCESS_CONDITIONS Table

ALTER TABLE MATERIAL_ACCESS_CONDITION ADD (
    CONSTRAINT MAC_MUD_FK
    FOREIGN KEY (MUD_MUD_ID)
    REFERENCES MATERIAL_UNIT_DELIVERY (MUD_ID)
)
/

PROMPT Adding FOREIGN Constraint To MATERIAL_ACCESS_CONDITIONS Table

ALTER TABLE MATERIAL_ACCESS_CONDITION ADD (
    CONSTRAINT MAC_TPER_FK
    FOREIGN KEY (TPER_TIME_PERIOD_ID)
    REFERENCES TIME_PERIOD (TIME_PERIOD_ID)
)
/

PROMPT Adding FOREIGN Constraint To MATERIAL_UNIT_DELIVERIES Table

ALTER TABLE MATERIAL_UNIT_DELIVERY ADD (
    CONSTRAINT MUD_MAT_FK
    FOREIGN KEY (MAT_VERSION,
                MAT_MATERIAL_ID)
    REFERENCES MATERIAL (VERSION, MATERIAL_ID)
)
/

PROMPT Adding FOREIGN Constraint To MATERIAL_UNIT_DELIVERY Table

ALTER TABLE MATERIAL_UNIT_DELIVERY ADD (
    CONSTRAINT MUD_UNDEL_FK
    FOREIGN KEY (UNDEL_UNDEL_ID)
    REFERENCES UNIT_DELIVERY (UNDEL_ID)
)
/

PROMPT Adding FOREIGN Constraint To UNIT_DELIVERIES Table

ALTER TABLE UNIT_DELIVERY ADD (
    CONSTRAINT UNDEL_COINT_FK
    FOREIGN KEY (COINT_INTERVAL_ID)
    REFERENCES COURSE_INTERVAL (INTERVAL_ID)
)
/

```

PROMPT Adding FOREIGN Constraint To UNIT_DELIVERIES Table

```
ALTER TABLE UNIT_DELIVERY ADD (  
    CONSTRAINT UNDEL_LOCTNS_FK  
    FOREIGN KEY (LOCTNS_LOCATION_CODE)  
    REFERENCES LOCATION (LOCATION_CODE)  
)  
/
```

PROMPT Adding FOREIGN Constraint To UNIT_DELIVERIES Table

```
ALTER TABLE UNIT_DELIVERY ADD (  
    CONSTRAINT UNDEL_UNIT_FK  
    FOREIGN KEY (UNIT_UNIT_CODE)  
    REFERENCES UNIT (UNIT_CODE)  
)  
/
```

PROMPT Adding FOREIGN Constraint To UNIT Table

```
ALTER TABLE UNIT ADD (  
    CONSTRAINT UNIT_UC_FK  
    FOREIGN KEY (login_name)  
    REFERENCES UNIT_COORDINATOR (login_name)  
)  
/
```

C.7 Create Sequences

/*

UMS.IND

Author: Michael Layng
Create Date: August 1998

This script creates the database sequences for the Unit
Material Management System.

User must be logged in to the user account that will own
the application.

*/

```
CREATE SEQUENCE enrolidsq;  
CREATE SEQUENCE intervalidsq;  
CREATE SEQUENCE undelidsq;  
CREATE SEQUENCE matdelidsq;  
CREATE SEQUENCE mataccessidsq;  
CREATE SEQUENCE matidsq;
```


C.8 Create Public Synonyms

```
/*
  UMS.IND

  Author: Michael Layng
  Create Date: August 1998

  This script creates the database sequences for the Unit
  Material Management System.

  User must be logged in as SYS.
*/

/*
  These DROP statement appear here rather than in UMS.DRP as this
  script requires the user to be logged in as SYS.

  Also, these drop statements will not cause any loss of data.
*/
DROP PUBLIC SYNONYM materials;
DROP PUBLIC SYNONYM course_intervals;
DROP PUBLIC SYNONYM locations;
DROP PUBLIC SYNONYM unit_deliveries;
DROP PUBLIC SYNONYM units;
DROP PUBLIC SYNONYM material_unit_deliveries;
DROP PUBLIC SYNONYM material_access_conditions;
DROP PUBLIC SYNONYM ip_domains;
DROP PUBLIC SYNONYM time_periods;
DROP PUBLIC SYNONYM enrolments;
DROP PUBLIC SYNONYM students;
DROP PUBLIC SYNONYM material_types;
DROP PUBLIC SYNONYM material_languages;
DROP PUBLIC SYNONYM unit_coordinators;
DROP PUBLIC SYNONYM security;
DROP PUBLIC SYNONYM enrolid;
DROP PUBLIC SYNONYM intervalid;
DROP PUBLIC SYNONYM undelid;
DROP PUBLIC SYNONYM matdelid;
DROP PUBLIC SYNONYM mataccessid;
DROP PUBLIC SYNONYM matid;

CREATE PUBLIC SYNONYM materials for mlayng.material;
CREATE PUBLIC SYNONYM course_intervals for mlayng.course_interval;
CREATE PUBLIC SYNONYM locations for mlayng.location;
CREATE PUBLIC SYNONYM unit_deliveries for mlayng.unit_delivery;
CREATE PUBLIC SYNONYM units for mlayng.unit;
CREATE PUBLIC SYNONYM material_unit_deliveries for
mlayng.material_unit_delivery;
CREATE PUBLIC SYNONYM material_access_conditions for
mlayng.material_access_condition;
CREATE PUBLIC SYNONYM ip_domains for mlayng.ip_domain;
CREATE PUBLIC SYNONYM time_periods for mlayng.time_period;
CREATE PUBLIC SYNONYM enrolments for mlayng.enrolment;
CREATE PUBLIC SYNONYM students for mlayng.student;
CREATE PUBLIC SYNONYM material_types for mlayng.material_type;
CREATE PUBLIC SYNONYM material_languages for mlayng.material_language;
CREATE PUBLIC SYNONYM unit_coordinators for mlayng.unit_coordinator;
CREATE PUBLIC SYNONYM security for mlayng.security;
```

```

CREATE PUBLIC SYNONYM enrolid for sys.enrolidsq;
CREATE PUBLIC SYNONYM intervalid for sys.intervalidsq;
CREATE PUBLIC SYNONYM undelid for sys.undelidsq;
CREATE PUBLIC SYNONYM matdelid for sys.matdelidsq;
CREATE PUBLIC SYNONYM mataccessid for sys.mataccessidsq;
CREATE PUBLIC SYNONYM matid for sys.matidsq;

```

C.9 Create User Accounts

```

/*
  UMS.USR

  Author: Michael Layng
  Create Date: August 1998

  This script creates test database accounts for the Unit
  Material Management System.  You may wish to alter
  the number of users created or the passwords that they
  are assigned

  User must be logged in as SYS
*/

CREATE USER tsmith IDENTIFIED BY password;
CREATE USER jweston IDENTIFIED BY password;
CREATE USER gprice IDENTIFIED BY password;
CREATE USER cfarquhar IDENTIFIED BY password;
CREATE USER dsmith IDENTIFIED BY password;
CREATE USER jallardyce IDENTIFIED BY password;
CREATE USER rwatts IDENTIFIED BY password;
CREATE USER dsaul IDENTIFIED BY password;
CREATE USER sjones IDENTIFIED BY password;
CREATE USER wsmith IDENTIFIED BY password;
CREATE USER pjones IDENTIFIED BY password;
CREATE USER dbaskin IDENTIFIED BY password;
CREATE USER gandrews IDENTIFIED BY password;
CREATE USER gmillet IDENTIFIED BY password;
CREATE USER sprice IDENTIFIED BY password;
CREATE USER llee IDENTIFIED BY password;
CREATE USER egleeson IDENTIFIED BY password;
CREATE USER jlayman IDENTIFIED BY password;
CREATE USER sandrews IDENTIFIED BY password;
CREATE USER gsmith IDENTIFIED BY password;
CREATE USER jhall IDENTIFIED BY password;
CREATE USER drobbins IDENTIFIED BY password;
CREATE USER ums_admin IDENTIFIED BY password;

```

C.10 Create Database Roles

```
/*
  UMS.ROL

  Author: Michael Layng
  Create Date: August 1998

  This script creates and grants database roles.

  NOTE: User www_user is required for interaction with Oracle
        Web Application Server.

  User must be logged in as SYS.
*/

CREATE ROLE student;
CREATE ROLE ucoord;
CREATE ROLE admin;

GRANT CREATE ANY TABLE TO www_user;
GRANT CREATE ANY TABLE TO student;
GRANT CREATE ANY TABLE TO ucoord;
GRANT CREATE ANY TABLE TO admin;

GRANT student to mlayng;
GRANT student to jweston;
GRANT student to gprice;
GRANT student to cfarquhar;
GRANT student to dsmith;
GRANT student to jallardyce;
GRANT student to rwatts;
GRANT student to dsaul;
GRANT student to sjones;
GRANT student to wsmith;
GRANT student to pjones;
GRANT student to dbaskin;
GRANT student to gandrews;
GRANT student to gmillet;
GRANT student to sprice;
GRANT student to llee;
GRANT student to egleeson;
GRANT student to jlayman;
GRANT student to sandrews;
GRANT student to drobbins;

GRANT ucoord to mlayng;
GRANT ucoord to jhall;

GRANT admin to mlayng;
GRANT admin to ums_admin;
```

C.11 Grant Access Privileges

```
/*
  UMS.GRA

  Author: Michael Layng
  Create Date: August 1998

  This script GRANTs access to the database objects required
  for the Unit Material Management System.

  User must be logged in to the user account that will own
  the application.
*/

GRANT connect, resource to student;
GRANT connect, resource to ucoord;
GRANT connect, resource to admin;

GRANT select on unit_coordinator to student;
GRANT select on unit_coordinator to ucoord;
GRANT all on unit_coordinator to admin;

GRANT select on unit to student;
GRANT select on unit to ucoord;
GRANT all on unit to admin;

GRANT select on unit_delivery to student;
GRANT select on unit_delivery to ucoord;
GRANT all on unit_delivery to admin;

GRANT select on material_unit_delivery to student;
GRANT all on material_unit_delivery to ucoord;
GRANT select on material_unit_delivery to admin;

GRANT select on material_access_condition to student;
GRANT all on material_access_condition to ucoord;
GRANT select on material_access_condition to admin;

GRANT select on time_period to student;
GRANT all on time_period to ucoord;
GRANT select on time_period to admin;

GRANT select on ip_domain to student;
GRANT all on ip_domain to ucoord;
GRANT select on ip_domain to admin;

GRANT select on material to student;
GRANT all on material to ucoord;
GRANT select on material to admin;

GRANT select on location to student;
GRANT select on location to ucoord;
GRANT all on location to admin;

GRANT select on course_interval to student;
GRANT all on course_interval to ucoord;
GRANT select on course_interval to admin;
```

```

GRANT select on material_type to student;
GRANT select on material_type to ucoord;
GRANT all on material_type to admin;

GRANT select on material_language to student;
GRANT select on material_language to ucoord;
GRANT all on material_language to admin;

GRANT select on student to student;
GRANT select on student to ucoord;
GRANT all on student to admin;

GRANT select on enrolment to student;
GRANT all on enrolment to ucoord;
GRANT all on enrolment to admin;

GRANT select on security to student;

```

C.12 Insert Test Data

```

/*
  UMS.POP

  Author: Michael Layng
  Create Date: August 1998

  This script gives a basic skeleton of test data.  Material Access
  Conditions etc should be set via the applications interface to
  Avoid troubles with sequence id numbers.

  User must be logged in to the user account that will own the
  application.
*/

INSERT INTO student VALUES ('0930129', 'LAYNG', 'Michael', 'ML',
'mlayng', '12 Smith St Morley');
INSERT INTO student VALUES ('0952263', 'SMITH', 'Todd', 'TS',
'tsmith', '12 Smith St Morley');
INSERT INTO student VALUES ('0929573', 'WESTON', 'Jodie', 'JW',
'jweston', '12 Smith St Morley');
INSERT INTO student VALUES ('0902039', 'PRICE', 'Gloria', 'GP',
'gprice', '12 Smith St Morley');
INSERT INTO student VALUES ('0972948', 'FARQUHAR', 'Colin', 'CF',
'cfarquhar', '12 Smith St Morley');
INSERT INTO student VALUES ('0955940', 'SMITH', 'Dean', 'DS',
'dsmith', '12 Smith St Morley');
INSERT INTO student VALUES ('0932362', 'ALLARDYCE', 'Jane', 'JA',
'jallardyce', '12 Smith St Morley');
INSERT INTO student VALUES ('0942153', 'WATTS', 'Richard', 'RW',
'rwatts', '12 Smith St Morley');
INSERT INTO student VALUES ('0923426', 'SAUL', 'Daisy', 'DS', 'dsaul',
'12 Smith St Morley');
INSERT INTO student VALUES ('0976940', 'JONES', 'Steve', 'SJ',
'sjones', '12 Smith St Morley');
INSERT INTO student VALUES ('0953295', 'SMITH', 'Wendy', 'WS',
'wsmith', '12 Smith St Morley');

```

```

INSERT INTO student VALUES ('0983623', 'JONES', 'Penny', 'PJ',
'pjones', '12 Smith St Morley');
INSERT INTO student VALUES ('0910984', 'BASKIN', 'Dean', 'DB',
'dbaskin', '12 Smith St Morley');
INSERT INTO student VALUES ('0933426', 'ANDREWS', 'Graeme', 'GA',
'gandrews', '12 Smith St Morley');
INSERT INTO student VALUES ('0931363', 'MILLET', 'Graham', 'GM',
'gmillet', '12 Smith St Morley');
INSERT INTO student VALUES ('0941363', 'PRICE', 'Sandra', 'SP',
'sprice', '12 Smith St Morley');
INSERT INTO student VALUES ('0982313', 'LEE', 'Lyn', 'LL', 'llee', '12
Smith St Morley');
INSERT INTO student VALUES ('0946432', 'GLEESON', 'Emma', 'EG',
'egleeson', '12 Smith St Morley');
INSERT INTO student VALUES ('0900002', 'LAYMAN', 'Joanna', 'JL',
'jlayman', '12 Smith St Morley');
INSERT INTO student VALUES ('0924532', 'ANDREWS', 'Stephen', 'SA',
'sandrews', '12 Smith St Morley');

INSERT INTO course_intervals VALUES (2, '1-jan-1998', '25-feb-1998',
'Summer School 1998 (Aus)');
INSERT INTO course_intervals VALUES (1, '1-jan-1998', '30-jun-1998',
'Semester 1 1998 (Aus)');
INSERT INTO course_intervals VALUES (3, '1-jan-1998', '31-dec-1998',
'Whole Year 1998 (Aus)');
INSERT INTO course_intervals VALUES (4, '1-jul-1998', '31-dec-1998',
'Semester 2 1998 (Aus)');
INSERT INTO course_intervals VALUES (5, '1-jan-1999', '30-jun-1999',
'Semester 1 1999 (Aus)');
INSERT INTO course_intervals VALUES (6, '1-jul-1999', '31-dec-1999',
'Semester 2 1999 (Aus)');

INSERT INTO location VALUES ('BU', 'Bunbury', 'Australia', 'Robertson
Drive Bunbury');
INSERT INTO location VALUES ('ML', 'Mt Lawley', 'Australia', 'Bradford
St Mt Lawley');
INSERT INTO location VALUES ('CH', 'Churchlands', 'Australia', NULL);
INSERT INTO location VALUES ('CL', 'Claremont', 'Australia', NULL);
INSERT INTO location VALUES ('JO', 'Joondalup', 'Australia', NULL);

INSERT INTO unit_coordinator VALUES ('jhall', 'Jean Hall');
INSERT INTO unit_coordinator VALUES ('mlayng', 'Mike Layng');
INSERT INTO unit_coordinator VALUES ('levans', 'Lynn Evans');

INSERT INTO material_type VALUES ('Slides');
INSERT INTO material_type VALUES ('Lecture Notes');
INSERT INTO material_type VALUES ('Sample Exams');
INSERT INTO material_type VALUES ('Assignment Questions');
INSERT INTO material_type VALUES ('Assignment Answers');
INSERT INTO material_type VALUES ('Exercises');

INSERT INTO material_language VALUES ('ENGLISH');
INSERT INTO material_language VALUES ('FRENCH');
INSERT INTO material_language VALUES ('JAPANESE');
INSERT INTO material_language VALUES ('MALAYSIAN');
INSERT INTO material_language VALUES ('INDONESIAN');
INSERT INTO material_language VALUES ('VIETNAMESE');

INSERT INTO time_period VALUES ('Monday 9am - 1pm', 1, 9, 13);
INSERT INTO time_period VALUES ('Monday 9am - 11am', 1, 9, 11);
INSERT INTO time_period VALUES ('Monday 1pm - 4pm', 1, 13, 16);

```

```
INSERT INTO time_period VALUES ('Tuesday 9am - 1pm', 2, 9, 13);
INSERT INTO time_period VALUES ('Tuesday 1pm - 5pm', 2, 13, 17);
INSERT INTO time_period VALUES ('Tuesday 5pm - 9pm', 2, 17, 21);

INSERT INTO ip_domain VALUES ('203.11.69.14', 'www.micromine.com.au',
'Micromine Pty Ltd');
INSERT INTO ip_domain VALUES ('139.230.164.169',
'hd19697.fste.ac.cowan.edu.au', 'ECU Workstation - 13.231');
INSERT INTO ip_domain VALUES ('139.230.164', 'www.cowan.edu.au', 'Mt
Lawley Postgrad Net');
```

Appendix D: Questionnaire

Dear

I would like to thank you very much for your attendance at the demonstration of the Unit Material Management System last Monday (26th October 1998).

I would now like to formally obtain your opinions about the applicability of flexible software in the World Wide Web environment. It would be greatly appreciated if you could take the time to answer the questions below.

Thanks very much,

Michael Layng
Supervisor: Ms Jean Hall

Questionnaire

Please enter your job title within Edith Cowan University _____

Please give a brief overview of your exposure to Web administration and/or unit material management?

Would unit coordinators generally like to have control over the administration of their academic materials or would they prefer that a Webmaster handles this?

Would all unit coordinators (from any faculty) be able to use such a system?

Do you see any problems associated with unit coordinators having this level of control?

How important is it to hide forbidden functionality (e.g. only showing students materials that they currently have access to)?

Please list any other comments you have about the concepts of software flexibility in a Web environment.

Do you give permission to be identified in my thesis? YES/NO

Initials _____

Date _____