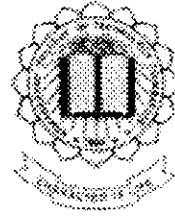


P-1477



INTERACTIVE AUTHORING TOOL

By
SATISH.A
(Reg. No:71202621039)



Of

**KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE**

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

*In partial fulfillment of the requirements
For the award of the degree
Of*

MASTER OF COMPUTER APPLICATION

June 2005

BONAFIDE CERTIFICATE


Certified that this thesis titled **INTERACTIVE AUTHORIZING TOOL** is the bonafide work of Mr. Satish.A, who carried out the research under our supervision. Certified further, that to the best of our knowledge the work reported herein does not form part of thesis or dissertation on the basis of which a degree or award has conferred on an earlier occasion on this or any other candidate.


GUIDE


HEAD OF THE DEPARTMENT

Submitted for the University Examination Held on 24-06-05


Internal Guide


External Guide
24/6/05

ABSTRACT

Interactive Authoring Tool(IAT), which are nothing but Structured Technical Manuals. IAT's were designed and developed by the US Department of Defense to replace traditional paper technical manuals. Their primary purpose is to support diagnostics, maintenance, and repair of complex technical systems. It is a manual that is written for a digital format, allowing it to operate interactively with the user and provide immediate feedback.

These IAT manuals enable a technician to walk through maintenance procedures in a logical sequence and through fault isolation techniques. Current initiatives are underway to create an International Standards Organization (ISO) standard for IAT development. An ISO standard for IATs will ensure that the data can be reusable into the future.

The proposed system is an Integrated database IETIS. Integrated Electronic Technical Information System (IETIS) for Interactive Presentation of Class 4 IETMs integrated in with the data for other processes including Expert-System rules for the display of information and other user-applications such as diagnostics or computer-managed training. Initially the project will focus on a useful software engineering process which will be best suited to the program to be developed. This is the blueprint to be followed when constructing the program. It is likely to be an integration based process, allowing the development of each sub-system and testing of that sub-system before integrating and testing further, until the full system is completed.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without mentioning the names of people who made it possible, whose constant guidance and encouragement crowns all efforts with success.

I would like to express my sincere thanks to **Dr. S.Thangasamy**, HOD, Department of computer Science and Engineering, Kumaraguru College of Technology for having permitted me to undertake this project.

I convey my earnest thanks to **Mr.A.MuthuKumar**, Assistant Professor, Department of Computer Science and Engineering, Kumaraguru College of Technology for his invaluable guidance, support and suggestions throughout the course of this project work.

I express my sincere thanks to the review committee member **Mr.P.GopalaKrishnan** for his suggestions and constructive criticisms.

I express my deep sense of gratitude to **Mr.R.Senthil Kumar**, Software Engineer, Newgen Software Technologies Limited, Chennai for his invaluable guidance, support and suggestions.

TABLE OF CONTENTS

TITLE	PAGE NO.
INTERACTIVE AUTHORIZING TOOL	i
ABSTRACT	iii
TAMIL ABSTRACT	iv
ACKNOWLEDGEMENT	v
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF ABBREVIATIONS	xiii
CHAPTER 1	1
INTRODUCTION	1
1.1 ORGANIZATION PROFILE	1
1.2 INTRODUCTION TO IETM	1
1.3 SYSTEM UTILITIES	3
1.4 PROBLEM DEFINITION	5
1.5 SYSTEM AIMS	5
1.6 MODIFICATION FROM THE PROPOSAL	6
1.7 FORMAT FOR THE REMAINDER OF THE REPORT	7
CHAPTER 2	8
BACKGROUND AND RESEARCH	8
2.1 SEMI-STRUCTURED DATA	8
2.2 EXTENSIBLE MARKUP LANGUAGE (XML)	8
2.3 MYSQL AND JDBC	9
2.4 JDBC	10

2.5 FOP	10
CHAPTER 3	11
SYSTEM DESIGN	11
3.1 SYSTEM DESIGN APPROACHES	12
3.2 REQUIREMENTS DEFINITION	12
3.2.1 Initial Requirements	12
3.2.1.1 Initial Requirements for Manual Editor	12
3.2.1.2 Initial Requirements for XML Storage/Retrieval	13
3.2.1.3 Initial Requirements for XML/XSL/XSL-FO Editor	14
3.3 SUB-SYSTEMS IDENTIFICATION	15
3.3.1 Identification of Sub-systems for Manual Editor	15
3.3.2 Identification of Sub-systems for XML Storage/Retrieval	16
3.3.3 Identification of Sub-systems for XML/XSL/XSL-FO Editor	16
3.4 DATABASE CREATION	16
3.5 FILE RETRIEVAL	16
3.6 FILE DELETION	17
3.7 SYSTEM INTERFACE DESIGN	17
3.8 SYSTEM ARCHITECTURE	17
3.8.1 Manual Editor	18
3.8.2 XML Storage/Retrieval	19
CHAPTER 4	20
SYSTEM IMPLEMENTATION	20
4.1 GENERAL IMPLEMENTATION DECISIONS	20
4.1.1 Database creator	20
4.2 IMPLEMENTATION OF SYSTEM ARCHITECTURE	20
4.2.1 Database implementation	20
4.2.1.1 Database implementation for Manual Editor	21
4.2.1.2 Database implementation for XML Storage/retrieval	23
4.2.1.3 Database implementation for XML/XSL/XSL-FO Editor	24

4.2.2 JDBC Driver	25
4.2.3 Java Applications	26
4.2.4 GUI (Graphical User Interface)	26
4.2.4.1 GUI for Manual Editor	26
4.2.4.2 GUI for XML Storage/Retrieval	26
4.2.4.3 GUI for XML/XSL/XSL-FO Editor	27
4.3 COMPONENT IMPLEMENTATION	27
4.3.1 Component Implementation for Manual Editor	27
4.3.2 Component Implementation for XML Storage/Retrieval	28
4.3.3 Component Implementation for XML/XSL/XSL_FO Editor	30
CHAPTER 5	31
THE SYSTEM IN OPERATION	31
5.1 MANUAL EDITOR	31
5.1.1 Main Option Screen	32
5.1.2 Menu	32
5.1.3 Open	32
5.1.4 Save	32
5.1.5 Close & Exit	33
5.2 XML STORAGE/RETRIEVAL	33
5.2.1 Conversion	33
5.2.2 Main Option Screen	33
5.2.3 XML File Selection Screen	34
5.2.4 XSL-FO File Selection Screen	34
5.4 XML Editor	34
CHAPTER 6	35
SYSTEM TESTING	35
6.1 JUSTIFICATION OF TESTING	35
6.2 TEST CASES	36
6.2.1 Manual Editor test cases	36

6.2.1.1 New Manual component test cases	36
6.2.1.2 Open Manual retrieval component test cases	37
6.2.2 XML Storage/Retrieval test cases	38
6.2.2.1 XML file storage component test cases	38
6.2.2.2 XML file retrieval component test cases	39
6.2.3XML Editor test cases	40
6.2.3.1 Create database component test cases	40
6.2.3.2 XML file component test cases	40
6.2.3.3 XML File retrieval component test cases	41
6.2.3.4 XML File formation component test cases	42
CHAPTER 7	43
SYSTEM EVALUATION	43
7.1 EVALUATION DESIGN AND METHODOLOGY	43
7.1.1 Aspects of the system to be tested	43
7.1.2 Evaluation metrics	43
7.2 EVALUATION SUMMARY	44
CHAPTER 8	45
CONCLUSIONS	45
8.1 REVIEW OF AIMS	45
8.1.1 Provision of a XML creator and store	45
8.1.2 Provision of data retrieval functions	45
8.1.3 Provision of a user interface	45
8.2 FUTURE WORKS	46
8.3 PERSONAL DEVELOPMENT	46
8.3.1 Software design skills	46
8.3.2 Implementation of unfamiliar packages and database	46
APPENDIX	47
1. METHODOLOGY	47
2. PROGRAMME OF WORK	47

3. SCHEDULE OF WORK	48
4. RESOURCES REQUIRED	49
5. SCREEN SHOTS	49
REFERENCES	54

LIST OF TABLES

Table	Description	Page No.
3.1	Initial requirements for Manual Editor	13
3.2	Initial requirements for XML Storage/Retrieval	14
3.3	Initial requirements for XML/XSL/XSL-FO Editor	16
6.1	New Manual component test cases	37
6.2	Open Manual component test cases	38
6.3	XML file storage component test cases	39
6.4	XML file retrieval component test cases	40
6.5	Create database component test cases	41
6.6	Create file component test cases	41
6.7	File retrieval component test cases	42
6.8	File deletion component test cases	43
7.1	Evaluation summary	45

LIST OF FIGURES

Figure	Description	Page No.
3.1	Manual System Diagram	20
3.2	XML Storage/Retrieval Diagram	21
4.1	The fault_isolation table description	23
4.2	The dtd table description	24
4.3	The subsystem_manuals table description	24
4.4	The system_manuals table description	25
4.5	The xslfo table description	25
4.6	The tags table description	26
4.7	The xsl table description	26
4.8	Database Connection Code	27
A.1	Main Option Screen	50
A.2	New Manual Screen	51
A.3	New Manual Screen	51
A.4	Open Manual Screen	52
A.5	Main Option Screen	52
A.6	XML File Selection Screen	53
A.7	XSL-FO File Selection Screen	53
A.8	XSL-FO File Selection Screen	54
A.9	XSL Editor Screen	54

LIST OF ABBREVIATIONS

IETM	-	Interactive Electronic Technical Manual
XSLT	-	Extensible Stylesheet Language Transformation
XSL	-	Extensible Stylesheet Language.
DTD	-	Document Type Definition
JDBC	-	Java Database Connectivity
API	-	Application Program Interface
FOP	-	Formatting Objects Processor
W3C	-	World Wide Web Consortium
XSL-FO	-	Extensible Stylesheet Language- Formatting Objects
UML	-	Unified Modeling Language
XML	-	Extensible Markup Language

CHAPTER 1

INTRODUCTION

1.1 ORGANIZATION PROFILE

Newgen is a young organization founded in 1992 by a technocrat management with over 22 years of experience in software development and marketing. In a short span of eight years, Newgen has achieved a predominant position in the field of Document and Imaging Management Solutions. Newgen has financial participation with Citibank Venture Capital Management and a joint development partnership with Canon Corporation, Japan. Newgen also has a 100% fully owned subsidiary company, Newgen Inc. in Virginia, U.S.A.

1.2 INTRODUCTION TO IAT

Interactive Authoring Tools(IAT), which are nothing but Structured Technical Manuals.IAT is based on IETM concept. IETM manuals enable a technician to walk through maintenance procedures in a logical sequence and through fault isolation techniques.

IETM manuals have been placed into categories or classes as follows.

Class 0. Non-Electronically-Indexed Page Images [Not an ETM] - Systems of Digitized Page Images that are intended for electronic archival filing or Print-on-

Demand. These allow pages to be viewed on an electronic display but have no detailed index for navigation through the document for purposes of on-line usage.

Class 1. Electronically Indexed Page Images - Systems of Digitized Page Images intended for Full-Page Display and use allowing navigation by means of an automated intelligent index to the page images for user access (e.g., Navy AUTOMATIC TERMINAL INFORMATION SERVICE).

Class 2. Electronic Scrolling Documents - Systems for Interactive Display of ASCII encoded Documents using an intelligent index and Hypertext tags inserted into a tagged document file. In general, the document is the result of a simple conversion from a page oriented document but with little reauthoring with the exception of adding hypertext tags. These allow a user to navigate through the document, but have very limited, if any, author inserted navigation aids or a content driven NEXT function.

Class 3. Linear Structured IETMs - Interactive Display of Technical Information which is SGML tagged using MIL-D-87269 tags to the maximum extent possible and using a Hypertext presentation system for display in accordance with MIL-M-87268. It is based on a linear SGML document file and not a hierarchically based Data Base. Navigation is based on author-developed constructs employing prompted dialog boxes and content driven logical NEXT function.

Class 4. Hierarchically Structured IETMs - Interactive Electronic Display of Technical Information specifically authored into and maintained in a non-redundant relational or object-oriented hierarchical database. These source data are subsequently packaged (i.e., "view-packaged") as a run-time database for

Interactive Presentation in accordance with the DoD IETM Specifications (MIL-M-87268, MIL-D-87269, and MILQ- 87270).

Class 5. Integrated Data-Base IETIS - Integrated Electronic Technical Information System (IETIS) for Interactive Presentation of Class 4 IETMs integrated in with the data for other processes including Expert-System rules for the display of information and other user-applications such as diagnostics or computer-managed training. Each of these classes has benefits over the current paper TM Systems and the degree of benefit increases with each higher class. Class 0 and Class 1 ETMs can be built at relatively low cost using scanned images or Postscript encodings, when available, of the present inventory of paper Technical Manuals (i.e., legacy data).

Class 1 system benefits are focused on eliminating problems with the excessive space and weight requirements of paper manuals and the problems of printing paper and maintaining change-page updates. Class 2 systems add the benefits of frame-based electronic presentation to documents which have been developed using a conventional publication system, with the format and content developed according to existing TM specifications. Class 3 is the class in which the TM authoring organization has an opportunity to reorganize for electronic presentation, augments, and converts an existing manual into an IETM data-element form in order to increase technician performance by better access and display of information, as well as, provide the benefits achieved by eliminating paper. Classes 4 and 5 take best advantage of the electronic media because they are specifically authored for and maintained for those media. Class 5 can only be loosely defined at this time; however, whatever it becomes it should be able to use the databases developed for Class 4 with no modification. Class 5 is included in the classification scheme to anticipate various future integrated concepts in which process data are added to static information, and the composite data integrated product demonstrated to achieve performance results better than the other classes of automated TM Presentation systems. Class 5 approaches are expected to

achieve better user performance as well as increase the scope of the IETM by closely integrating additional applications such as a "just-in-time" training, active automated expert-advice, or an other computational diagnostic process performed at display time to enhance the presentation of static Technical Information to the user.

1.3 SYSTEM UTILITIES

The system has the following utilities that are described as follows.

Frame Navigation: Frame Navigation feature enable user with easier frame traversal within an IETM manual. This is not applicable between manuals. Totally, five frame navigation options are provided.

Physical Navigation: Back and Forward buttons will take to the immediate previous and next frame in the manual.

Logical Navigation: Previous and Next buttons will take to the previous and next frames visited from the time of opening the manual.

Go to Frame: User can move to a particular frame by entering the frame number in the Text Box.

History: History will list the frames visited in that manual till that time. User can select a frame from the list and go to that frame. User can also clear the selected items or the entire History list at any point of time.

Notes: Notes are of two types – Public and Private. Public notes will be visible for all users who have rights to access Notes. Only 'Admin' user can modify or delete the Public notes. Private notes will be visible only to the user who added the notes.

Search: Intelligent search can be done on the manual. IAT supports FTS(Full Text Search). FTS is performed for a manual when it is loaded for the first time. Thereafter if the manual is updated using IAT XML, XSL, DTD Editors, again FTS

is performed. The keywords are highlighted when the user views the page found in search dialog.

Bookmark: User can add Bookmark for a frame.

Troubleshooting: Troubleshooting enables user to clarify their queries or any error/problem occurred in the maintenance procedure through Multi-Level Search. By default, the Troubleshooting interface will list some basic search results when opened. User can view the solution by selecting each search result. Two different interfaces are available for troubleshooting in Authoring tool and Viewer.

Troubleshooting – Admin: In authoring tool, user can add new troubleshooting queries and solutions to NGIAT database.

Troubleshooting – Viewer: In viewer, user cannot add new queries and solutions to the database. User has to key in their query in the 'Search Keyword' area. Keywords from the query will be filtered and the search results matching for his queries are listed in the search results area. On selecting a result set, the solution is displayed below. If a search result has relative links, 'See Relative Links' option will get enabled which facilitates Multi-Level Search. User can drill down to the next level using this See Relative Links. User can traverse back using 'Go Previous' option and clear the Troubleshooting display areas with 'Clear' option.

Fault Isolation: This will show a dialog to the user about a specific fault and then provides the action for that specific fault. Then it would ask the user whether the fault has been corrected. If the user option indicates no then it would ask a series of questions and depending upon the user response the solution to the particular problem will be given to the user.

Print: This will show a dialog to the user, whether he wants to see a preview of the current frame or print the current frame or multiple frames. If the user chooses Preview option, a preview window with zoom facilities will be opened and shown to the user. The user can zoom in or zoom out the frame view and he can print that frame. If the user chooses Print option, the current frame will be printed. Users with admin rights or specific Print rights can use this option.

Mute Audio: NGIAT supports Audio and Video files within IETM manual. Any Audio file started from within the manual can be stopped half way using Mute Audio option.

Glossary: This will open an index page for the manual being displayed in the manual display area. If the manual is modified in the Authoring tool, the FTS search will be done again for the entire manual to get the keywords for index page. In Viewer, only the existing keywords in the database for the particular manual will be displayed.

Home: This will take to the home page of an IETM manual being displayed.

Close: This will close the IAT manual.

Exit: This will close the IAT Authoring Tool / Viewer itself.

1.4 PROBLEM DEFINITION

This project proposes to solve this problem using a simple store. Consequently the semi-structured data in the form of XML must be placed into a database. This is in contrast to many of today's XML stores, which simply intake and store XML documents, in their current format. The three major aspects of the system are therefore Manual Editor, XML storage, XML retrieval and XML format conversion with the use of XSL-FO along with the XML, XSL and XSL-FO Editor.

1.5 SYSTEM AIMS

The aim of the system will be to develop a program for efficiently storing and querying semi-structured (XML) data. The system will take XML data, and store the entire XML document in the database table. The database should then allow the system to provide efficient querying of the data held within.

- The system must store the XML documents in a database.
- The system must provide data retrieval functions.
- The system must provide queries to support the data retrieval function.
- The user interface should provide the user with a simple navigation through the possible operations it can perform.
- The system must provide the XML format conversions as per the user needs.
- The system must provide the user with XML, XSL and XSL-FO Editor so that the user can construct the corresponding files.

1.6 MODIFICATION FROM THE PROPOSAL

The system has outgrown the ideas set out in the project proposal. The original idea of using software available for breaking down XML documents into RDF data then into triples, although time saving, provided triples that were of little use to the system when the need to retrieve XML data from the database arose. There was an immense amount of data loss in retrieval of the document-centric XML documents. This was due to the poor quality of the parser, which provided null elements in the triples (i.e. either the subject, predicate or object was a null value). This problem is overcome by storing the entire XML documents in a single column field of the database table so as to prevent the data loss and maintain the integrity of the XML document.

1.7 FORMAT FOR THE REMAINDER OF THE REPORT

The remainder of this report is structured as follows:

- Chapter two provides the relevant background and research into the technical areas the system employs.
- Chapter three discusses the design of the system. It provides the requirements derived from chapters one and two as well as outlining the design method used to create the system.
- Chapter four details the implementation of the system providing discussion of the key implementation decisions.
- Chapter five illustrates the system in operation. This provides an overview of how the system works from the users' perspective.
- Chapter six provides the testing implemented to ensure the usability, reliability and success of the system and its components.
- Chapter seven provides a critical evaluation of the system. The evaluation is based on user feedback, testing and completion of the requirements set out in chapter three.
- Chapter eight concludes the report by revisiting the aims expressed in chapter one. Areas of improvement and future work will also be discussed in the conclusions.

CHAPTER 2

BACKGROUND AND RESEARCH

2.1 SEMI-STRUCTURED DATA

Semi-structured data is data with no associated schema that is loosely defined and can have irregular structure. The data defines itself to give its structure and is therefore called self-defining data. XML is a language that adds more structure to documents and is almost identical to the semi-structured data model. XML is therefore the format of data that will be used in the system.

2.2 EXTENSIBLE MARKUP LANGUAGE (XML)

XML, originally developed for document exchange, is tipped to become the standard language for structured documents and data exchange and the next-generation HTML. Six characteristics of XML that make it so desirable are:

- Tractability: XML is text, not binary, so it can be easily referenced and updated by humans.
- Structure: XML is structured and information can be added to it by adding tags.
- Independence between data and style.
- Extensibility: XML supports the definition of new types of tags.
- Openness: XML is independent of specific vendors.

- XML can be used with Web technologies.

XML, the eXtensible Markup Language, is a subset of the Standard Generalised Markup Language (SGML), designed especially for the Web. It is the universal format for structured documents and data on the Web and facilitates the transaction of data between computer applications. XML was developed by W3C as a standard markup language for adding structure to data i.e. to describe the data. It is this description of data that provides flexibility and adaptability of information identification. Furthermore XML provides self-describing data, a means of storing data within its own structure.

2.3 MYSQL AND JDBC

It is important to the system that the database to be used is both relational and simple yet offered the programmer options advanced enough to allow JDBC to run its more advanced features seamlessly. Relational databases are databases that store data according to the relational model. This relational model relies upon three major components; structure, integrity and manipulation. The structural aspect demands the data to be represented in tables, which contains columns and rows. The integrity aspect demands that the tables follow certain constraints. Finally, the manipulative aspect demands that the tables can be operated upon i.e. certain operations can be performed on them to create new tables. They provide ease of use and analytical flexibility, both qualities demanded by this project. This language allows the managing, querying and updating of relational databases and is the language associated with JDBC. MySql provides adaptability with JDBC, allowing the use of the pure JDBC Bridge.

2.4 JDBC

Java Database Connectivity is an API, enabling Java programs to execute MySQL statements on the database. It is the industry standard for database connectivity. It provides an interface for Java to access databases using the pure JDBC driver. The classes provided facilitate multiple connections thus simultaneous access to many databases. It provides the “Write Once, Run anywhere” ability that all systems strive for.

2.5 FOP

FOP (Formatting Objects Processor) is the world's first print formatter driven by XSL formatting objects (XSL-FO) and the world's first output independent formatter. It is a Java application that reads a formatting object (FO) tree and renders the resulting pages to a specified output. Outputs that are currently supported include PDF, PCL, PS, SVG, XML (area tree representation), Print, AWT, MIF and TXT.

FOP uses the standard XSL-FO file format as input, lays the content out into pages, and then renders it to the requested output. One great advantage to using XSL-FO as input is that XSL-FO is itself an XML file, which means that it can be conveniently created from a variety of sources. The most common method is to convert semantic XML to XSL-FO, using an XSLT transformation. The goals of the Apache XML FOP Project are to deliver an XSL-FO to PDF formatter that is compliant to at least the Basic conformance level described in the W3C Recommendation.

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM DESIGN APPROACHES

The first system used in this project is the system design process. This is built up from the following activities:

- Partition requirements
- Identify sub-systems
- Assign requirements to sub-systems
- Specify sub-system functionality
- Define sub-system interface



This design process encourages a component-based architecture where individual components representing any number of the system functionalities are developed individually of each other. These components are then brought together, component by component, tested through an integrated testing process and finally emerge as the completed system.

3.2 REQUIREMENTS DEFINITION

Requirement engineering is an essential beginning to this design process. It provides the focus for the rest of the design and extends the system aims.

3.2.1 Initial Requirements

The first tables of requirements are initial requirements that were obtained from the initial aims and research, before system design. As the design moves forward, requirements may be added or changed, as may sub-requirements.

3.2.1.1 Initial Requirements for Manual Editor

The table 3.1 explains the initial requirements for the Manual Editor.

Requirement ID	Requirement Description
RM1	The system should provide the list for the available data-modules for every data-module type.
RM2	The system must provide the functionality for adding the data-modules.
RM3	The system should provide the functionality to move the data-modules up and down.
RM4	The system should report the error if the data module is already added.
RM5	The system should report the error if the system with the

	same name already exists in the database.
RM6	The system must have a function to retrieve the data-module files that are stored in the database.
RM7	The interface must provide the user with relevant information regarding what is in the database and the options at each stage.

Table 3.1 Initial requirements for Manual Editor

3.2.1.2 Initial Requirements for XML Storage/Retrieval

The table 3.2 explains the initial requirements for the XML Storage/Retrieval.

Requirement ID	Requirement Description
RX1	Output should be made as usable to the user as possible. This is because there is no way of telling why the user wants the data so screen output is not sufficient.
RX3	Each process carried out by the system must provide feedback to the user to inform them of the current status of the process and if that process is successful or not.
RX4	The system must provide queries on all data in the database i.e. all files queried together.
RX5	The system must provide queries on all single files

	within the database.
RX6	The system must have a function to retrieve whole XML documents to allow the database to be used as storage only.
RX7	The interface must provide the user with relevant information regarding what is in the database and the options at each stage.
RX8	The database must be a relational database containing tables.
RX9	The system must provide security on advanced options so only designated users can use those options.
RX10	A blank database should be able to be created by a user.
RX11	XML documents must be stored in the database.

Table 3.2 Initial requirements for XML Storage/Retrieval

3.2.1.3 Initial Requirements for XML/XSL/XSL-FO Editor

The table 3.3 explains the initial requirements for XML/XSL/XSL-FO Editor.

Requirement ID	Requirement Description
RXX1	The system must take care that all the XML/XSL/XSL-FO documents that are created are well formed.

	documents that are created are well formed.
RXX2	Editor should provide functionalities to create new XML/XSL/XSL-FO file.
RXX3	The contents of the file should be stored in database for processing.
RXX4	Editor should enable the contents to be saved to the database.
RXX5	Functionalities has to be provided to display the list of tags for XML, XSL and XSL-FO.
RXX6	The system must have a function to modify and delete the tags associated with XML, XSL or XSL-FO.
RXX7	Allow to add new template, new attribute & new variable for the tag.
RXX8	Allow the display in the bottom panel to vary according to the tags selected.

Table 3.3 Initial requirements for XML/XSL/XSL-FO Editor.

3.3 SUB-SYSTEMS IDENTIFICATION

3.3.1 Identification of Sub-systems for Manual Editor

- A sub-system must provide a way of creating the database.
- A sub-system for creating a system manual.
- A sub-system for creating a sub-system in which we can add the data-modules.
- A sub-system for adding the data-modules to the sub-systems.

3.3.2 Identification of Sub-systems for XML Storage/Retrieval

- A sub-system that deals with the parsing of documents.
- A sub-system must provide a way of creating the database.
- A sub-system for handling the querying of the database.
- A sub-system for production of full XML documents.
- A sub-system to interact with the user through an interface.
- A sub-system to provide deletion of files from the database.
- A sub-system to change the XML file to other file formats through XSL-FO

3.3.3 Identification of Sub-systems for XML/XSL/XSL-FO Editor

- A sub-system must provide a way of creating the database.
- A sub-system for storing the xml/xsl/xsl-fo files to the database.
- A sub-system for retrieving the xml/xsl/xsl-fo files from the database.
- A sub-system to interact with the user through an interface.

3.4 DATABASE CREATION

This sub-system must provide the following functions.

- Create tables to hold all essential data and indexes.
- Authenticate whether the user is allowed to perform the task.
- Ensure only one database can exist at any one time.

3.5 FILE RETRIEVAL

This sub-system must provide the following functions:

- user defines the name of the file to be returned
- list the files available to the user
- translate into a well-formed XML document

3.6 FILE DELETION

This sub-system must provide the following functions:

- remove all traces of the file from the database
- authenticate whether the user is allowed to perform the task
- maintain the integrity of the database

3.7 SYSTEM INTERFACE DESIGN

This sub-system must provide the following functions:

- provide information to the user on system status.
- provide information to aid the users decision-making.

3.8 SYSTEM ARCHITECTURE

Together the diagrams provide an overview of the complete system architecture and the interaction between the different levels of architecture.

3.8.1 Manual Editor

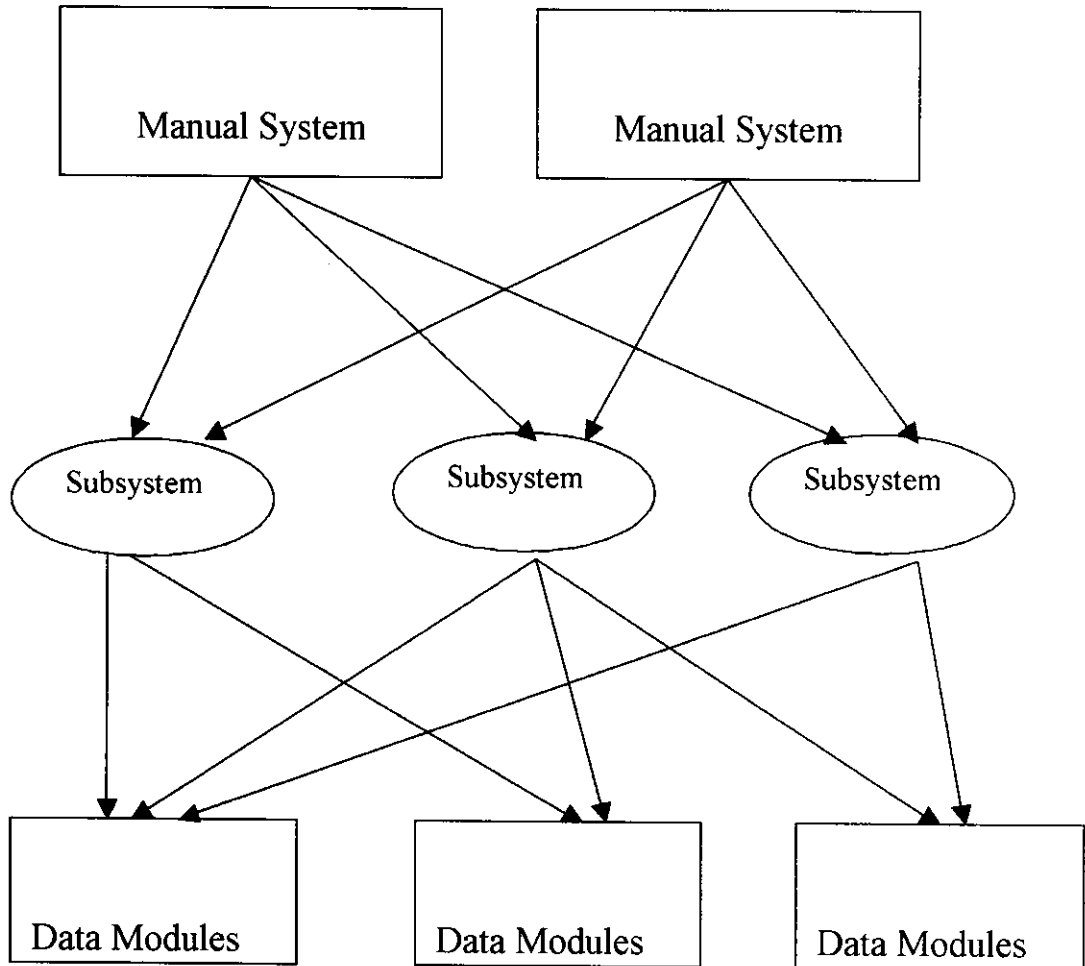


Figure 3.1 Manual System Diagram

3.8.2 XML Storage/Retrieval

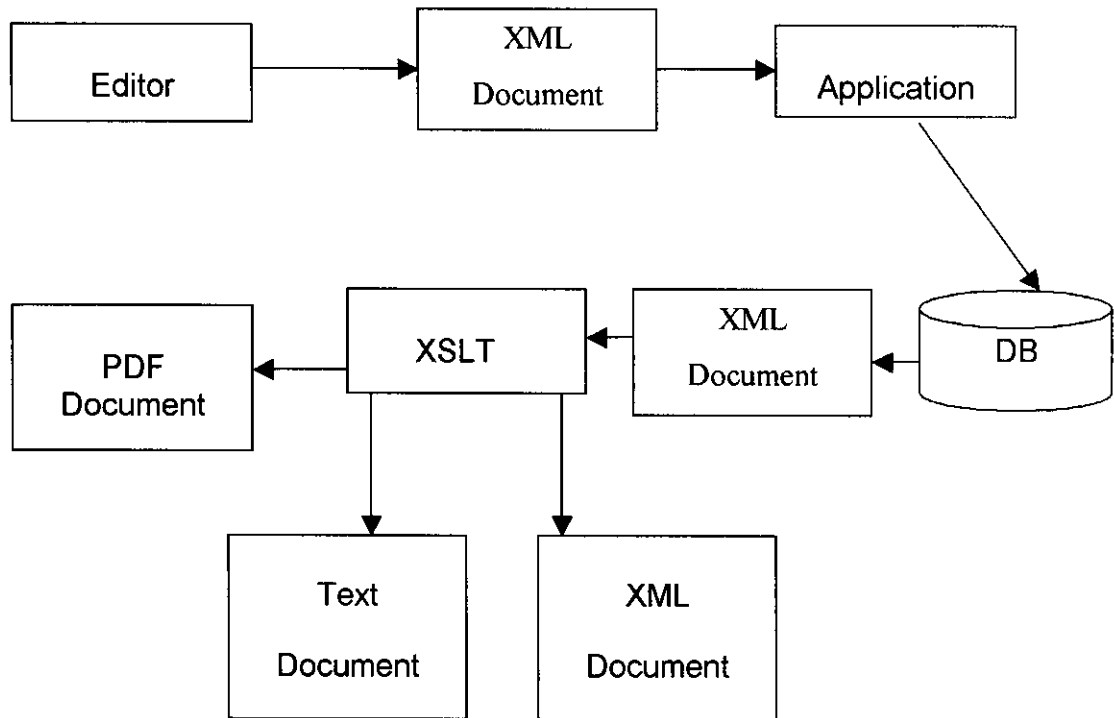


Figure 3.2 XML Storage/Retrieval Diagram

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 GENERAL IMPLEMENTATION DECISIONS

4.1.1 Database creator

This component creates the tables in the database. The tables created can be viewed in figures given below.

4.2 IMPLEMENTATION OF SYSTEM ARCHITECTURE

This section provides an overview of how the system architecture highlighted in figure 3.1 and 3.2 is implemented. Each component of the architecture will be discussed in turn (working from the bottom of the diagram) and the implementation decisions highlighted.

4.2.1 Database implementation

The first decision for the system was concerned with the database. To provide the required functionality it was decided that the package of choice would be MySQL. This provides a relational database that is both simple to use and highly recognized within the database industry. The modeling of the tables was the next implementation decision. Figures 4.1 to 4.7 shows the structure of the tables.

Figure 4.1 shows how the XML documents are stored in an order facilitating retrieval and queries of the data. Figure 4.2 shows the table storing the document header. Figures 4.3 to 4.5 show tables that store the triples of information. Each of these tables represents a different element of the triple. The decision to have nine separate tables holding the data modules XML documents rather than a single table was taken to provide future extensibility to this system.

4.2.1.1 Database implementation for Manual Editor

```
mysql> desc fault_isolation;
```

Field	Type	Null	Key	Default	Extra
DataModuleCode	varchar(50)	YES	NULL		
Content	blob	YES	NULL		

```
2 rows in set (0.03 sec)
```

Figure 4.1 The fault_isolation table description

```
mysql> desc dtd;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| FileName  | varchar(50) |      | PRI |          |       |
| Content   | blob        | YES  |     | NULL    |       |
| IsOpened  | char(1)     | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Figure 4.2 The dtd table description

```
mysql> desc subsystem_manuals;
+-----+-----+-----+-----+-----+-----+
| Field              | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| SubsystemName     | varchar(50) | YES  |     | NULL    |       |
| DataModuleCode    | text        | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.08 sec)
```

Figure 4.3 The subsystem_manuals table description

```
mysql> desc system_manuals;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| SystemName     | varchar(50)   | YES  |     | NULL    |      |
| SubsystemName | varchar(50)   | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.08 sec)
```

Figure 4.4 The system_manuals table description

4.2.1.2 Database implementation for XML Storage/retrieval

```
mysql> desc xslfo;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| FileName  | varchar(25)   |      | PRI |         |      |
| Content   | blob          | YES  |     | NULL    |      |
| IsOpened  | char(1)       | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Figure 4.5 The xslfo table description

```
mysql> desc tags;
+-----+-----+-----+-----+-----+
| Field      | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| type       | enum('xsl','html','xsl-fo') |      |     | xsl      |       |
| tagname    | varchar(35)         |      |     |          |       |
| attributes | text                | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)
```

Figure 4.6 The tags table description

4.2.1.3 Database implementation for XML/XSL/XSL-FO Editor

```
mysql> desc xsl table;
+-----+-----+-----+-----+-----+
| Field      | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| FileName   | varchar(50)         | YES  |     | NULL     |       |
| Content    | blob                | YES  |     | NULL     |       |
| Open_Status | char(1)            | YES  |     | NULL     |       |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Figure 4.7 The xsl table description

4.2.2 JDBC Driver

This provides the link between the database and the Java application requiring communication with the database. It is essential this link be established for any application wishing to talk to the database. This link was established by the following code:

```
try{
    Class.forName ("com.mysql.jdbc.Driver").newInstance ();
    Connection con=DriverManager.getConnection ("jdbc: mysql://
                                                192.168.7.30:3306/ngiat","root","");
}catch (java.lang.ClassNotFoundException e)
{
    System.err.print ("ClassNotFoundException: ");
    System.err.println (e.getMessage ());
}
```

Figure 4.8 Database Connection Code

4.2.3 Java Applications

These applications access the underlying database. They provide the tools for manipulation of the data in the system. Each Class and its implementation are described in full detail in section 4.2.3.

4.2.4 GUI (Graphical User Interface)

4.2.4.1 GUI for Manual Editor

The Class "ManualEditor.java" which creates and edit manuals from data modules. Class "NewManual.java" Provides the interface and functionalities for creating a new system or subsystem. Class "OpenManual.java" which provides the interface and functionalities for opening an existing system or subsystem.

4.2.4.2 GUI for XML Storage/Retrieval

The Class "Xslt.java" provides the XML Editor to facilitate the author to update the XML manual and view the transformed output immediately. Updates can be reverting back since the changes will be temporarily available in memory. DTD elements corresponding to the manual can be newly added and attribute values in XML can be updated. Class "Conversion.java" provides the functionality to perform the XML document conversion to various file formats such as pdf, text, awt or svg. A class "FileSelection.java" performs the file selection for the file conversion.

4.2.4.3 GUI for XML/XSL/XSL-FO Editor

The Class “XMLEditor.java” provides the interface to the user for creating and manipulating the xml/xsl/xsl-fo file.

4.3 COMPONENT IMPLEMENTATION

This section describes how the functions specified in Chapter 3 have been implemented into the working system. Each sub-section will represent one of the components/sub-systems specified in Chapter 3. Each of the functions refers to the summaries of functions in chapter 3.

4.3.1 Component Implementation for Manual Editor

The class “ManualEditor.java” handles the implementation of the functions for the manual editor component. This class creates and edits manuals from data modules. This requires the use of the methods:

- createPanel
- addSubsystems
- listDataModuleTypes
- listSubsystemTypes
- listSystem
- listSubsystem
- modifyData
- saveData

Each of the methods belongs to this class. The method `createPanel()` is for Creating Manual Editor Screen. Method `addDataModules()` is for creating a string containing all data modules added. Method `addSubsystems()` is for creating a string containing all subsystems added. Method `listDataModuleTypes()` is for listing the set of Data modules under a particular subsystem. Method `listSubsystemTypes()` is for listing the set of subsystems under a particular system. Method `listSystem()` is used to open a system and display its subsystems. Method `listSubsystem()` is used to open a subsystem and display its corresponding data modules. Method `modifyData()` is used to check if the contents of system or subsystem are modified. Method `saveData()` is used to save the modifications to database. The Class "NewManual.java" Provides the interface and functionalities for creating a new system or subsystem. This requires the use of the method `createDialog()` that creates a new dialog. Class "OpenManual.java" which provides the interface and functionalities for opening an existing system or subsystem. . This requires the use of the method `openSystem()` that open a system and display its subsystems and a method `openSubsystem()` that open a subsystem and display its corresponding data modules.

4.3.2 Component Implementation for XML Storage/Retrieval

The class "Xslt.java" handles the implementation of the functions for the retrieval of the XML document. It provides the XML Editor to facilitate the author to update the XML manual and view the transformed output immediately. DTD elements corresponding to the manual can be newly added and attribute values in XML can be updated. This requires the use of the methods:

- `loadDtd`
- `loadXsl`
- `loadXml`

- createNewDataModule
- getTextFromEditor
- apply
- doTransform
- getImages
- retrievalImages
- updateDocument
- showFileDialog
- openFile
- findDTDXML
- findXmlIds
- parseAttributes
- findElement
- findMatch
- moveToPage
- savefile

The Class "Conversion.java" provides the functionality to perform the XML document conversion to various file formats such as pdf, text, awt or svg that uses a method createElements() to create the elements. Class "FileSelection.java" performs the file selection for the file conversion that uses a method createElements() to create the elements and another method displayFiles() to display the files.

4.3.3 Component Implementation for XML/XSL/XSL_FO Editor

The class XMLEditor.java provides the functionalities to create and manipulate the xml,xsl and xsl-fo files.This requires a list of functions that include

- formTreeView()
- formElementList()
- placeControls()
- setPopupMenu()

The function formTreeView() is used in order to form a tree structure from the xml,xsl or xsl-fo file. Function formElementList() will list all the tags that are associated with xml,xsl or xsl-fo. Function placeControls() is used to place controls at appropriate location. The function setPopupMenu() provides the popup menus for the user.

CHAPTER 5

THE SYSTEM IN OPERATION



5.1 MANUAL EDITOR

Manual Editor is a user interface that helps the user to view the list of available systems and subsystems. If the user selects system for viewing its detail, the screen will display the list of available system that gets displayed under "Available System" and the list of systems that are needed for a particular module will be displayed under "Added System".

The buttons between the list boxes are provided to move the elements of the system to either side. The top button (>>) is used to add a system to the list of added system. The bottom button (<<) is provided to remove a system from the list of added system. In addition system can be moved to an indicated location. The button on the extreme right is provided to alter the order of occurrences of system. "UP" button is to move the system selected up in the listing and "DOWN" button to move the selected system down in listing. Menu has been provided in order to create new system/subsystem, to open an already existing system/subsystem. Save the modifications made and finally to close the editor.

The basic difference in the interface for system and subsystem is that an additional feature to display the types of data modules has been provided. This can help the user to list the data modules and add the data modules as required.

5.1.1 Main Option Screen

The main option screen is shown in figure 5.1.

5.1.2 Menu

The file menu provided in the manual editor helps to create new system or subsystem. In New menu options have been provided to provide the name of the manual as well as an option to select whether the user is going to create a manual for system or subsystem. The "OK" button will be enabled only if some entries have been provided in the text box given to enter the manual name. If the user is selecting system the manual editor interface will give the listing of all the available subsystem so that the user can select the required subsystems to form their system. And if they are selecting subsystem the manual editor will give the list of data modules for the user to form their subsystem.

5.1.3 Open

The Open option provided under the file menu is used in order to view the list of system or subsystem. In the open menu options have been provided to display the list of available systems. Selection can be made by means of a combo box whether to display system or subsystem. The "OK" button will become active only if some system or subsystem displayed has been selected.

5.1.4 Save

The save option has been provided under the file menu so that the user can save the changes that had been made. The entire contents that are saved gets updated in the database so that the user can access it as and when required. For performing the modifications and making the possible updating in database some

tables are maintained. These tables include system_manuals, subsystem_manuals, datamodule_types, etc.

5.1.5 Close & Exit

The close option can be used to close the manual that was displayed and the Exit option can be used to close the application.

5.2 XML STORAGE/RETRIEVAL

5.2.1 Conversion

In this module we are inputting the XML file and XSL-FO file for converting into particular file format such as PDF and Text using FOP. FOP is the world's first print formatter driven by XSL formatting objects. It is a Java 1.1 application that reads a formatting object tree and then turns it into a PDF document. The formatting object tree, can be in the form of an XML document (output by an XSLT engine like XT or Xalan) or can be passed in memory as a DOM Document or (in the case of XT) SAX events. FOP is part of Apache's XML project.

5.2.2 Main Option Screen

This screen shows the details of the conversion module. The file type combo box holds the file formats such as PDF and Text. File type can be selected from the combo box.

5.2.3 XML File Selection Screen

The XML file text field is there for holding the particular XML file. By clicking the browse button we can have the window that contains list of XML file retrieved from the database. We can select particular XML files from the database.

5.2.4 XSL-FO File Selection Screen

The XSL-FO file text field is there for holding the particular XSL-FO file. By clicking the browse button we can have the window that contains list of XSL-FO file retrieved from the database. We can select particular XSL-FO files from the database.

In the output file we have to specify the newly created output file with particular extension. After that click ok button to have the newly generated file.

5.4 XML Editor

XML Editor screen contains three parts. First part will contain the list of XML tags that can be included. Second part will have the XML coding and the third part reflects coding details. In addition it also provides menu option to users.

CHAPTER 6

SYSTEM TESTING

6.1 JUSTIFICATION OF TESTING

The architecture of the system provides a requirement that the workings of each of the components should not interfere (unexpectedly) with the workings of any of the other components. The testing must therefore implement an integration based testing system where each component is tested as far as possible separately, but then other components are integrated to provide full testing of the component.

6.2 TEST CASES

6.2.1 Manual Editor test cases

6.2.1.1 New Manual component test cases

Test case	Test case description	Test result	Comments
A1	Manual added to empty database	Success	All tables were added to database
A2	Manual added to database containing data	Success	All tables were updated and the relevant data added
A3	The first manual entered is retrieved. Is checked against original manual.	Success	First manual in database is stored correctly.
A4	The second manual entered is retrieved	Success	Manuals at the middle are therefore stored
A5	The last manual in the database is retrieved	Success	Manuals at the end are therefore stored correctly.

Table 6.1 New Manual component test cases

6.2.1.2 Open Manual retrieval component test cases

Test case	Test case description	Test result	Comments
B1	The first manual entered is retrieved. Is checked against original manual.	Success	Manuals are the same as stored in the database.
B2	The second manual entered is retrieved	Success	Again the manuals are the same
B3	The last manual in the database is retrieved	Success	Manuals are the same

Table 6.2 Open Manual component test cases

6.2.2 XML Storage/Retrieval test cases

6.2.2.1 XML file storage component test cases

Test case	Test case description	Test result	Comments
C1	File added to empty database	Success	All tables were added to in the expected way
C2	File added to database containing data	Success	All tables were updated and the relevant data added
C3	The first file entered is retrieved. Is checked against original file.	Success	First file in database is therefore stored correctly
C4	The second file entered is retrieved	Success	Files not at the beginning or end of the database tables are therefore stored correctly
C5	The last file in the database is retrieved	Success	Files at the end are therefore stored correctly.

Table 6.3 XML file storage component test cases

6.2.2.2 XML file retrieval component test cases

Test case	Test case description	Test result	Comments
D1	The first file entered is retrieved. Is checked against original file.	Success	Files are the same.
D2	The second file entered is retrieved	Success	Again the files are the same
D3	The last file in the database is retrieved	Success	Files are the same

Table 6.4 XML file retrieval component test cases

6.2.3 XML Editor test cases

6.2.3.1 Create database component test cases

Test case	Test case description	Test result	Comments
E1	Create a database when there is no database	Success	Database was created with all expected tables
E2	Try to create a database when one already exists with data already entered	Success	No database was created therefore the Singleton pattern works

Table 6.5 Create database component test cases

6.2.3.2 XML file component test cases

Test case	Test case description	Test result	Comments
F1	XML File added to empty database	Success	All tables were added to in the expected way
F2	XML File added to database containing data	Success	All tables were updated and the relevant data added

F3	The first file entered is retrieved. Is checked against original file.	Success	First file in database is therefore stored correctly
F4	The second file entered is retrieved	Success	Files not at the beginning or end of the database tables are therefore stored correctly
F5	The last file in the database is retrieved	Success	Files at the end are therefore stored correctly.

Table 6.6 Create file component test cases

6.2.3.3 XML File retrieval component test cases

Test case	Test case description	Test result	Comments
G1	The first XML file entered is retrieved. Is checked against original file.	Success	Files are the same.
G2	The second file entered is retrieved	Success	Again the files are the same
G3	The last file retrieved	Success	Files are the same

Table 6.7 File retrieval component test cases

6.2.3.4 XML File formation component test cases

Test case	Test case description	Test result	Comments
H1	Check whether the root element is present in the file	Success	Root element is properly inserted into the file
H2	Check whether all the child elements appear in proper order	Success	All the child element appears in proper order
H3	Check whether all the elements opened has been properly closed.	Success	All the elements have proper closing tags.

Table 6.8 File deletion component test cases

CHAPTER 7

SYSTEM EVALUATION

7.1 EVALUATION DESIGN AND METHODOLOGY

7.1.1 Aspects of the system to be tested

The evaluation of system must determine how successful the system is. The criterion is the requirements specified. These requirements overlap into the different sub-system components that consequently mean that the evaluation process will evaluate against each requirement in turn rather than on the component-by-component basis.

7.1.2 Evaluation metrics

The quantification of success for each aspect of the system requires a metric that yields an accurate assessment of the system. As the system aims to successfully achieve each of the requirements, the metric used will provide a rating as to how successful the aspect was in its achievement. A three-tier scale of provides this satisfied, partially satisfied and unsatisfied.

7.2 EVALUATION SUMMARY

The results of the evaluation process are summarized by Table 7.1.

Requirement ID	Evaluation Classification
RX1	Satisfied
RX2, RX11, RM2	Satisfied
RX3, RM4, RM5	Satisfied
RX4, RM1	Satisfied
RX5	Satisfied
RX6, RM6	Satisfied
RX7, RM7	Partially satisfied
RX8	Satisfied
RX9, RX10	Satisfied
User Interface	Satisfied

Table 7.1 Evaluation summary

CHAPTER 8

CONCLUSIONS

8.1 REVIEW OF AIMS

8.1.1 Provision of a XML creator and store

The first aim of the system was to be able to take an XML document and model this data in the database. It has been proven that an XML document can be stored in the database. This component satisfies this initial aim.

8.1.2 Provision of data retrieval functions

This aim was to develop a system to retrieve data. The first through the file retrieval component provided a way to retrieve a document stored in the database. The second is implemented with the query component and is discussed in the following section. These two functions satisfy the aim.

8.1.3 Provision of a user interface

Chapter seven provides a discussion on the interface and its level of success. The aim was to provide a user interface that would provide the user with a means of accessing the system functionality. As described this was a success.

8.2 FUTURE WORKS

Remodeling the design of the XML store was mentioned in chapter two and three. Irrespective of the whole XML document to be stored as a single column element of the table we can further break the XML document. This would decrease the amount of memory required by the database and possibly speed up queries as less data needs to be searched through.

8.3 PERSONAL DEVELOPMENT

8.3.1 Software design skills

The design of the system was crucial to its success and through the use of requirements and design architecture skills were developed in the ability to provide a design solution specific to a task.

8.3.2 Implementation of unfamiliar packages and database

The use of pure JDBC Driver, XML documents, XSL-FO and MySQL database and the interface implementation using Java Swing components were all areas that had previously never been used. Therefore the successful implementation of each of the above provides confidence in the ability to handle unfamiliar languages and software.

APPENDIX

1. METHODOLOGY

The project will use Java programming environment. Java is a very useful modern object-oriented programming language. It is likely the waterfall model (not the original model which was inflexible) of design will be followed throughout the project. Following this model will allow the successful integration of components and sub-systems allowing enough versatility for changes to be made throughout the process.

2. PROGRAMME OF WORK

This section reveals how the project will be conducted in relation to what is to be done and when it is to be done. The project will be developed to completion between December 2004 and the May 2005. The majority of work will be carried out in 20 weeks.

The lists of tasks includes:

- Design and develop the system for storing data
- Test sub-systems
- Integration and testing of the system
- Conformance with design
- Provide user interface

3. SCHEDULE OF WORK

Weeks one to four will see the completion of designs and first prototypes. This will allow testing of the sub-systems to begin. Stages two, three and four are then revisited as needed for weeks five to eleven as the design is developed. Once the sub-systems are considered complete, full system tests will start. This will start as early as week seven although is more likely week fourteen. Allowing again for changes to previous stages, this should be complete by week fifteen. At this point validation and verification are very important. From week seventeen the user interface can be developed which should take only a few weeks.

Throughout the project the documenting and reporting will be carried out. This means every stage will be recorded as it happens hence not allowing events to be forgotten.

Week fourteen will be the first milestone. Progress will be checked at this point. By this point there will be available a prototype of the two programs. It is likely to be limited and not perform all tasks required. Final designs should be documented by this time. Week fifteen is the next milestone and by this point the project should be nearing completion. There should be available a prototype of the full system minus the fully developed user interface.

Week sixteen will be the deadline for the draft report. Week twenty is the deadline for the finalized project including the report. This point must complete everything.

4. RESOURCES REQUIRED

A workstation containing software, which must include Java Runtime Environment 1.4 or above with MySQL 3.23.52 Installation. This is essential, as the programs will be written within this environment.

5. SCREEN SHOTS

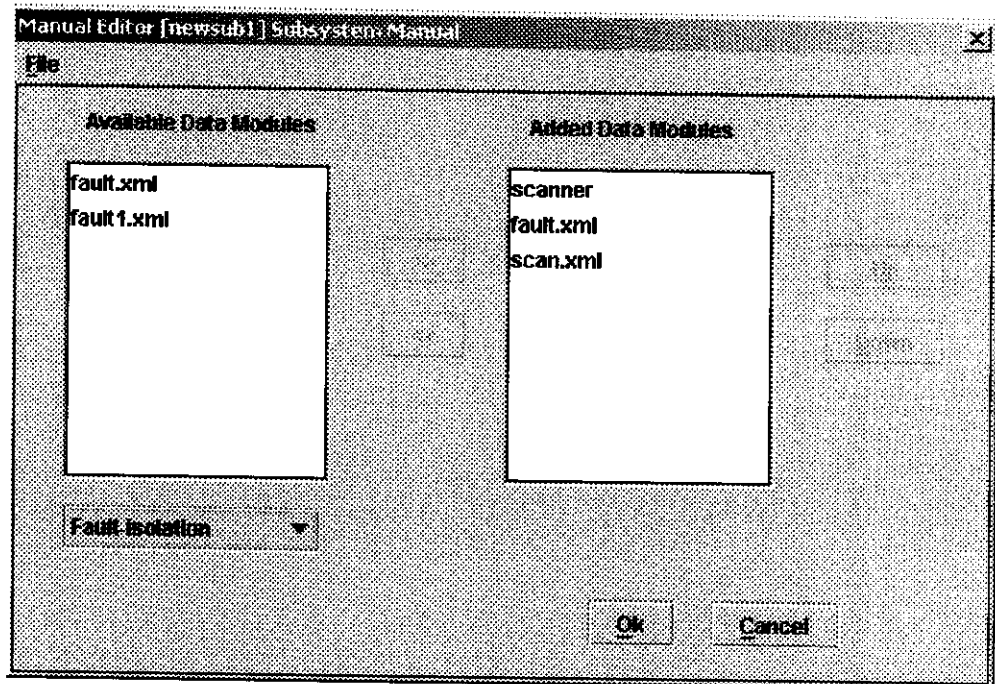


Figure A.1 Main Option Screen

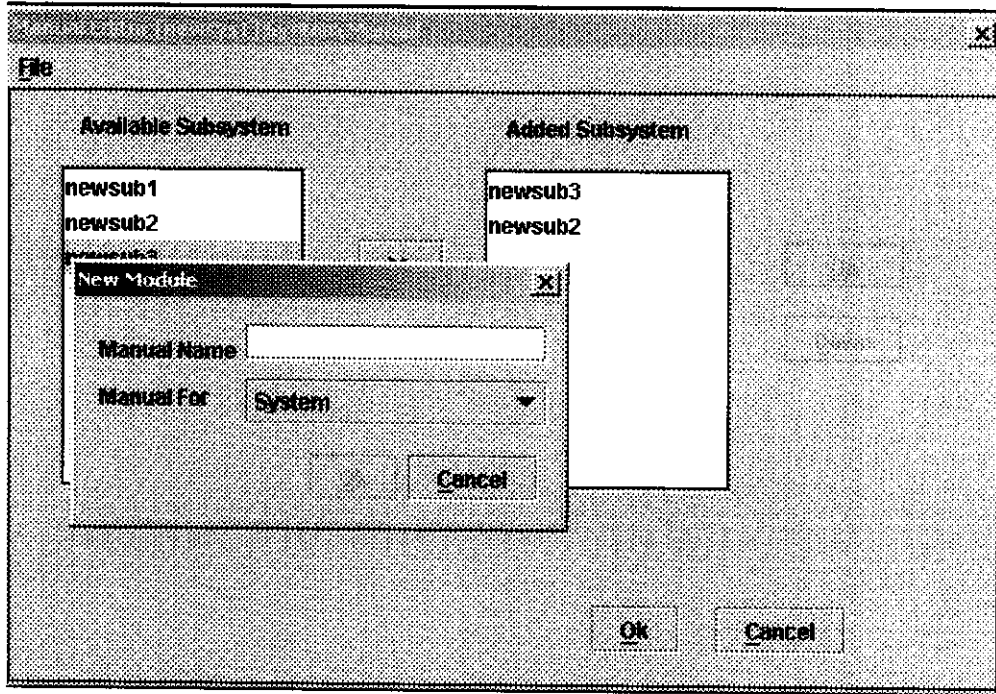


Figure A.2 New Manual Screen

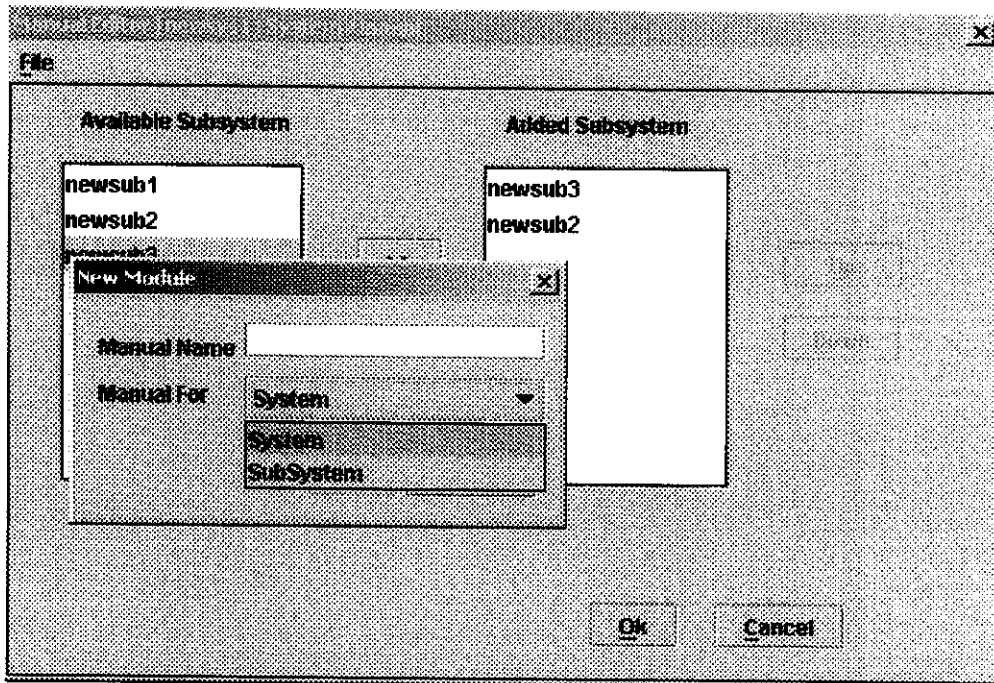


Figure A.3 New Manual Screen

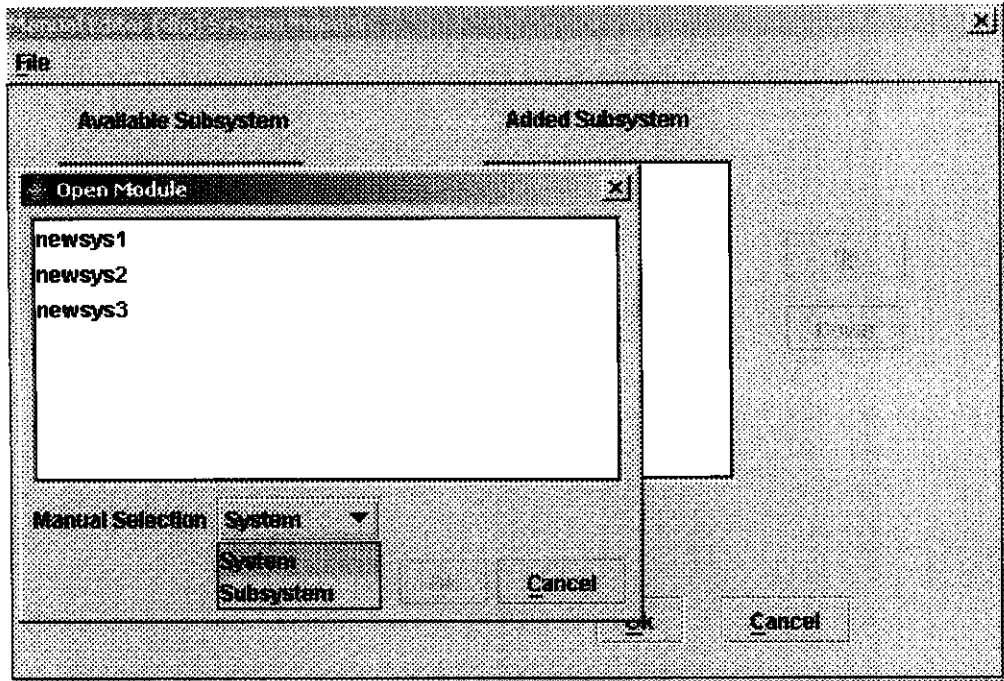


Figure A.4 Open Manual Screen

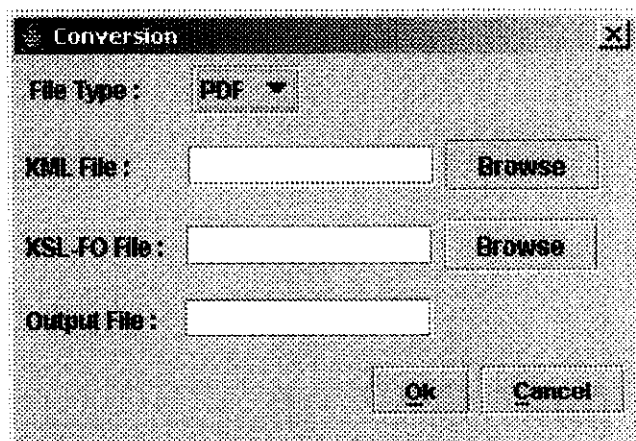


Figure A.5 Main Option Screen

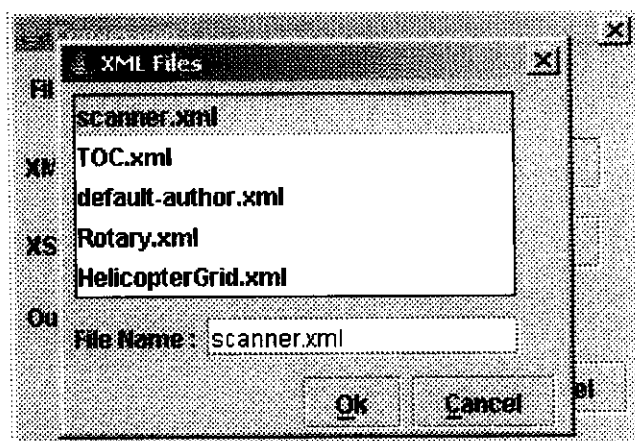


Figure A.6 XML File Selection Screen

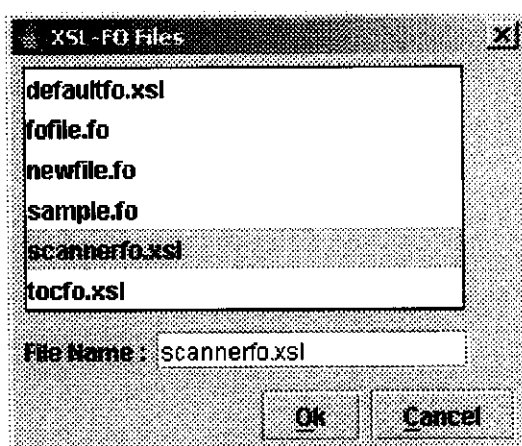


Figure A.7 XSL-FO File Selection Screen

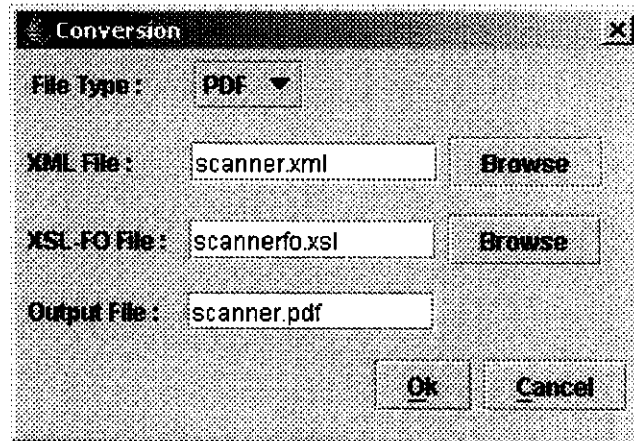


Figure A.8 XSL-FO File Selection Screen

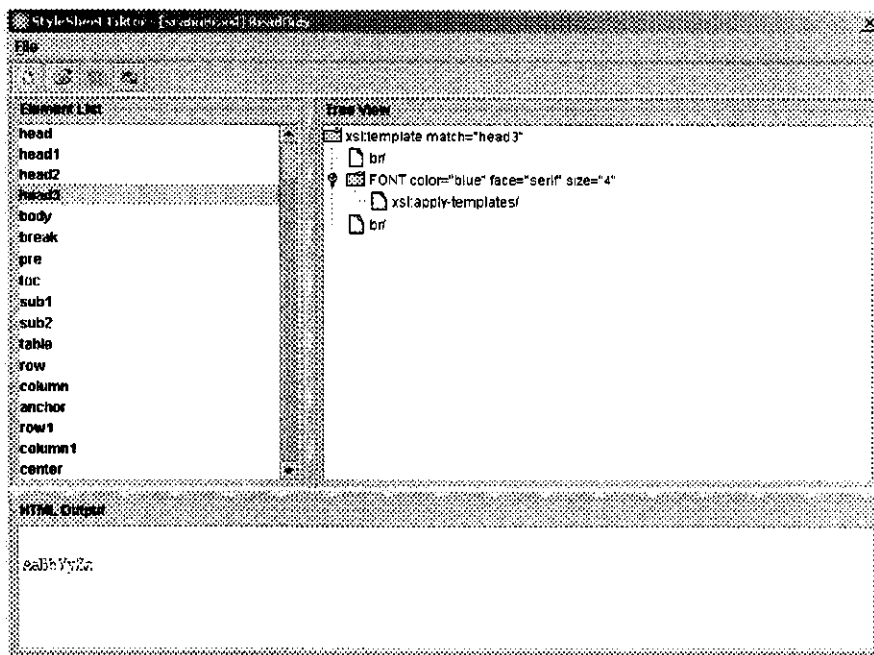


Figure A.9 XSL Editor screen

REFERENCES

1. Cay S. Horstmann and Gray Cornell (2000), 'Core JAVA 2', Pearson Education Asia
2. Patrick Naughton and Herbert Schildt (1999), 'The Complete Reference Java 2', Tata McGraw-Hill
3. V.Yasuo Yamane VNobuyuki Igata Visao Namba (2004), 'High-performance XML Storage/Retrieval System', Tata McGraw-Hill
4. Wen Qiang, Wang Mong, Li Lee, Beng Chin Ooi, Kian-Lee Tan Department of Computer Science, National University of Singapore, 'A Scalable Storage Mapping Scheme for XML Data', <http://www.comp.nus.edu.sg/~wangwq>.
5. <http://www.w3schools.com/xml>
6. <http://www.comp.nus.edu.sg/~wangwq>
7. www.ietm.net
8. www.oneil.com
9. www.cpt.fsu.edu