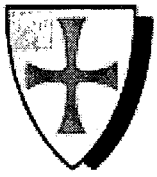
An investigation of web-based hypermedia design support: methods and tools

Kyaw, Phyo

UNIVERSITY OF DURHAM



DEPARTMENT OF COMPUTER SCIENCE

An Investigation of Web-Based Hypermedia Design Support: Methods and Tools

Phyo Kyaw

MSc Thesis

September 1998

ABSTRACT

Since the Internet networking was first established, the World Wide Web (or WWW) provides a new opportunity to deliver information and to communicate with others. Therefore, many organisations and industries have joined this exciting technology to take advantage of the Web. In recent years, the opportunity has arisen for other tasks to be carried out on the Web apart from delivering information. As the Web applications and documents have become larger and more complex, they have experienced many design and development problems which often lead to very high maintenance cost. To improve the quality of Web sites and the structure of information, the designers need structured design methods, guidelines, and tools to assist their work. Some researchers have proposed hypermedia design methods and guidelines, which contain development cycle with formal design techniques to assist the construction of Web page designs.

To overcome the design and development problems, this research is carried out by surveying currently available design methods. It shows the ways to apply these methods for developing structured Web sites. The results of this research led to identifying the design stages involved in developing Web sites using hypermedia methods. It also presents a CASE tool to provide a development environment for producing Web pages based on hypermedia design stages. This encourages Web designers to apply structured hypermedia design methods to improve the quality of design and to reduce the maintenance cost. The thesis is relevant for end-users, Web designers from organisations, industries, and institutes for those who want to apply structured hypermedia design methods for producing their Web documents.

Copyright

Declaration

No part of the material offered has previously been submitted by the author for a degree in the University of Durham or in any other University. All the work presented here is the sole work of the author and no one else.

ACKNOWLEDGEMENTS

I am very grateful to my supervisor Dr. Cornelia Boldyreff for helping me all the way and for moral support and to all the members of Department of Computer Science for the facilities provided. I also would like to thank to my parents and my family for encouraging me for this M.Sc. course and for providing me with everything.

CONTENTS

Table of Figures

List of Tables

1. Introduction

1.1 Chapter Abstract

The main objective of this chapter is to introduce the project research area and the scope of the project. In the chapter, a brief description of the feasibility study, the purpose of the research, and an outline for the rest of the thesis are identified. In the feasibility study, the description of research methods, requirements, proposed research fields, and scope of the project are also presented. After the research area has been outlined, the main objectives of the research are addressed and followed by the criteria for success of the project.

1.2 Introduction and background

In recent years, World Wide Web (also known as WWW or the Web) has become a critical issue for every organisation. It is considered a primary medium for success and can provide rich and clear information. With the help of navigational features of hypertext and multimedia, WWW has become a more effective medium than traditional information storage (i.e. printed books, etc.). Therefore, many individual users, businesses, and institutions have rushed into the WWW. However, most of the orgainsations have produced their Web based applications and information systems without structured and efficient design. They omitted the user needs and application design requirements. As a result, the development of WWW applications has become unstructured and incomplete, and the users who browse these Web documents have often headed to 'lost in hyperspace' (Morris and Hinrichs, 1996). Accordingly, careful planning, design, standards and guidelines are essential infrastructure for developing WWW applications.

The WWW is based on hypermedia technology. Hypermedia lets the information on the WWW be accessed, navigated and shared quickly and efficiently and thus hypermedia is a crucial element of the WWW applications. Consequently, the design and development of hypermedia on the WWW applications becomes important.

1.3 Feasibility study of research area

The main objective of the feasibility study is to introduce the project research area and analyse the requirements for research in this area. The research area includes the detailed review and evaluation of the existing structured design methods for WWW applications. The research will be focused on current hypermedia, specifically hypermedia design methods, existing tools and evaluation of their strengths and weaknesses. A review will be made using the usability criteria, and assessing their support for project management of Web-based hypermedia applications. A clear comparison of the development of Web-based hypermedia applications and traditional software engineering approach will be made.

Research will be focused on user interface design and user-oriented development for Web-based hypertext, hypermedia and multimedia applications. As technology becomes more advanced, for the development of the WWW, different tools and browsers have emerged from various software organisations such as Netscape Navigator, MS Frontpage, Macromedia and Backstage studio. Together with different tools, different standards and languages also rapidly emerged from different sources such as Hypertext Markup Language (HTML), ActiveX, Java applets, Javascripts, CGI, etc. With these tools and languages, developers can perform many different tasks easily. However, these tools and languages are mainly for interface design and implementation, and are unconcerned with the standard development stages, project management, and design methods to assist the designers to produce high quality Web documents during development.

The extensive use of different tools and languages on the Web may create problems in future years if the resultant Web applications are not well designed and developed. Therefore, there is a need to formalise, and to develop a series of engineering based stages, (Bieber and Vitali, 1997) CASE tools, and guidelines for the Web and Web application management. From the designers' point of view, they need to obtain support from these structured development methods and tools, which is one of the most important factors to produce Web based hypermedia applications with lower maintenance costs. Accordingly, the research will also be focused on the development of WWW applications using different standard development tools together with currently available structured hypermedia CASE tools. The detailed description of the existing hypermedia design methods will also be analysed later in this thesis. However it is impossible to analyse every detailed presentation of individual information objects to fit with every type of application. Therefore the research will take a broad view of information structuring using hypermedia design methods during the development of applications and it will omit the implementation details of each application

components. Furthermore, the purpose of this research includes the analysis of existing CASE tools that are proposed by various researchers in recent years. The research will be also focused on different features of these tools, including:

- How these CASE tools can handle linking and navigation for documents,
- Classes of authors (i.e. Professional developers, Web page authors, etc.),
- How can they support relationship management,
- And how they can integrate with organisational structures for large scale Web sites.
- And finally, the development of a CASE tool to support for hypermedia design methods.

The target of this research also includes the identification of an overall approach to develop Web-based hypermedia applications using more formal design methods. The following section summarises the main objectives of the research.

1.4 The objectives of the Research

In a summary, the objectives of the research include the following:

- The analysis of existing hypermedia design methods and their related issues, including their development stages and other implications.
- The evaluation of these existing methods by applying them to the development new applications (forward engineering) and to redesign the existing applications (reverse engineering).
- The identification of the basic design and development stages that are required to construct Web-based hypermedia applications using model based hypermedia design methods.
- The development of a prototype of a CASE tool to support the design and construction of Web-based applications based a basic model, which includes predefined design stages that are derived from existing hypermedia design methods.

- The evaluation and trial usage of the CASE tool by developing structured Web documents.

1.5 Criteria for Success

At the end of the research, the following criteria need to be achieved:

- Understanding of hypermedia design methods and Web based hypermedia applications will be gained.

- An approach to develop Web applications based on the existing hypermedia design methods will be identified.

- A CASE Tool that allows the user to design and construct Web based applications based on hypermedia design methods will be built.

- An Evaluation of the CASE tool will be made.

1.6 Organisation of Thesis

The second chapter describes the background of the Web and its applications. It also covers hypermedia features and illustrates how hypermedia design methods are related to the development of Web applications. This chapter also surveys on existing hypermedia design methods and shows how they are derived from software engineering approaches.

The third chapter discusses the case study based approach to evaluating different hypermedia design methods.

The fourth chapter identifies the design of a new CASE tool to be built and its requirements including design environments and conceptual and navigational design supports.

The fifth chapter focuses on the implementation of the CASE tool and shows a sample application using the tool.

The sixth chapter describes the outcomes on the CASE tool during the trial usage and evaluates the results. It also includes a comparison between the tool that has been implemented with another existing tool.

Finally, the seventh chapter concludes with the overall evaluation of the research and a discussion of further work.

1.7 Summary of Main Points

In this chapter, we have identified the problem area and the scope of the project. It also includes a brief description of how hypermedia design methods can be applied to develop and structure Web based hypermedia documents. It has also pointed out existing problems and how these problems are related to the scope of the research. Finally it outlined the main objectives of the research and criteria for success.

2. A Survey of Hypermedia Design Methods

2.1 Chapter Abstract

The use of the World Wide Web and its extended features from hypertext technology give a variety of benefits ranging from marketing to computer based training programmes. Many organisations have been developing Web based applications for a variety of purposes, but these Web documents are created mainly by a few Web masters in an *ad-hoc* approach. At the same time, many researchers have suggested structured hypermedia design methods and guidelines to assist the development of Web applications. This chapter surveys and presents various hypermedia design methods and indicates how they can assist the designers for the development of Web applications. It also discusses how these design methods are related and derived from one another, and the way they are related to conventional software engineering methods. Furthermore, it describes three methods in detail, namely, RMM (Isakowitz, Strohr, 1995), HDM (Garzotto, Paolini, 1993) and OOHDM (Schwabe, Rossi, 1996). It assesses the implications of applying these design methods for the development of large and complex Web applications. and proposes a set of criteria to evaluate different methods to reveal both the benefits and limitations of design methods or models. These descriptions are relevant for the Web designers and users from various organisations who are considering using structured design methods for the development of Web applications.

2.2 Introduction

In recent years, WWW has become one of the critical issues for every organisation as it is one of the primary mediums for successful communication consequently. The development of most Web based hypermedia applications are of primary importance to every organisation that wants to take advantage of the Web. It can provide rich and clear information and give users direct access to the content of the application. With the help of navigational features of hypertext and multimedia, WWW becomes a more effective medium in comparison with traditional information storage. Therefore many individual users, businesses, and institutions have rushed into the WWW. However most organisations have produced WWW applications and information systems without any structured and efficient design. They omitted the application design requirements. As a result, WWW applications have become unstructured

and are causing maintenance problems. Later in this chapter, we will be looking at how hypermedia design methods can assist developers to implement maintainable Web applications effectively.

The scope of WWW applications can be varied for many different purposes.They are also intended to interact with a variety of users at different levels of skills, however, the growing amount of data and links within the Web documents have become increasingly complex and hard to maintain. Furthermore, many WWW applications that have been developed are mainly in an ad-hoc manner without any structured design. Accordingly, careful planning, design, standards and guidelines are essential infrastructure for the development of WWW based hypermedia applications (Bieber and Vitali, 1997).

Recently, developers and researchers have proposed specific hypermedia design methods. They include HDM (Garzotto, Paolini, 1993), RMM (Isakowitz, Strohr, 1995), Extended RMM (Isakowitz, Kamis, 1997), OOHDM (Schwabe, Rossi, 1995), and OOHM (Hendrikx, Olivié, 1997), etc., of which three of the core methods will be fully evaluated later in this chapter. The authors of these methods have applied some of their techniques and approaches from their past experiences of software engineering and other techniques such as Object Oriented approach and Database design techniques (McFadden and Hoffer, 1994). However, the development of Web based applications can be differentiated from traditional software projects. The use of specific design methods can provide structured guidelines for the developers, which leads to better Web sites. We will describe how these methods are related to Web application development and evaluate what features they can contribute to the development of maintainable and structured Web sites.

2.3 Background of Hypermedia and the World Wide Web

All the Web applications use hypermedia features. Hypermedia can be defined as a mechanism to manage complex information by linking nodes of data. In other words, data stored in a network of nodes are connected by links, and allows people to share information from a variety of sources.

Hypermedia not only supports conventional text but also includes different types of media objects such as graphics, audio, video, animation and programs. The combination of hypertext and multimedia, known as 'Hypermedia', provides flexible access to large and complex information stores by integrating features of navigation, annotation, user interface, cross referencing, links, and nodes.

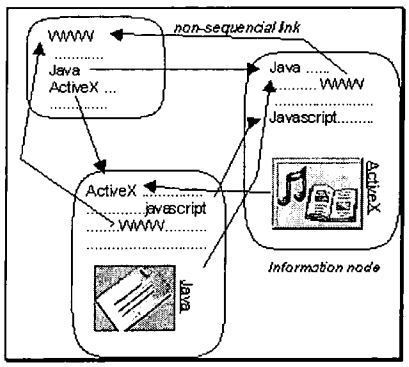


Figure 1 - Example of Web Nodes and Links

As shown in Figure 1, information nodes are linked together as a semantic network. Each node represents a single concept or topic of information, which may include different forms of media items. Links can be made from one document to another or from one part to another part in the same document. This is done by adding embedded links (a piece of text, icon, or picture. etc.) within the contents to allow the user to reach nodes with more detailed or simply related information. Links can be bi-directional which allows backward cross-referencing. These links and nodes allow authors to efficiently construct the whole context. In a hypertext system, a basic tool for collaboration is the use of annotations. Examples of hypertext-like systems and applications are Microsoft® Windows Help System, Multimedia encyclopaedias and the most fundamentally, the World Wide Web.

The use of links and nodes avoids traditional linear structure and leads to a more structured presentation. Moreover, authors can present the same information in many different ways to suit different users. On the other hand, the users can access information in the most effective way by navigating through associated information but, excessive and unstructured inter-linking can lead to user disorientation and a user being 'Lost in hyperspace' can be observed (Morris and Hinrichs, 1996). Therefore, authors should provide their hypermedia applications with orientation and navigation features to free the users from disorientation, cognitive overheads and other constraints. To provide these supports authors should include:

- Global overview diagrams for the entire site.
- Navigational assistants such as indices and guided tours, minimal links to only relevant information within the context.
- For dynamic features, all the diagrams and links should be dynamically updated based on the navigation made by the user.
- The use of formal hypermedia design methods to set the information structure.

The background idea of hypertext has existed for a very long time in books, dictionaries and encyclopaedias but it was only introduced by Bush in his 'memex' system in 1945 (Bush, 1945). Memex was designed to store and record books by using microfilms as inputs. Although the system was not implemented at that time, his idea is still in use today. In recent years, many organisations and publishers have transferred from traditional printed text systems to hypermedia information systems in order to take the full advantage of the Web. The development of training programs, encyclopaedias, dictionaries and educational manuals are well suited for hypermedia information systems. These applications can offer a variety of benefits through the use of hypermedia features. Some of the advantages of hypermedia over printed text are:

- Hypertext can support non-sequential access whereas printed text is sequential in nature.

- When comparing the storage, the amount of information that can be stored on electronic media is far greater than in printed books.

- Information can be instantly searched by indexing keywords or browsing through hypermedia systems whereas users have to search through printed books, or use published indices, which are fixed and infrequently updated.

- Cross-referencing is limited in printed text where as the main objective of the hypermedia is to provide flexible navigating via cross-referencing of information.

- Multimedia, including graphics, sounds, animations, 3D objects and programs, can be applied by using hypermedia where printed text is limited by text and picture.

- Information on hypermedia systems can be easily modify or updated.

The trend for conventional publishers to move towards electronic publication of their material is largely so that these advantages can be realised.

2.4 Web application development and Hypermedia design methods

Web applications are varied in scale from personal home pages to major Web sites such as agovernment Web-site, company product information, etc. The development of large scaled Web sites requires structured hypermedia design together with other standards and programs such as CGI scripting, multimedia presentation, and information retrieval from distributed

database links. Most importantly, it requires skillful professional authors to produce a robust design. Authors or developers should not only be able to apply structured design methods but also be capable of constructing the design itself. This means that the authors should be able to evaluate different methods and choose the most appropriate one for performing any specific design activity. The use of structured methods assists the authors in many ways during the development of Web applications. Therefore if a hypermedia application is developed using a design method rather than by taking an ad-hoc approach, the following benefits are likely:

- The authors are able to construct large and dynamic systems by applying current Software Engineering design methods.

- The use of well-defined steps in design methods assists the authors by guiding them through the project. It also separates the fields of development (i.e. separation of conceptual, navigation and user interface, etc.).

- The supported techniques used in design methods (such as object-oriented, relationship management, etc.) can assist the authors in solving large and complex design problems in various application domains.

- The re-usability of design diagrams leads to the presentation of interface design in various perspectives for one conceptual design.

- The use of CASE tools supported by some design methods supports developers in the design phase more effectively.

- The openness of the product constructed using a design method can be implemented in various ways (i.e. HTML, ToolBook, or Macromedia Director, etc.).

However, there are also limitations by applying these methods. These are listed below :

- The complexity of design methods requires not only skilled developers but also those with a software engineering background. This means that the authors must understand various design techniques including database design, object oriented design, navigational design, etc.

- Most of the current design methods that are proposed mainly produce stand-alone, retrieval oriented hypermedia application (Bieber and Vitali, 1997). Therefore they are limited to centralised database oriented applications.

- In comparison with an ad-hoc approach, applying design methods and models requires much more time and effort in both design and phase.

- Since hypermedia design methods are still a research area, these is very little automated support for CASE tools based on these methods.

2.5 *Traditional Software Applications vs Hypermedia Applications*

Hypermedia development projects involve people from various departments with different skills. They include not only programmers but also authors, designers, artists, musicians, video editors, etc. However, traditional software applications do not necessarily require various people from various departments. When designing traditional software, developers have to deal only with specific tasks whereas a hypermedia application developer may be required to deal with a variety of tasks including the design of the foundation of the whole organizational structure (Isakowitz, Strohr, 1995). Designers of hypermedia systems have to breakdown large and complex application domains into clear, small and accessible related domains that can be presented to end users. For example, to create a large WWW site for a university, designers have to breakdown the university into various departments and again into smaller departmental modules and courses. The development of traditional software applications often requires information only for a specific domain. During the development of hypermedia applications, developers require varieties of tools to manipulate various objects, navigational links and multimedia aspects such as

authoring tools, audio tools, video editing tools, etc. However, traditional software applications can be developed using specific tools. Hypermedia applications often require well-structured design and intensive testing because the systems will be used by various classes of users with very low user tolerance to failure of the system or system complexity. This means that end users must be completely guided by the system (Nanard and Nanard, 1995).

2.6 *Hypermedia design models overview*

Some hypermedia models are derived from database design models, and some are derived from object oriented design models. However, similar to conventional software engineering, all currently available models consist of basic development stages to construct hypermedia applications. As shown in Figure 2, development stages are differed from stages in software engineering. These stages are as follows:

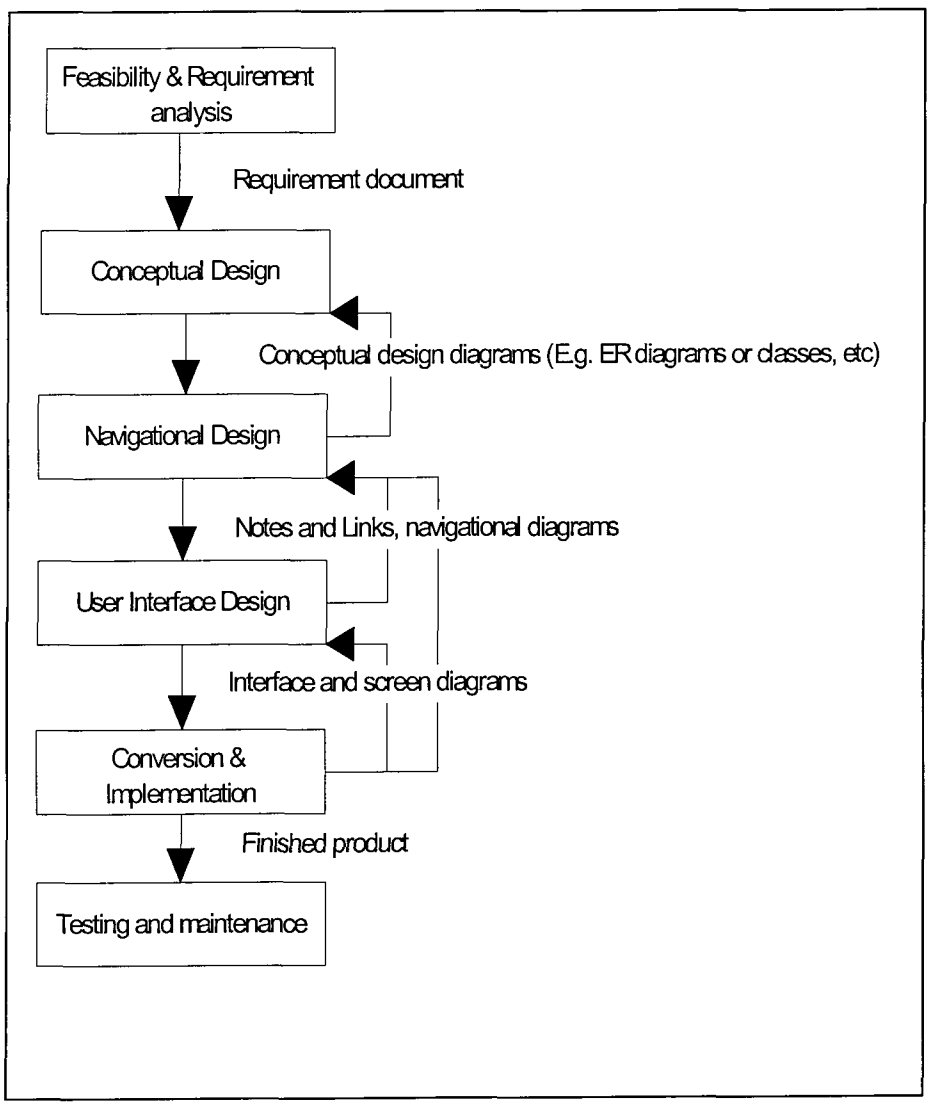


Figure 2 - Hypermedia development stages

Feasibility study & Requirement analysis - All development projects include a feasibility study and requirement analysis to assess the problem domain and identify the requirements. In this stage the designer has to carefully analyse the problem domain and break down the different aspects of the domain.

Conceptual Design - Unlike conventional developments, hypermedia applications require information structuring and the conceptualisation of organisational structures. It includes the design of structural aspects of the application. Some models such as HDM (Garzotto, Paolini, 1993) and RMM (Isakowitz, Strohr, 1995) use notations and structures from database design methods (e.g. Entity-Relationship (ER) model), while models such as OOHDM (Schwabe, Rossi, 1996) and OOHM (Hendrikx, Olivié, 1997) adopt object oriented classes and objects such as OMT, and BOOCH. Depending on the type of the application to be developed, the designer can select an appropriate method. For instance, if the application is only to present information without any concern for object behavior or functions, the designer can apply an ER model to develop the conceptual design.

Navigational Design – In comparing with traditional software application, the hypermedia applications require consistent and reliable navigational patterns. Since structured navigation is essential in the hypermedia applications, all hypermedia design models comprise navigational design to construct both static links and dynamic links between information nodes. In this stage designers have to provide rich and flexible links in many ways to create semantic networks of nodes. Many different navigational models can be constructed from one conceptual model to fit with the needs of different sets of users. Three main primitives used in the navigational design: index guided tours and indexed guided tours (Schwabe, Rossi, 1996). These navigational structures can be derived from nodes and relations from the conceptual design stage.

User Interface Design - User Interface Design can be regarded as detailed design for the navigational models to present information units to end-users. In this stage, designers identify and construct windows, forms, menus and templates, anchors and object presentation, links and other Web resources to obtain a consistent structure throughout the application. It is also necessary to consider the behavior of each object when interacting with users. Synchronisation has to be achieved between navigational objects and presentation objects. Some models such as RMM and HDM only focus on conceptual and navigational design and leave a detailed user interface design to the developers' choice. However, with the use of Abstract Data View (ADV) (Schwabe, Rossi, 1995), the authors of OOHDM have provided a detailed representation of interface object and links between navigation objects

and interface objects.

Conversion & Implementation - All methods are platform independent, except methods which are only for the development of WWW applications, and the product design of any method can be converted into the developer's chosen hypermedia development tool. In this stage, the designer can convert or generate a specific platform (e.g. HTML or independent hypermedia application) by using nodes and navigational patterns from previous stages.

Testing & Maintenance - As in software engineering, hypermedia applications require rigorous testing to ensure that they can be used by various sets of users. The main objective of applying design methods to hypermedia developments is to be able to maintain and update them easily.

These are a few design methods that have been proposed during the recent years. These include

- Hypermedia Design Method (HDM) (Garzotto, Paolini, 1993) by F. Garzotto,
- Object Oriented HDM (OOHDM) (Schwabe, 1995) by D. Schwabe,
- Enhanced Object-Relationship Model (EORM) (Lange, 1994) by D. Lange,
- Relationship Management Model (RMM) (Isakowitz, Strohr, 1995) and Extended RMM (Isakowitz, Kamis, 1997) by T. Isakowitz,
- WebArchitect(Takahashi, 1997) by K. Takahashi which extends RMM by identifying scenario analysis.
- HM Data Model (Maurer, Kappe, 1993) by H. Maurer which focuses on hypermedia database and object oriented approach.

Furthermore, other methods such as Structured Hypermedia Design Technique (SHDT) (Bichler and Nusser, 1996) by M. Bichler, which is presented for the development of WWW sites and Object Oriented Hypermedia Model (OOHM)(Hendrikx, Olivié, 1997) by K. Hendrikx also have been proposed. Figure 3 shows the relations between different hypermedia methods that have been proposed. Authors of these methods have based their methods on well matured existing techniques from database and object oriented models. Among these methods, some provide clear design stages, but not all of them have been presented with clear and explicit descriptions of design steps and structural guidelines. Later in this chapter, we will analyse three core methods namely HDM, OOHDM and RMM. Table 1 shows the different development stages of some hypermedia design methods. However all methods employ the same kind of basic design stages that are described above.

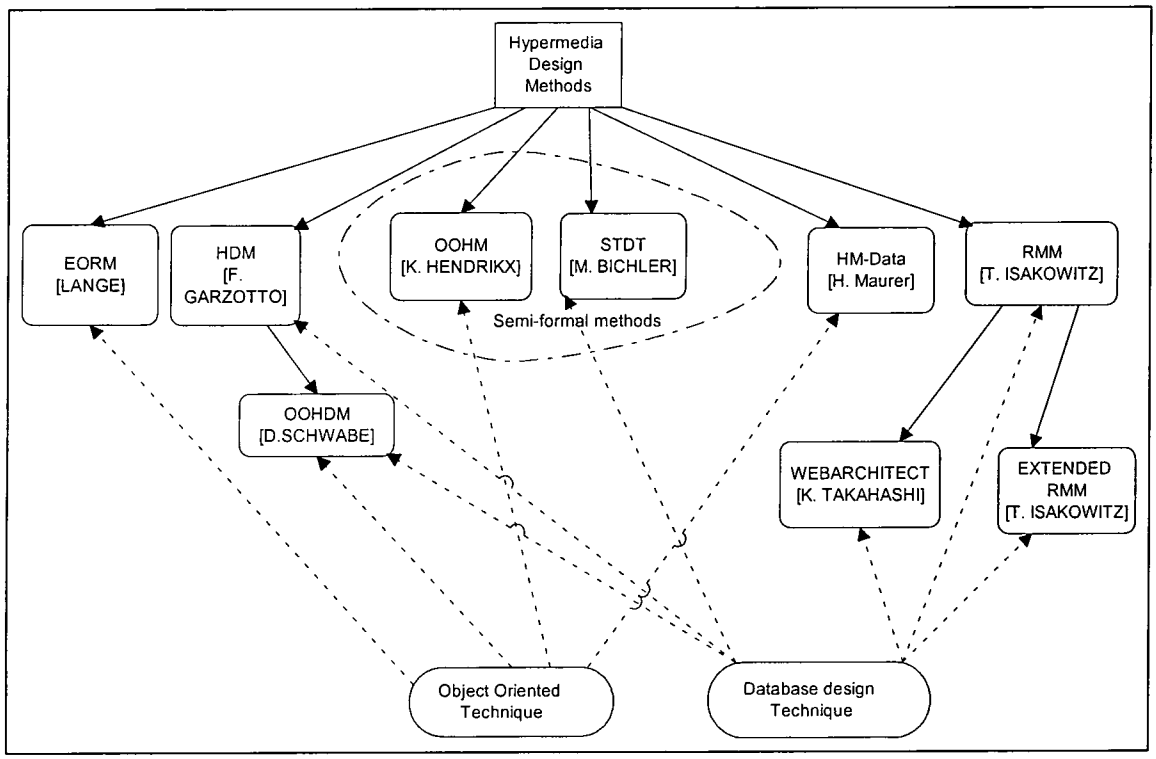


Figure 3 - Proposed hypermedia design methods (Relationship hierarchy)

HDM	OOHDM	RMM	SHTD	Extended RMM	Web Architect	OOHM
ER Design	Conceptual modeling	ER Design context	Information structuring	ER Design	ER Design	Hypermedia Model
Entity types	Conceptual classes, objects	entity, attribute, relationships	Navigational design	M-Slice Design	Scenario analysis	set-based nature
Derived Entities	Navigation modeling	Slice Design Context	Organisational design	M-Slice Navigational and Interface Design	Architecture design	Object-Orientation
Navigational Design	Navigational classes. objects, links	Navigational Design Context	Interface design		Attribute definition	Hypermedia Methods
Links (Perspective, Structural, Application)	Navigational transformation	Node-link Conversion Context	Implementation		Construction	Object-Oriented methods + Navigational structure + Relationships + Layout
Derived links	Abstract Interface Models	User-interface design Context	Introduction & maintenance		Maintenance	Implementations
Browsing Semantics	Abstract Data View (ADV)	Hyperbase Population Context				OOHS
Different semantics	Implementation	Prototyping Context				

Table 1 - Design and development stages from different methods

2.7 Comparing the methods

In this chapter, we will consider nine main criteria that should be covered by a hypermedia design method. These are as follows:

Notations of design: Notations of different design approaches in each development stage (e.g. such as E-R diagrams, Object Oriented class diagrams, etc) include the representation of the application domain from a very general top view to all the attributes and the relationships among them.

Multimedia support: The adequate support for accessing, presenting and navigating multimedia data for hypermedia application domain.

Modeling constructs: The consideration of data modeling constructs by each design methods. This includes conceptual and navigational design constructs and interface design activities. It also consists of modeling approaches for static and dynamic navigation.

Database support: The support of hypermedia access and navigation structure for building hypermedia database application domain adequately.

Reusability: It includes the reusability of pre-existing classes, models and design constructs. For example, how well a design method can support reusability to develop new application classes or entities using existing components.

Maintenance: Maintenance of hypermedia applications is a time consuming and difficult task since they are constructed using complex designs and not directly accessible at run time. An evaluation will be made on how efficient it is to maintain and update the applications developed using different design methods.

Types of applications: Since there are a broad range of hypermedia applications, there is no formal design approach that is relevant to designing all of them. Evaluation will also be made on which design methods are most suited to different types of applications.

User Interface Design: The support of user interface design, including GUI techniques and other widgets.

Tools support: The use of CASE TOOLS to support each method.

Three main hypermedia design methods, namely HDM, OOHDM and RMM will be discussed later in this chapter. The main reason for choosing these three methods is that each is presented with well defined stages and also represents a major class of hypermedia design methods as can be seen from Figure 3.

First, a general overview of the methods itself is given, then such particulars as: what levels of complexity the method covers, how the method is applied, what are the goals of each stage, what deliverables and diagrams will be produced by the method.

Current design methods described above inherit methods from different existing techniques, though in general, they consist of the same basic development stages. All development starts with feasibility study and requirement analysis. Conceptual design is to capture and understand the domain. It includes the identification of the necessary attributes within objects. It also contains the identification and construction of relationships between objects. Based on the method used to develop the application, the method of capturing the domain can be varied. Relationship Management Model (RMM) uses ER diagrams (i.e. entities, attributes, and relationships, etc.) and on the other hand Object-Oriented Hypermedia Design Method (OOHDM) uses class and objects to represent the relationships between objects. However the main purpose of this stage is to capture the context of the application domain. The navigational design helps developers to specify the navigational features using various notations. It includes the construction of links, nodes, access structures (i.e. menus, indices and guided tours) based on user tasks. Various types of extra links can be produced to fit different user needs. One or more navigational design models can be produced from one conceptual design for different type of users (e.g. students, visitors, etc). Once the navigation design is defined, user interface design can be constructed which makes the application more perceptible to the user.

2.8 *Hypermedia Design Model (HDM)*

2.8.1 The Method Overview

HDM is the first method that contains the full lifecycle for developing hypermedia authoring applications. It was presented by F. Garzotto in 1993 (Garzotto, Paolini, 1993). The HDM model is focused on presentation of information objects and navigational structures of complex applications without any implementation details. This is therefore a system and tool independent design model. The method focuses on the construction of large and complex applications, what the author called authoring-in-the-large. It refers to the specification and design of global and structural aspects of the hypertext application. HDM is based on authoring-in-the-large and provides a schema, which consists of a set of guidelines to help an author captures the structural interconnections. HDM helps the authors to design and construct the application structure efficiently with the use of its primitives. These are less focused on the detailed user interface design and implementation of the application. With the help of CASE tools, the full application can be produced.

Like Relationship Management model (RMM) (Isakowitz, Strohr, 1995), HDM is based on database design principles to conceptualise a given application. Garzotto has presented the HDM with many different levels of abstraction. It uses techniques and terminology from database design models of software engineering such as data modeling and entity relationships.

2.8.2 Design Process of HDM

As shown in Figure 4, the HDM design hierarchy consists of different levels of hierarchies of components. Firstly, *Entity*, which represents a real world object of the application, for instance, a house, a car, etc. Entities with related features are grouped as *Entity types*, which become more abstract, such as a vehicle, which can be a car, a truck, or a bus, etc. These representations are derived from the Entity-Relationship Model (ER) which is a data modeling method from the database design. On the other hand, they are similar to abstract classes of object-oriented environment. In comparison with the conventional database ER model, HDM consists of more link structures to support hypertext features. However unlike object-oriented models, entities represent only object attributes and relationships and do not cover the behavior of the object.

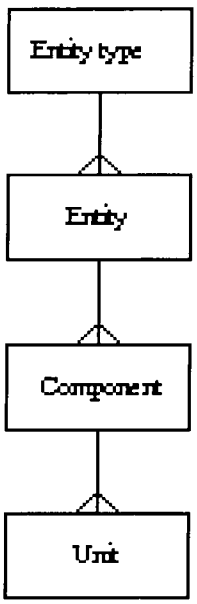


Figure 4 HDM design hierarchy

An entity consists of a group of *Components*. A real world entity is organised as a series of components. For instance, an engine is a component of a car. Components can then be in a hierarchical order from top to bottom. This means that a component can be composed of a group of smaller components. A component consists of one or more *Units*. A unit represents the actual information in an object and can be presented in various formats (picture, text, audio, etc).

Garzotto has described the hierarchy in HDM as a syntactical structure to organise information when building application which the diagram is pointed out in Figure 5. To support different users and their intended tasks, he describes the use of different *Perspectives*. The main purpose of perspectives is to present the same conceptual information from different perspectives to meet different user needs. For instance, the description of a car engine can be in the form of text, voice and pictures or the description of same information in many different languages. The perspectives can be applied in any level of the hierarchy (i.e. from unit level to entity type level) in the HDM model. These different perspectives in HDM play the same role as different navigational models in OOHDM (Schwabe, 1995) with some differences. Navigational models in OOHDM are more focused on the global navigational views of the same domain while perspectives in HDM are focused on different representations of the same information.

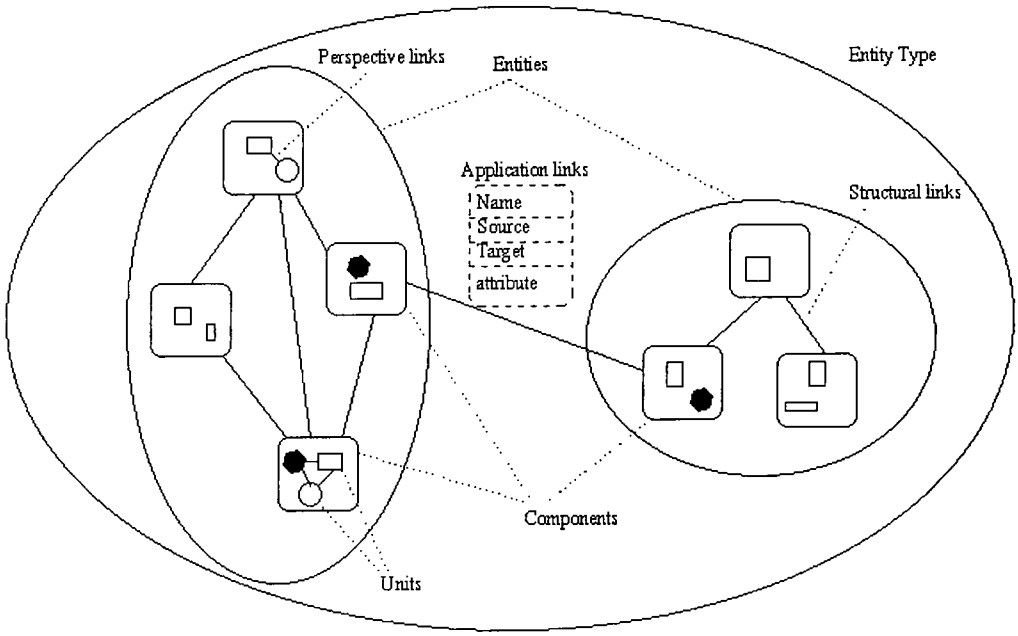


Figure 5 HDM primitives and link types

HDM provides different types of links to support various features of hypertext. In HDM, Garzotto emphasises on a representational role and a navigational role during the construction of links. He stresses three categories of links. They are as follows:

- *Perspective links* - Perspective links are low level links that connect units within the same component.

- *Structural links* - Structural links interconnect different components within an entity.

- *Application links* - The author can choose these links based on the nature of the application and they are domain dependent. They can be used to link among entities and their components. Garzotto has described the application link with four attributes to clarify relationships between objects, namely a *name*, a *set of source*, *target* and a symmetry *attribute*.

HDM also includes *Outlines*, which are ordered trees of the components. They are different from entities, as they are additional links to non-leaf components with entities. They are not typed as in entities and can be freely added by the designer to aid user navigation (Garzotto, Paolini, 1993).

Garzotto has also described how the *derived links* and *derived link types* play the same role

as navigational links in OOHDM (which will be discussed later in the chapter). Derived links are the links which can be automatically derived from their basic link structures. They can be derived from structural links, or derived from perspective links, or derived from applications links. For example, based on the structure of components and units, perspective links can be generated automatically.

As an overview, HDM has contributed the way to the design of logical structure of the applications. It comprises the use of database design methods with different modeling primitives to represent structured conceptual and navigational design.

2.8.3 Notations

Garzotto has not proposed new notations for HDM to construct hypermedia applications. However HDM is rich in design primitives for both conceptual and navigational design. Since HDM uses database design, notations from ER diagrams can be applied. Garzotto has also described the use of different links and link types and leaves it to the designer to create different notations to reflect his or her understanding.

2.8.4 Multimedia support

Garzotto has not described the manipulation of multimedia objects in user interface design, HDM is focused on design of general information nodes with logical views.

2.8.5 Modeling constructs

From the modeling perceptive, HDM provides a variety of link types and primitives for modeling application structures independently. It also consists of a collection of navigational primitives (e.g. Outlines, application links, structural links, etc) and browsing semantics to support modeling of navigational structures. However, HDM is only suitable for database oriented application such as information retrieval and does not emphasise other functional and business activities.

2.8.6 Database support

Since HDM uses database design approaches with navigational primitives, HDM is well suited for database oriented applications and applications for information retrieval.

2.8.7 Reusability

In the HDM model, information can be reused within the application or generate a new application (Garzotto, Mainetti, 1996). The same components and Units can be reused *by reference* method from different places using navigation. Furthermore structural links within entity can also be used by associated components.

2.8.8 Maintenance

The main objective of using HDM modeling approach is to construct application structures for easier and quicker maintenance. Since the use of hierarchical primitives in HDM provides modular structures, these can be easily updated.

2.8.9 User Interface Design

Unlike OOHDM's (Schwabe, Rossi, 1996) abstract interface design, HDM focuses only on conceptual and navigational design. The use of collection of links within different levels of hierarchy together with HDM primitives can transform richly connected interface objects, and cross-referenced bodies of information (Balasubramanian, 1994). However, Garzotto has not described the issues of interface design components such as Web views, maps, and the synchronisation of multimedia objects.

2.8.10 Tools support

In 1993, HYTEA software development project implemented a set of tools based on HDM model to support the development of large and complex hypermedia application under the ESPRIT programme.

2.9 *Object Oriented Hypermedia Design Methodology (OOHDM)*

2.9.1 The Method Overview

It seems that the Object Oriented Hypermedia Design Methodology (OOHDM) combines the entire abstraction and inheritance features from Objects Oriented technology together with complex navigational features from HMD. It was presented by D. Schwabe in 1995 (Schwabe, 1995). The model focuses on the design of the logical structure of the application domain in terms of navigation and interface design aspects. The use of objects contrasts with the relationship models (described above with respect to HDM) where different constructs are required for different levels of data item detail and where there is a clear distinction between data and its description (metadata) (Brown, 1991).

2.9.2 Design Process of OOHDM

The process consists of four main design stages. They are as follows:

Conceptual design

The main objective of this stage is to analyse the global application structure using object-oriented models. It includes the design of conceptual classes, generalisation/specialisation hierarchies, aggregation, object instances, abstractions and subclasses. Since it is the first stage of the development, it has very little concern about the detailed tasks of users and navigational patterns (Schwabe, Rossi, 1995). The output of this stage comprises domain classes, subclasses and their relations.

Navigational Design

In this stage, the designers develop all the navigational structures of the application, which reflect the user requirements and tasks. It uses navigational contexts and link notations to perform structured navigation of the classes from previous stage. Like other methods, nodes, links, indexes and guided tours are used to navigate between nodes. In OOHDM, nodes are regarded as conceptual classes. Each node consists of a different set of attributes, which can be accessed between conceptual schemas. In navigational design, different sets of navigational models are developed for the same conceptual schema to express different sets of users and tasks. In other words, different navigation paths can be constructed if the application will be used by more than one user profile.

During the development of navigational model, the designers have to consider all possible ways of navigation required by different users. The navigational design models can be inherited from the relationships between classes based on the conceptual design or can be developed independently. In the later case, they are easier to maintain. The navigational links can be static or dynamic. When designing dynamic links, designers have to define the transformations that occur while traversing links. Once the conceptual design and navigational structures are completed the framework of application can be visualised. Iterative maintenance may be performed to fix bugs within nodes and links.

Abstract Interface Design

This is the step where abstract interface models are designed with text and multimedia objects such as images, animations and audio. Navigational objects are mapped with interface objects that produce perceptible appearance. By integrating navigational objects and interface objects designers can produce many interface models to the same navigational object for usability.

The method uses Abstract Data View (ADV) charts from object-oriented models to represent interface objects. It includes detailed interface design of each object in different perceptive including their appearance and related user events. The detailed approach of designing interface objects and ADV charts can be found in (Rossi, Schwabe, 1995) and (Schwabe, 1995).

Implementation

The product of navigation and interface design can be implemented with currently available hypermedia development tools such as Director, Toolbook, or HTML, etc.

2.9.3 Notations

Since the model uses object oriented modeling principles, notations are applied to conceptualise the application domain. These include notations for the construction of sub-systems, classes, and relationships. However Schwabe has not presented new notations explicitly for the construction of conceptual design. Standard OO notations (such as OMT, UML, Booch, Etc.) may be applied.

2.9.4 Multimedia support

OOHDM is primarily focused on the construction of information objects and classes in general. However with the use of an object oriented multimedia database, different media objects can be defined and instantly constructed. The designers must carefully deal with navigational classes, static objects (text, graphics, etc) and dynamic media objects (sound, movie, etc.).

2.9.5 Modeling constructs

With the use of currently available object oriented design approaches, (Rumbaugh, Blaha, 1991) OOHDM provides rich modeling primitives for the development of conceptual and navigational design. For the interface modeling point of view, OOHDM uses Abstract Data Views (ADV) (Cowan and Lucena, 1995) to model interface design. In general OOHDM can be regarded as an approach for advanced modeling applications. Furthermore, with the use of OOHDM, different navigational models can be constructed based on the same conceptual design for different sets of users and their tasks.

2.9.6 Database support

OOHDM uses object oriented design principles to construct relational and network data models with data oriented records. With the power of object oriented design, advanced database oriented hypermedia applications can be produced using OOHDM. The composition of an object oriented database with hypermedia structures offers the representation of high level structures with real world entities (Brown, 1991).

2.9.7 Reusability

The term reuse can be used to mean either the reuse of components, the reuse of structure, or simply the reuse of information. It is also possible to reuse different parts within the same application or in two different applications (Garzotto, Mainetti, 1996). The product design of OOHDM can be reused in various ways, e.g. since the model can be platform independent, the same application structure and information can be implemented on different platforms. However, two different applications with different structure will have very little to reuse or may not be well suited to reuse.

2.9.8 Maintenance

Information updates and changes can be easily done. However, if the changes of structure of the application are likely, the designer will have to redesign part of the module or the whole model depending on the complexity of the application.

2.9.9 User Interface Design

The use of ADVs charts allows the designers to model the user interface design including interaction and graphical details on user generated events.

2.9.10 Tools support

Currently there is no information available for any graphical CASE tool that supports OOHDM model.

2.10 Relationship Management Model (RMM) and Extended RMM

2.10.1 The Method Overview

Relationship Management Model (RMM) is a method for the design and management of relationships among various information domains to produce structured hypermedia documents. RMM was presented by T. Isakowitz in 1995 (Isakowitz, Strohr, 1995). It is based on HDM, which has been discussed above in the chapter.

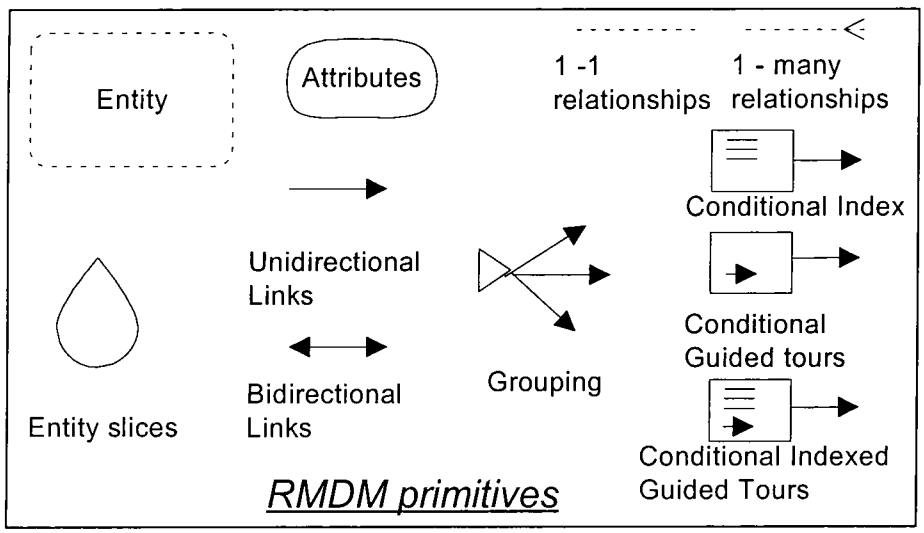


Figure 6 RMDM design primitives

RMM is based on Entity Relationship Model (ER) to describe information objects and their relationships. The main difference between ER and RMM is that RMM consists of not only notations to represent conceptual design but is also used for the purpose of navigational design. As shown in Figure 6, it comprises RMDM data model which includes several design primitives (Isakowitz, Strohr, 1995). The reason for using ER diagrams is to breakdown large and complex information domains into smaller entities. The use of ER diagrams not only assists the designers by representing the entities graphically but also represents abstract objects and their relationships. To group the related attributes within an entity, RMM uses slices, as shown in Figure 8. The slice primitive is the group of attributes within the associated entity. Like other methods RMM uses six access primitives for navigation, as shown in Figure 6, namely, unidirectional and bi-directional links, grouping, indices, guided tours and indexed guided tours.

RMM consists of seven main design and development stages excluding requirement analysis. However, every application development has to begin with a feasibility study and requirement analysis. The main objective of requirement analysis is to collect all the information about the application domain. The information mainly includes objectives of the system, scope of the users, different media to be used and cost-benefits analysis (Isakowitz, Strohr, 1995).

2.10.2 Design Process

The main design processes of RMM are as follows:

Stage 1: ER Design - The use of ER diagrams helps the developer identify data objects and their relationships graphically. ER diagrams can represent the data content contained in the requirement dictionary and information dependencies. By designing the application domain using ER diagrams graphically, designers can visualise the foundation of the hypermedia application. An ER design consists of entities and relations between entities. Each entity consists of different types of attributes, which can then be grouped using different slices (which will be discussed in stage two). ER diagram also consists of 'associative relationships' which represent relations between associated entities. The relations can be one to one, one to many or many to many. As shown in Figure 7, one lecturer teaches many students (one to many). One student takes many courses and one course consists of many students (many to many). In general the main objective of this stage is to define associated entities and relationships amongst them.

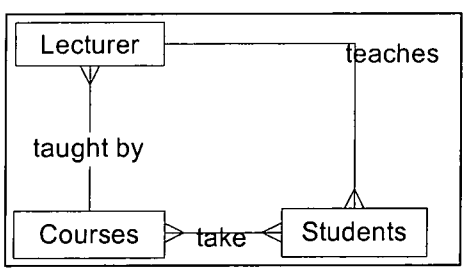


Figure 7 Example Entity-Relationship (ER) diagram

Stage 2: Slice Design - A slice is a piece of information of attributes within an entity to be presented to the user. In other words, an entity consists of a group of related slices. Therefore the user can view the information in structured and meaningful units. Each slice consists of one or more attributes related to that slice and may share a common attribute. M-Slices are constructed based on extended RMM M-Slice design. M-Slices are the extended version of

slice which have the ability to group information attributes from various related entities. Each entity consists of one or more slices. From the navigational point of view, slices can be linked using unidirectional and bi-directional links. Therefore each entity consists of attributes, slices and links between slices. The designer must be aware that creation, updating and deleting the links can effect the information context within the slices and entity. The designers must have analysed the application domain before deciding on the header slice and other related slices. The information within the header slice must start from general information with links to more detailed slices. It also important for the user that each slice should consist of only relevant information attributes.

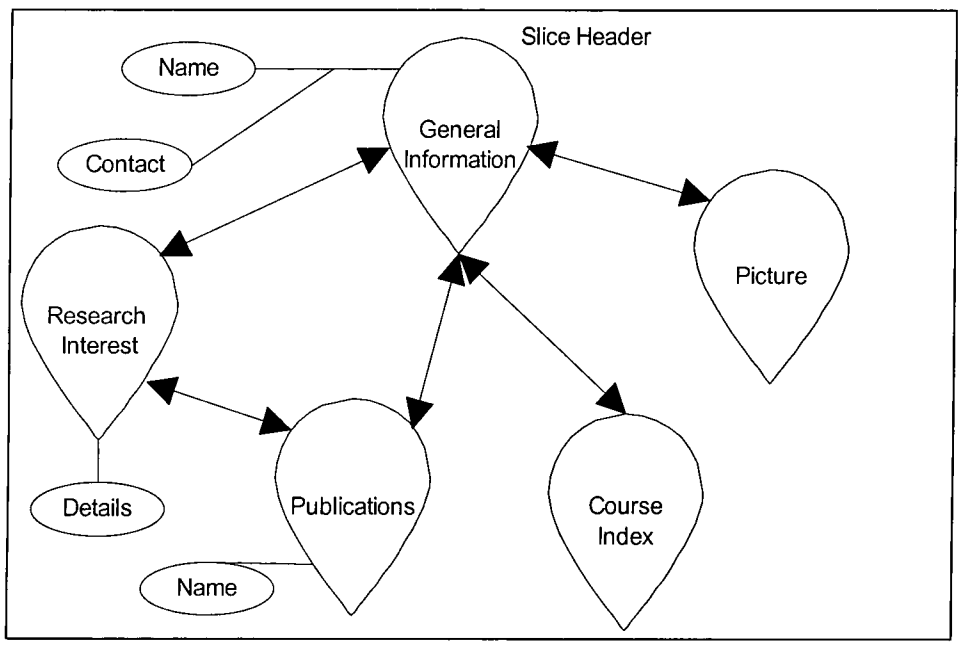


Figure 8 Example SLICE Diagram within an Entity

As shown in Figure 8, both slices, 'General Information' and 'Publications', share the same attribute called 'Name'. An entity consists of a header slice and its default initial information. As shown in the figure, the Lecturer's general information is the header slice within the lecturer entity. General information will be displayed first when the user accesses the lecturer's page.

Stage 3: Navigational design - As this stage, navigational links are created using access structures that can be indices, guided tours, or indexed guided tours. In RMM navigation design, navigational contexts are designed in a generic way, which can be updated easily. This means that there are no hard coded links between entities and links are made by referring to properties of entities and of relationships (Isakowitz, Strohr, 1995). Since the navigation is an important aspect of hypermedia application development, designers have to analyse carefully all the associated relations between entity relationship diagrams. Figure 9 shows an example navigational design for lecturer and courses relationship.

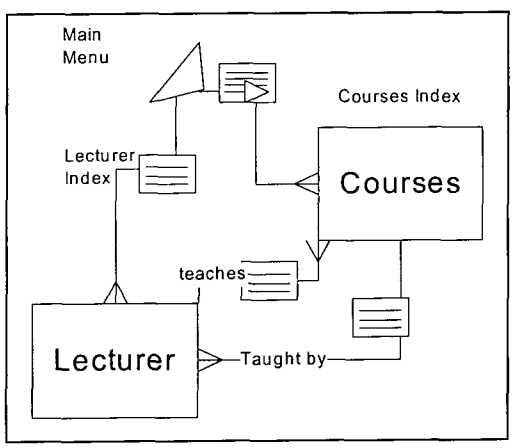


Figure 9 Example of Navigational Design

Stage 4: Conversion protocol design - In this step, ER and navigational diagrams are transformed into the designer's chosen platform (HTML, TOOLBOOK, etc.).

Stage 5: User-interface design - In this step, screen layouts are constructed based on RMDM diagrams. These include the construction of graphics, text, navigational aids, backtracking mechanisms, etc (Isakowitz, Strohr, 1995). Although user interface design is essential for any hypermedia application, the authors of RMM do not explore the detailed user interface design patterns using RMM.

Stage 6: Runtime behavior design - Based on the size of the information content, the designers have to decide which information will be static and which information will be dynamically computed during runtime.

Stage 7: Construction and testing - In this stage, the application is constructed and tested based on the user's requirements.

Extended RMM

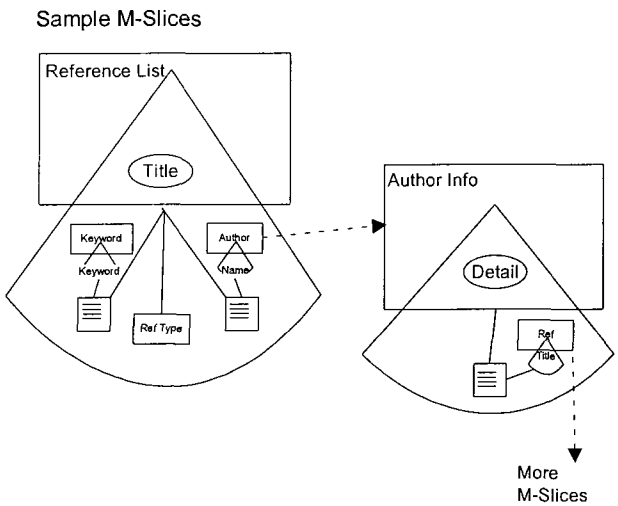


Figure 10 Sample M-Slice example

Isakowitz has also proposed an extended RMM by adding m-slices as an improved model. (Isakowitz, Kamis, 1997) The main purpose of adding m-slice is to overcome the limitations of RMM. Since RMM only allows the designer to add own attributes and slices within the entity, there are no multiple links (i.e. relations to slices from other entities) from one slice. Consequently, the resultant links are in separated pages. With extended RMM this problem can be overcome by adding m-slices. Isakowitz has stated the M-Slice as:

'M-Slices are used to group information into meaningful information units. M-Slices can be aggregated and nested to form higher level M-slices.'

Figure 10 shows an example of M-slice. The information about reference details (owned by reference entity) can be presented together with 'Title' and a list of author name(s), which is owned by the author entity, and keywords, which are owned by keyword entity. These can be integrated together by using relations. Therefore, unlike RMM, the users do not have to go though to acquire information about author or keywords.

2.10.3 Notations

The notations for describing the structure of the system are the ER based diagrams with navigational primitives, as shown in Figure 6. Accordingly, they are easy to construct and modify.

2.10.4 Multimedia support

There is no specific detailed information for storing and retrieving multimedia data items for static items (i.e. images, maps) and dynamic items (i.e. audio, video, animation, etc.). The author of RMM only focused on overall design of the method and omitted the details of each multimedia object.

2.10.5 Modeling constructs

When describing modeling constructs, Isakowitz has claimed that the method attempts to model the whole of the development lifecycle. The development stages are included in the RMM help to validate this claim.

2.10.6 Database support

Since RMM is based on database design approaches, database oriented applications (that have consistent data structure with constant updates) are well suited to this approach.

2.10.7 Reusability

It is unlikely that it will be possible to reuse the structure of one hypermedia application in another application. If the problem domain of the new application is similar to the new domain, existing diagrams can be modified to fit with the new applications. However, the developers should not force old diagrams to fit the new application.

2.10.8 Maintenance

To update or maintain the content of the application can be easily done. However the maintenance for the structure of the application is likely to require that the designers redesign most or part of the existing design.

2.10.9 User Interface Design

In RMM, Isakowitz has not described detailed modeling for mapping navigational objects and information objects. Designers have to carry out the product design using RMM by employing currently available hypermedia development tools.

2.10.10 Tools support

Isakowitz has developed RMCase (Díaz and Isakowitz, 1995), a computer-aided environment to support the RMM design method. It supports the essential basis for the development of hypermedia applications such as information feedback loops (Nanard and Nanard, 1995) and instant level creation and cloning of objects. However, RMCase is only a prototype and is unsuitable for the construction of large and complex applications. The main features of the RMCase tool will be assessed in chapter four when defining requirements for new CASE tool.

2.11 An overview of design methods

The above section shows three main hypermedia design methods and their benefits and limitations. Although these methods have different design stages and approaches, they consist of the same basic development stages presented in Figure 2. Depending on the problem domain, it is the designer's task to choose the approach.

2.12 Summary of the Main Points

As a summary, we have presented different hypermedia features including how hypermedia can assist the end users in presenting and structuring the information to meet user needs. We have also described how hypermedia design methods can be applied for developing Web application including their benefits and limitations. We also analysed different design methods and have shown how they are derived from one another and evaluated the three main methods among them. The main contribution of survey is to understand the existing hypermedia design methods and to assist the designers and developers in choosing the right method and tools for any specific applications.

However, due to the proliferation of methods in recent years, standardisation is required between different methods in terms of notations, modeling and implementation. As the development of Web based hypermedia applications is increasing, the need for flexible

design tool support is obvious yet only very few CASE tools are currently available. Accordingly, A new CASE tool will be implemented and the implementation features of that tool will be discussed in chapter five.

3. Case Study based Evaluation of Hypermedia Design Methods

3.1 Chapter Abstract

After surveying existing hypermedia design methods, this chapter evaluates these methods by redesigning an existing Web site (reverse engineering) to a new structured Web site. Most fundamentally, it also illustrates how existing complex and large Web sites can be redesigned into maintainable structures using systematic design methods. Since most of the Web sites are already implemented, it shall be a simple presentation for those who intend to create or update their WWW sites and wish to take full advantage of structured design methods. In previous chapters, we have discussed various hypermedia design methods and presented their features. This chapter shows the ways to reverse engineer an existing Web site to a new structured Web site by using two hypermedia design methods that were presented in chapter 2, namely RMM and OOHDM. Since this chapter has only focused on design issues, any implementation details will be omitted.

3.2 Introduction

The use of WWW (World Wide Web) and hypermedia applications have dramatically increased. Many organisations, institutes and others have created numerous Web sites and they are still producing many Web sites for different aims and objectives. However many WWW designers and novice users are currently developing their Web applications with their own knowledge without applying any explicit design methods. Potentially the design of large applications becomes a remarkably complex task and is likely to result in maintenance problems. On the other hand, researchers and developers have proposed different hypermedia design methods based on software engineering approaches to support the development of Web applications. Later in this chapter, we will be showing how existing Web sites can be redesigned using structured methods by applying these to an existing Web site as a case study.

3.3 Case study background

The Software Engineering Group (SEG) Web site is designed to present information about the software development projects at the Department of Computer Science in University of Durham. These projects are given to undergraduate students to enable them to obtain experience of a software production environment similar to that found in the computer industry, i.e. working as a part of a design and implementation team to tight schedules. The SEG Web site includes the following information.

- Project Structure – Project structure includes information concerning project aim, attributes of the project and group organisation. Project aims include information related to software development, group work, Programming Style, Realism, etc. Group organisation contains information about Group Members, Software Consultant, Group Meetings, Software Project Co-ordinator, and System Support.

- Project Outline – It presents information about the system that the department wishes students to develop. It contains a brief description of the background of the system to be developed and the main requirements that need to be implemented by the students.

- Group allocation – It includes information about students who are developing the projects and their group supervisor allocation.

- Project Phases – Project phases are presented and along with the deliverables to be produced. It also includes information about the development time (in weeks) and when each project development stages to be completed.

- Intermediary Tasks – This section presents the group-work tasks to be carried out throughout the year. It also includes information about each task to be completed in each week.

- File structure - All documents produced will be stored (and delivered) in BSCW workspaces. Each member of a group will be allocated a BSCW UserID and password for use in a private workspace. All work for this project should be undertaken under this UserID and should have the fixed project file structure. All documents for submission should be produced in HTML, and when they are ready for submission they should be placed in the Documents folder within the groups' private workspace. The BSCW workspace consists of three main folders: Private Noticeboard, Documents, Practical Exercises. The Private

Noticeboard folder can be used by a group to post notices relating to the project. The Documents folder is used to store the student's HTML documents, and consists of the following sub-folders : Documents / require, Documents / reqappr, Documents / design Documents / imple, Documents / test.

Inside each folder all documentation appropriate to that phase must be placed. The final deliverable for each phase should be in the appropriate folder and named <folder-name>.html This document should be under version control. The highest numbered version of the document will be the one collected and marked. The folder Practical Exercises is used to store the deliverables for the Intermediary Tasks.

- SEG Lab – To carry out various development tasks, students need to use SEG laboratory.

This section presents detail information about booking information for booking of SEG Lab. The students can create new bookings or cancel existing bookings.

3.3.1 Requirements for the New Web Site

The reverse engineering of the SEG Web page was started by the analysis of application domain to identify the requirements for the new Web site. The analysis mainly includes current Web page design, information grouping and navigational structures. But it also consists of the identification of domain objects and their navigational links. Furthermore, the analysis is made on the current use of navigational access structures such as indices and guided tours. It takes considerable amount of time to evaluate every detail sections of the site so consequently the reverse engineering was made on top level view of the current design.

The HTML documents presented in the current Web site are implemented without using hypermedia design methods. Although information presented in current Web site is complete, the structure of information is inconsistent. The navigational links are not carefully defined, i.e. there are many embedded links that are neither consistent nor link between related materials. After carefully analysing the current Web site, the following requirements are found from the design point of view.

- Restructuring the information into related groups.
- New navigation patterns to provide links between related topics throughout the Web site.
- Present information in an easily readable format using standard templates.
- Provide different navigational views for different set of users.
- Increase the level of accessibility and functionality.

3.4 Reverse engineering using OOHDM

In this section, the SEG Web-Site is reverse engineered and redesigned using OOHDM, for the purpose of evaluation of the usefulness and limitations of OOHDM. The evaluation includes: the object abstraction capabilities, reusability of objects, portability, notations, navigational perspectives, interface design constructions etc.

Furthermore, the nature of the hypermedia fields and design problems of SEG Website were presented in previous section. Accordingly, this section will not elaborate the requirements of the CASE STUDY nor actual implementation artifacts but will rather stress the design steps, development cycles, navigational views of the SEG Web site and the evaluation of OOHDM.

Brief outlines of development stages are listed below to present a clear orientation of the OOHDM design process. The full description of OOHDM is contained in chapter two.

- The first design step of OOHDM is the development of a conceptual model using object oriented primitives such as classes, subclasses, methods and attributes.

- Once the conceptual model is completed, the navigational model can be constructed using both features from object oriented method and hypermedia design methods. It includes:

    1. The construction of navigational node classes with different perspectives, which may include one or more attributes from all or part of the conceptual classes.
    2. Different navigational views which are defined for one conceptual schema to allow the different users to perform different tasks.
    3. The construction of various types of links including application links, structured links and access structures.

- Once the navigational structure has been constructed, abstract interface design is defined to make the structure perceptible to the users. It includes interface objects, ADVs and other multimedia objects.

- The products from abstract interface design can be implemented using any Web page authoring tools.

3.4.1 Reverse engineering of SEG Web site

The main steps of reverse engineering are to reverse engineer current implementation in order to design and rebuild that design to new implementation.

3.4.2 New Conceptual design

During Conceptual design a model of SEG Web site is designed using objects, classes and relationships between class. Objects are the instances of classes, and sub-classes are the types of classes.

As shown in Figure 12, boxes represent the classes, links represent relationships and small diamonds indicate aggregation. It shows a simple relationship between project and its development phases and the tasks to be carried out for the implementation of the project. The small classes project 'phase' and 'tasks', which are part of the project schema. The main 'project' class is part of the schema of the SEG Web site. In this stage, the representation is only used to capture the domain semantics without any navigational structures for user tasks. Figure 11 shows the simplified version of conceptual design of the SEG Web site. It consists of three main classes, 'project', 'person', and 'file system'. As shown in the Figure 11, 'student' and 'supervisor' are types of 'person' class and 'file structure' class consists of three subclasses, which inherit the main class. Each class consists of various attributes that can be implicitly defined in other classes as a relationship. Each attribute can have more than one perspective to be presented to the users. For example, 'project structure' can be presented using both text and diagrams. The relationships between objects can be mapped with a navigational view, however the navigational classes require more link structures to present information from various perspectives. For instance, there are no entry nodes for either students or the projects.

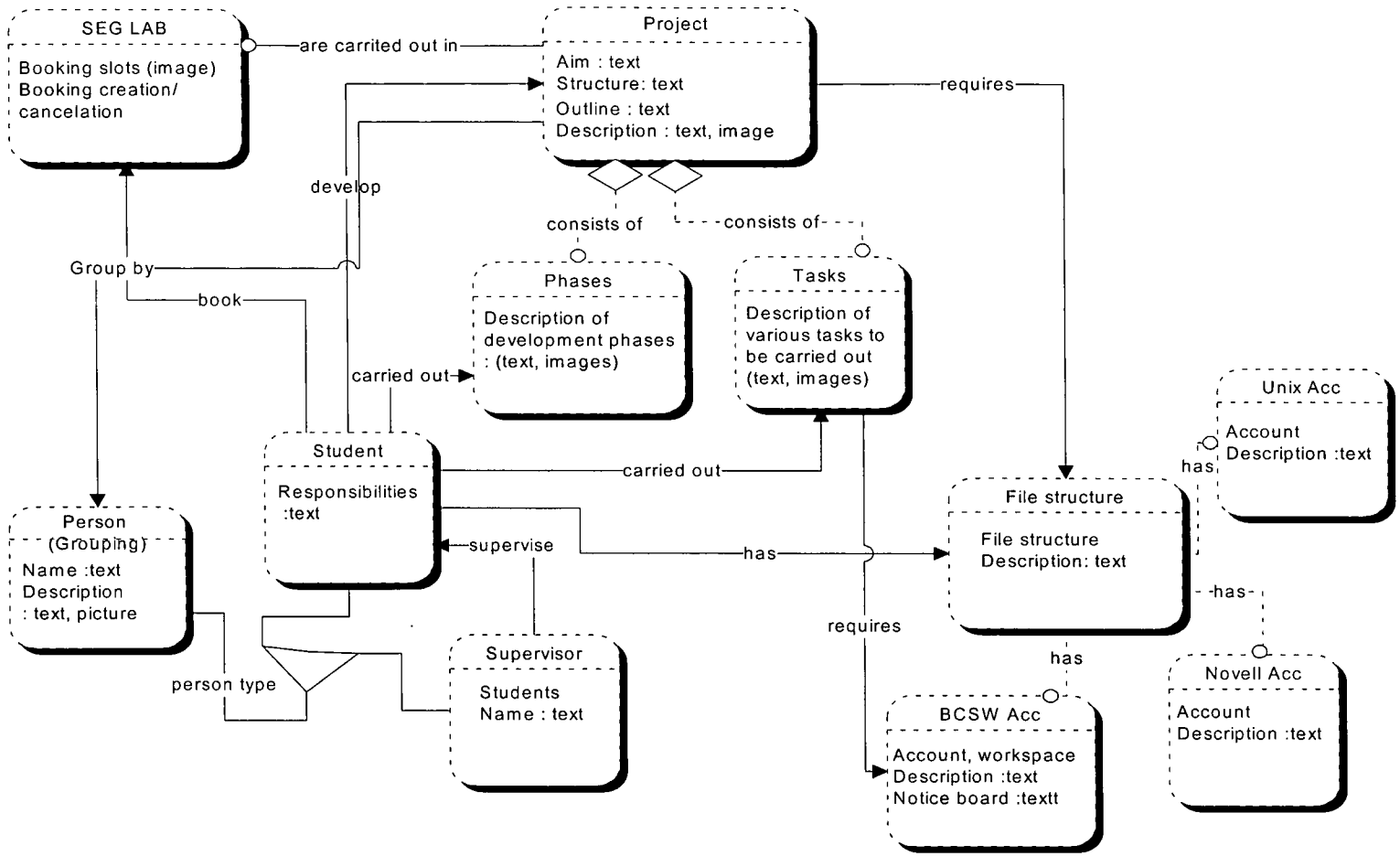


Figure 11 Conceptual design of SEG Web site (simplified version)

During the conceptual design the construction of aggregation and inheritance is vital to simplify and complex application domain. However the development of conceptual design requires many skills and an object orientation background., hence when conducting the design process, the designer requires adequate formalisms and tools for handling the formal aspects of the design with the support of designer's incremental and feedback loops (Nanard and Nanard, 1995). Adequate design environments are essential for the productivity of the designer in each design step.

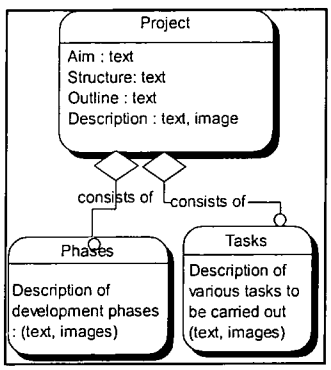


Figure 12 A sample Conceptual Schema (part of SEG project)

3.4.3 New Navigation Design

During this step, navigational node class and links are constructed based on user tasks. The navigational classes can be derived from one or more conceptual classes from a previous stage. When designing navigational models, different navigational views (models) can be built from the same conceptual schema to present to different users (Schwabe, Rossi, 1996). However for the SEG new design, two navigational views are constructed for the same conceptual schema. They are 'students' perspective and 'visitors' perspectives. Defining a navigational model consists of various classes such as navigational nodes, links, access structures, etc. Navigational nodes are windows for the users that may contain various attributes and anchors. This means that a node may consists of attributes from one or more conceptual class. Links represent the relationship between nodes. Link classes can be defined by specifying link attributes and behavior, source and target nodes and cardinality.

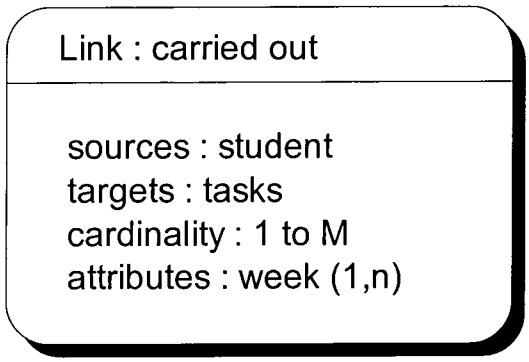


Figure 13 Link class

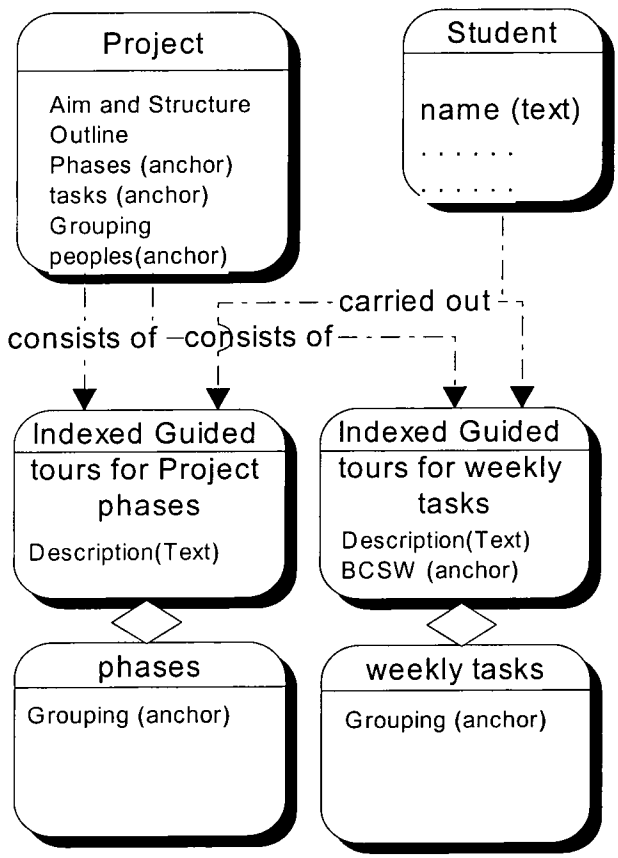


Figure 14 A sample Navigational Schema (part of SEG project)

Figure 14 shows nodes and links that are part of the navigational design of the SEG Web site. Boxes represent navigational node classes and the doted links represent the links between nodes. Nodes are derived mainly from corresponding conceptual classes. Figure 15 represents how navigational nodes are derived from conceptual classes. In the figure, each node consists of one or more attributes (components and units) from one or more attributes. For example navigational node 'project' is derived from 'project' conceptual class. Node class 'project' contains attributes such as project aim and structure and anchors to other classes. Since project 'aim' and 'structure' information will not be frequently seen by the students, they are embedded in the 'project' node. However, for the visitor's view, project 'aim' and 'structure' are presented explicitly as aggregate to 'project' node. The attributes and anchors of a node class can be any media types (text, graphics, sound, etc) but the detail apparent in each attribute can be designed in user interface stage. The attributes of each navigational node can be derived from a single conceptual class or combination of different conceptual classes.

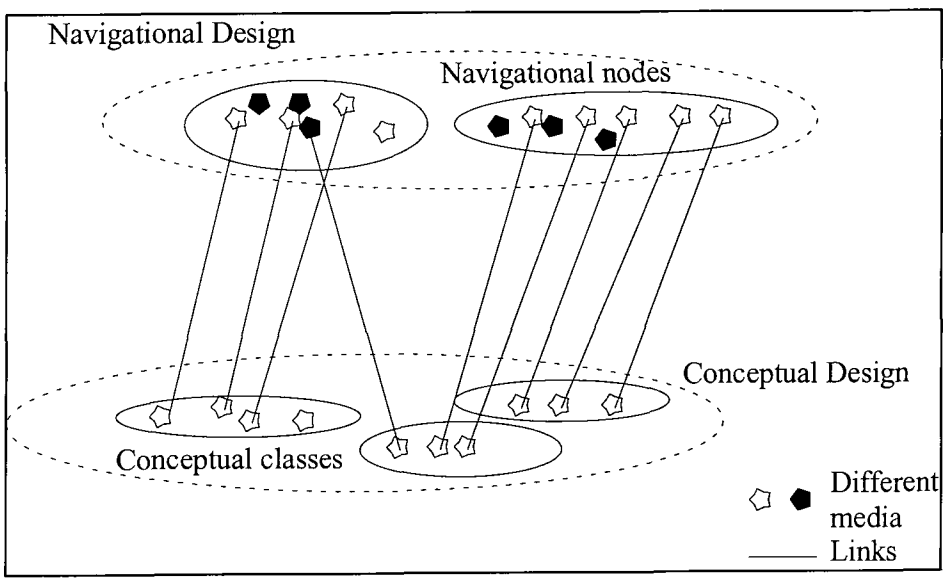


Figure 15 Conceptual design to navigational design

Nodes are interrelated by application links class and access structures. The rationale for creation of links and access structures is to explore fully the power of hypermedia and to stimulate the user's interest by implementing various types of connections among the nodes. (Schwabe, Rossi, 1996) The links can be created for various reasons. These include the representation of relations between object, associations, annotations and sequential links between objects. (Signore, 1996) In Figure 14, the relationship between 'student' node and 'task' node is presented by 'carried out' link class. As shown in Figure 13 the 'carried out' class consists of source: student, target: tasks, cardinality: 1 to N and attributes: weeks(1-n). In addition to link classes, Figure 14 shows access structures. Similar to RMM, access structures can be indices, guided tours or indexed guided tours. Access structures help the

reader to access the information easily. For example, project phases are presented using indexed guided tours. Access structures can be added freely by the author based on different perspectives. For instance, in additional access structures provide direct access to required information. Therefore students can directly access detail tasks, project phases and BCSW folder structure, however, from the 'visitors' point of view, project aims and structures are more useful than detailed weekly tasks.

The design that was constructed for this CASE STUDY requires only static navigational structures. However, the dynamic structure requires defining the navigational transformations that occur while traversing links (Schwabe and Rossi, 1995). In OOHDM, D. Schwabe presented the use of Navigation Charts to construct dynamic navigational behaviour (Schwabe, Rossi, 1996).

3.4.4 User Interface Design and Implementation

During this step, navigational classes are presented to the user by designing user interface. In this step, the new navigational design has to transform in a detail level to each interface metaphor and interaction styles. When designing interface objects, the author has to ensure what and how each object will be presented to the user. For example, what primitive (text, image, animated image or buttons, etc) will be used to provide the interface for a particular object. The consideration also includes the group of objects and how they are mapped to the navigational objects.

Figure 16 shows general Abstract Data View (ADV) for 'project' node. It includes the definition of group links and perceivable objects that describe the aim and structure of the project. These objects may be presented by text, images or diagrams and may include embedded links. The use of ADVs assists the design and transformation of navigational objects to perceptible interface objects. It also helps to design interface objects independently from navigational objects and hence different interface aspects can be derived from the same navigational view.

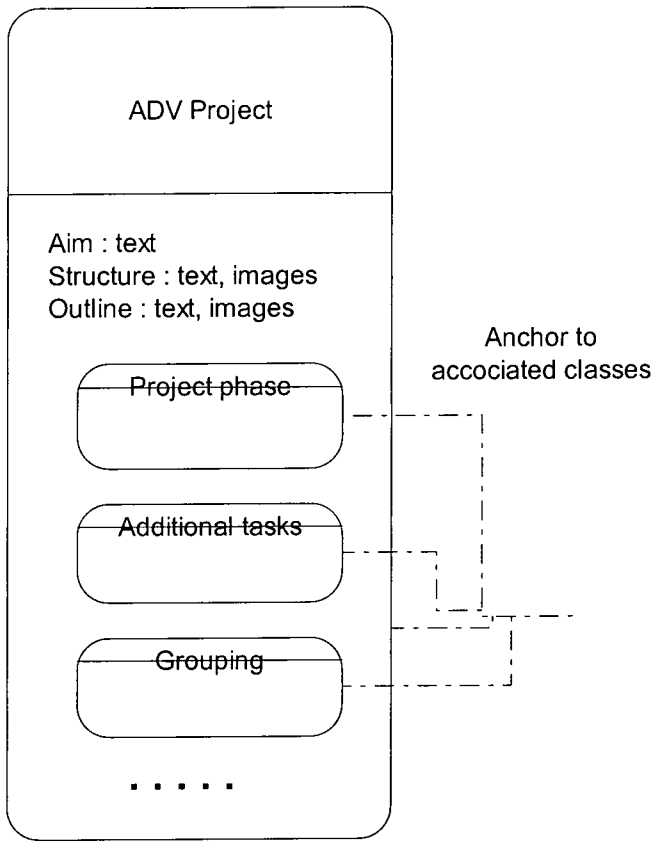


Figure 16 Simplify version of Abstract Data View (ADV) for 'Project' node

3.4.5 Conversion and Implementation

After identifying user interface design, the new prototype can be implemented from navigational design and interface design diagrams. For this example, we have implemented only parts of the whole Web structure into HTML pages as a trial usage. The whole construction was not made since it needs time to implement all the detailed features for each object in the interface design. Although in this example Web documents are needed, the resultant model from navigation design and interface design can be transformed into any other application platforms depending on the designer's choice. During the conversion, the designer may encounter the need for minor changes (such as adding user response, appearance of each layout), depending on the implementation platform.

3.4.6 Evaluation of OOHDM

During the development of SEG Web-site, OOHDM guidelines help to eliminate inconsistencies and mistakes. If the application is conducted in a relatively straightforward manner in order to convert linear text into hypertext structures, OOHDM is not appropriate. OOHDM applies hypermedia features and design guidelines from HDM model with notations and principles from object-oriented design methods. The difference between OOHDM and HDM is the use of object-oriented nature. That said, it adds the nature of object oriented techniques from object oriented approach. And consequently, it has the power of object oriented concepts including abstract classes, methods, inheritance, aggregation, generalisation, etc. Moreover, for the construction of interface design, it uses a powerful abstract interface model (ADV charts) which allows the designer to build different interfaces for the same navigational design. Despite this, there is no CASE tool available for the development of both interface design and translation of models to actual implementations.

One of the main advantages of OOHDM is that different navigational models can be derived from the same conceptual domain design, which can potentially add different perspectives for different user views. However, generally speaking though, since the detailed design of the OOHDM can be considered to be complex, the development of the hypermedia system lies on the background skill and knowledge of the author and the design environment (Nanard and Nanard, 1995). This means the author should not only be able to organise and structure different object-oriented models, but also should also be able to observe and evaluate hypermedia structures in each design step. RMM, which will be discussed in next section, differs from OOHDM in that RMM is mainly based on hypermedia database applications whereas OOHDM can be applied for both retrieval-oriented information systems and other hypermedia systems. OOHDM also provides the development of interface objects that can be visualised in various ways, whereas RMM concentrates on navigational features.

3.5 Reversing engineering using RMM

The RMM is suitable for applications with hypermedia front-ends of database or legacy applications. Hence, SEG Web site is not well suited to the RMM method since the data remains unchanged over time. However, the main idea of this chapter is to evaluate all the different design and construction phases using various hypermedia design methods. Consequently, the new design of SEG page can obtain benefits from RMM design guidelines.

Once the analysis of the problem domain was completed, the reverse design was started by reversing the current Web site into Relational Management Data Model (RMDM). The following steps are carried out during the reverse engineering process.

1. Each section is grouped based on RMDM modeling primitives. As describe in previous chapters, RMDM modeling primitives consist of E-R domain primitives, more detail slices and associative relationships. Figure 17 shows the simplified version of the existing data model for the SEG Web site. It contains the abstract view of the conceptual model for the SEG Web site. Many problems have been encountered to the structure on information during analysis of the current design. 'File Structure' for example, consists on three types of different accounts. In current Web document, the information for the first two accounts (i.e. Unix, and Novell) is embedded in 'File Structure' and the information for third account is anchored as another node.

2. The second step is the construction of navigational structures based on ER design. This construction is entirely based on the structure of current Web site. The main purpose is to analyse the current navigational structure and find out the ways to improve the navigation. This includes grouping of information objects and attributes, designing of navigational access structures based on the relationships made in previous stages. After analysing the current navigational structures, many inconsistencies have been found in the navigational structure of current design. As shown in Figure 18 for example, the current design only uses indices for all links for nodes such as *intermediary tasks* and *project phases.* For this instance, guided tours should be applied, from the student's point of view, to go though intermediary tasks and phases. In Analysing the current navigation design, another interesting issue has raised, namely: the inconsistent layout of attributes within each page. Although this issue is more relevant to interface design, it makes the user disoriented.

3. After the current design has been produced, the design is transformed into new structured

design using RMCase (Díaz and Isakowitz, 1995), which is a CASE tool for RMM design method. This includes the creation of new ER designs, slices and navigational links with the assistance of RMCase. The new conceptual design was implemented to simplify the conceptual design. All necessary nodes are grouped and linked according to their relationships. Figure 19 shows an abstract view of the conceptual design of SEG Web site. The main purpose of showing this figure is to illustrate the conceptual design, not the features of RMCase which are fully evaluated in the next chapter. Due to the lack of support from the tool, there is no hierarchical view for the slices and attributes of each entity. Figure 20 shows the new navigational design for the SEG Web site.

4. Once the basic informational structure and navigational design has been completed, the new design is tested and evaluated. In comparing the two resultant applications form RMM and OOHDM, most of the structures and navigation patterns turn out to be very similar, although there are a few differences. For example, the application produced by the RMM is simple and can present information in structured manner, while the resultant from OOHDM is more functionality for each object with the design.

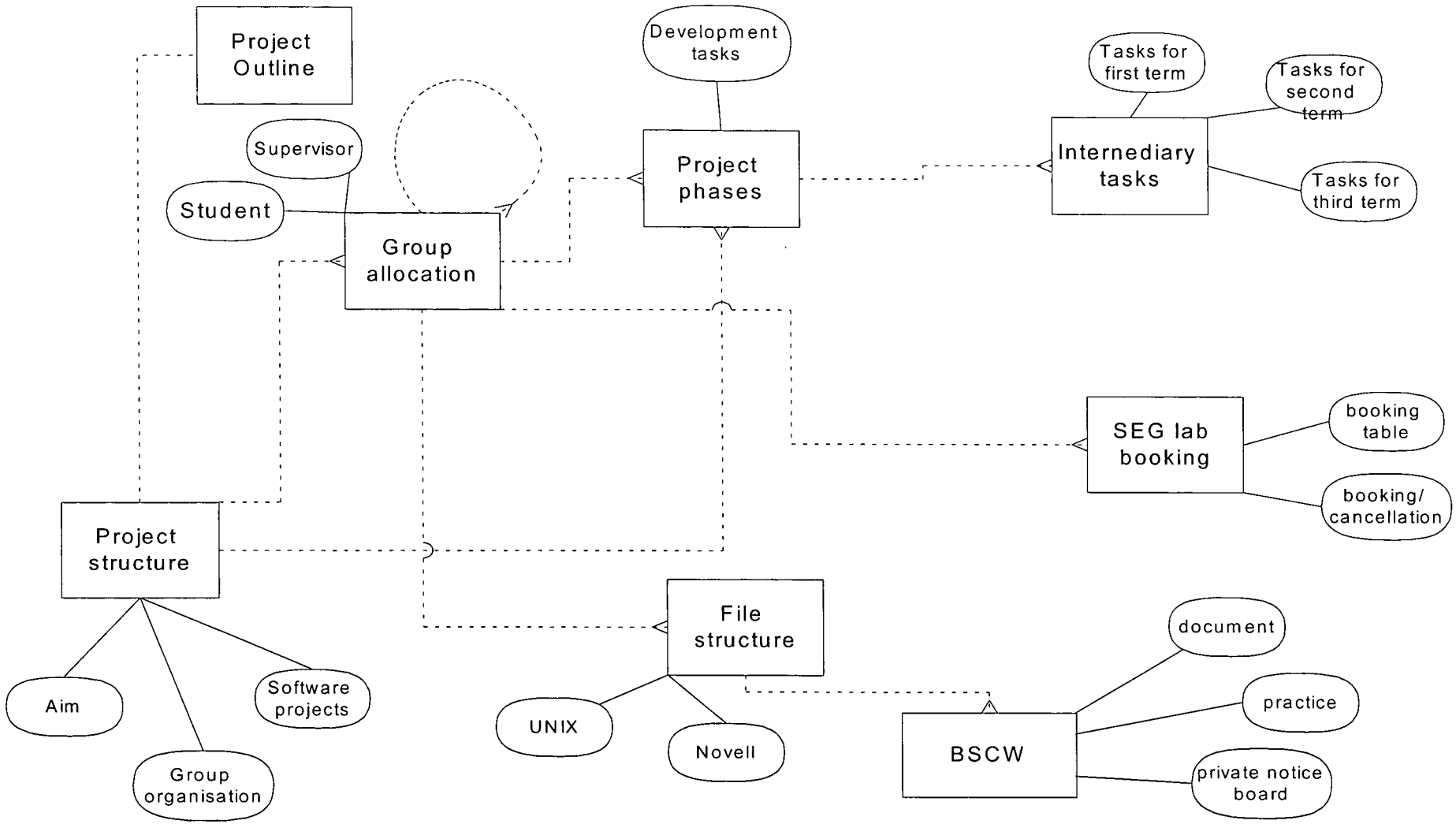


Figure 17 ER diagram of conceptual model for SEG Web Site

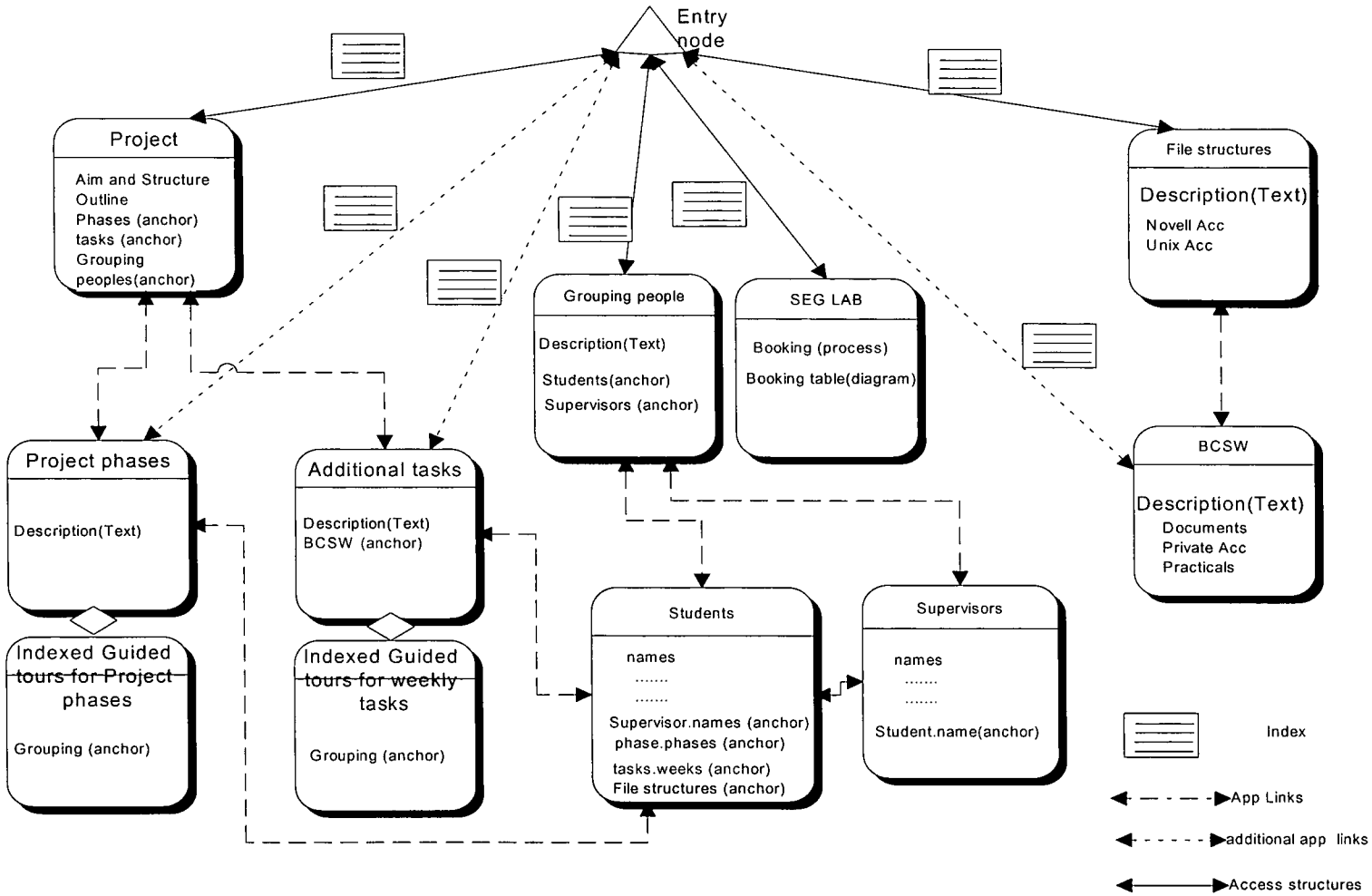


Figure 18 Navigational diagram for SEG Web Site

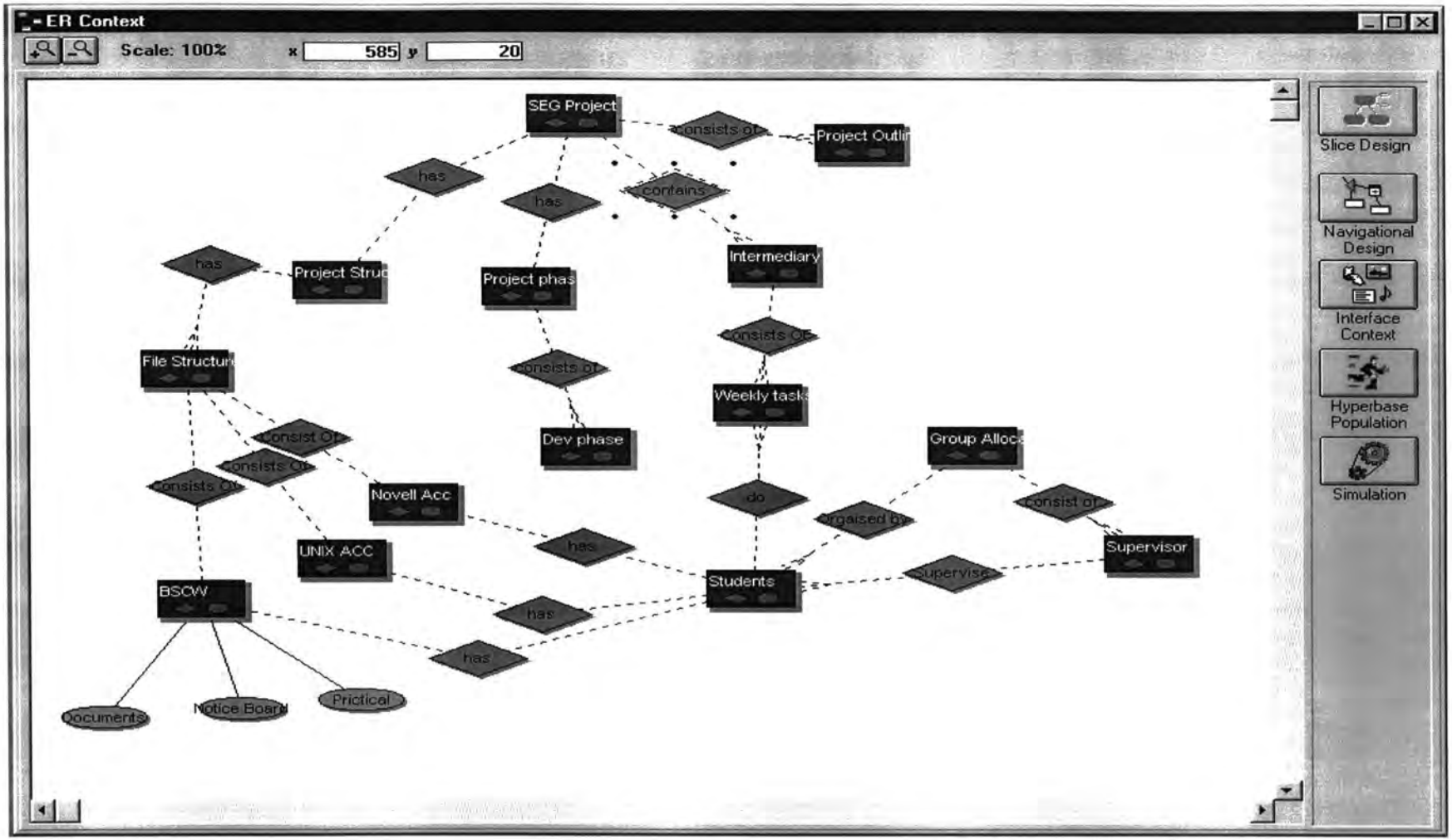


Figure 19 New conceptual design for SEG Web site using RMCase

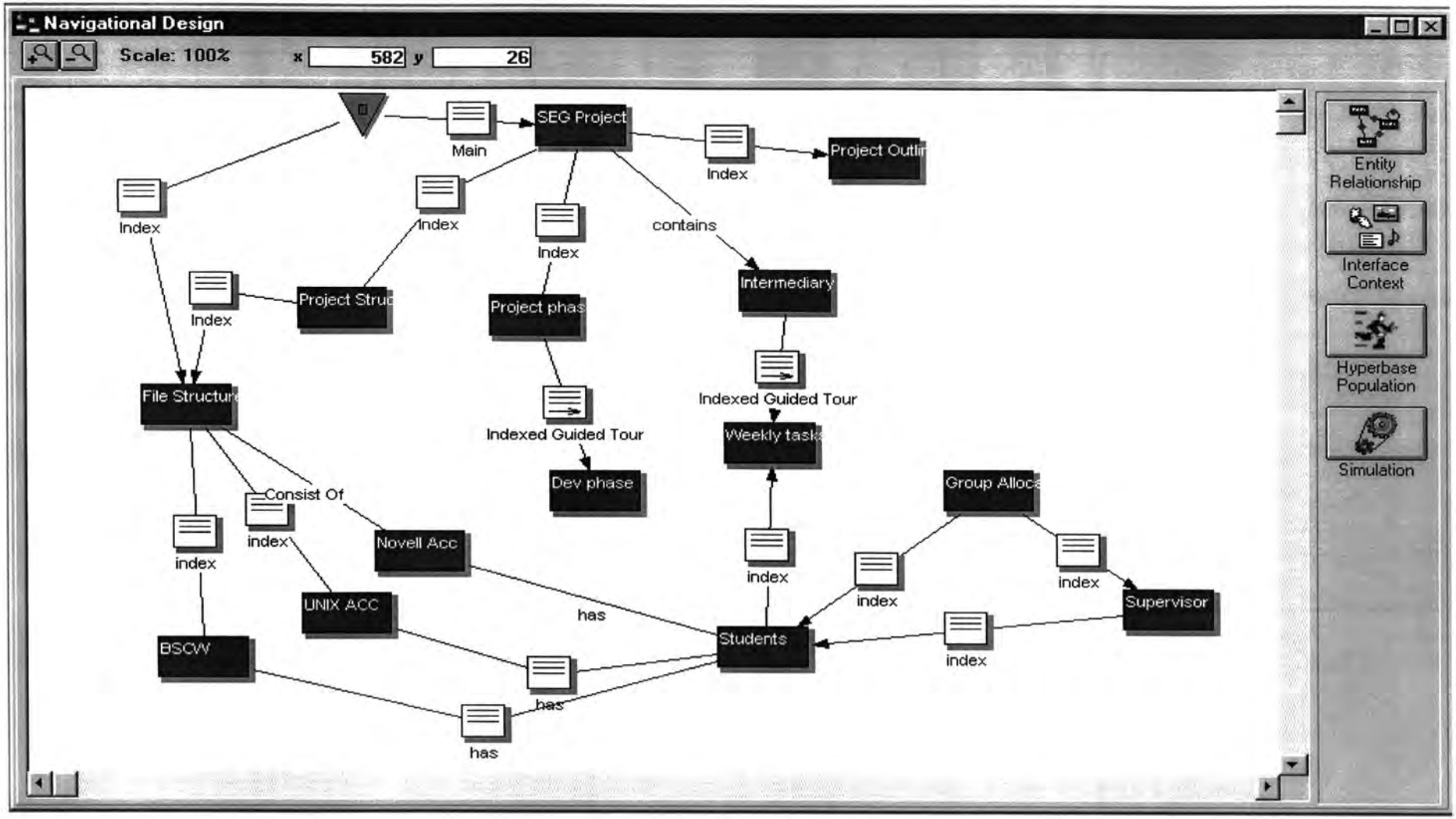


Figure 20 New Navigational design for SEG Web site using RMCase

3.5.1 Evaluation of resigning the SEG Web site using RMM

Since the aim of RMM is the development of systems that interact with databases and which are integrated with other various information systems, SEG was not well suited with RMM design methodology. RMM is only suited to the hypermedia development of large and dynamic database information systems. SEG Web site was developed without any formal hypermedia design approaches and there is redundant data and very little use of hypermedia navigation. However, during the redesign stage, using RMM design method, many bugs and inconsistencies were eliminated in terms of grouping data and navigation. The use of RMM Slices helps the design to split information entities into meaningful group of attributes. This eliminates the disoriented information, which is undesirable for users. The use of RM-CASE, that supports RMM design and development, also assists the redesign of SEG Web. Since RM-CASE tool is still a beta and incomplete version, no user-interface design can be constructed. But the use of General tools such as HTML editors can offer some support for interface design implementation.

3.6 An overview of the design and implementation of the SEG Web site

As an overview, the above section shows the design and implementation of new SEG Web site and points out the important issues during the design process. It also shows the use of two design methods and discusses the way to implement new design prototypes using hypermedia design methods. The new SEG design has been shown and the SEG administrator found to be very useful.

3.7 Summary of Main Points

The main contribution of our work is to assist the designers and developers to be able to choose the right method and tools for any specific application. We have described and tested major hypermedia design models and explored their benefits and limitations. We also described the features of hypermedia, which can then be applied for the development of WWW applications. We also described how hypermedia application and models are related and illustrated their advantages and limitations including different notations and other evaluation criteria.

In this chapter we have reverse engineered the currently developed (SEG) Web site with hypermedia design models and to evaluate the improvements in terms of quality and richness of functions. We also looked at ideas of different navigational views for different collection of users and talks for the same conceptual design. Currently, WWW sites and applications

are implemented without any formal design approaches and constructed mainly in an ad hoc manner. This research is leading to designing new models and analysing existing hypermedia design modél based on their usability criteria. In next chapter we will be looking at different design requirements for designing a new CASE tool to support the development of Web documents using structured design with complete life cycle (i.e. full design stages).

4. Requirements for a New CASE Tool and its Design

4.1 Chapter Abstract

The purpose of this chapter is to analyse the requirements for the development of an integrated hypermedia CASE tool to support the development of large and complex hypermedia applications. The primary objective of the project is the construction of CASE environment that can produce Web based documents, which are maintainable and updateable. The target is ranging from end users to organisational designers and authors. The tool is planned to be a prototype, and also to be completed within two months. Since the development time is limited, the tool is only focused on the high level design of the applications and supports for the actual implementation are omitted. Later in this chapter all the requirements and design issues will be discussed. Furthermore, it will describe currently existing CASE tools and their features, including benefits and limitations.

4.2 The need for a new CASE Tool

Orgainsations and institutes are employing a few Web masters to construct Web-based applications without using any formal hypermedia design methods. However, as applications become larger, designers have encountered various problems, mainly application design and maintenance. Most of the design tools available on the market (e.g. MS Frontpage, etc.) can assist the authors to design only for an individual page without any logical structure as a whole. There are very few tools that allow the designer to construct Web sites using different design and development stages based on structured hypermedia design methods.

Therefore, we are surveying different ways to assist authors to apply structured methods for hypermedia design activities. We have also analysed existing methods and their supported CASE tools as a part of the background study. These include database-oriented methods and object oriented methods. We have also identified basic fundamental development stages for designing Web-based hypermedia applications in chapter two. As a part of this MSc research project on Web based hypermedia design methods, a prototype computer-aided environment CASE tool is to be produced based on structured design methods.

4.3 Current tools and design methods

We have reviewed currently available methods such as RMM (Isakowitz, Strohr, 1995), HDM (Garzotto, Paolini, 1993), OOHDM (Schwabe, Rossi, 1996), HM-Data (Andrews, Maurer, 1993), and many others. The developers of some methods have provided with CASE tools to support their methods for the development of hypermedia applications. Before identifying requirements for a new CASE tool, we examine two CASE tools namely, HM Card and RMCase, and discuss their benefits and limitations in the following section.

4.3.1 HM Card

HM-Card is a set of tools based on HM Data Model. HM Data Model is an object-oriented hypermedia database model presented by Maurer (Andrews, Maurer, 1993). It is designed to create applications such as Computer Support training, multimedia authoring and presentations, and other information management systems. It consists of a set of tools for the construction of applications. Although the purpose of this tool is to produce standalone hypermedia applications, the resultant applications can be integrated or transformed into HTML documents or use the HM Card viewer as a plug-in for the Web browser. It consists of three main CASE environments as three different programs, namely: HM-Editor, HM-Linker, and HM-Viewer, to provide the designer for designing, linking and view of hypermedia application prototypes. Each program performs different sets of functions.

4.3.1.1 HM-Card Editor

Editor is a part of the HM-Card tool set for creating *objects* and *S-Collection*. S-Collection is the nested grouping of objects which can be reused. Detailed information is presented in (Andrews, Maurer, 1993). The Editor provides the author with powerful facilities to incorporate existing materials (text files, graphic images, sound, etc.) into the presentation being developed. With the HM-Card editor the designer can construct the conceptual layout of the application. It also supports the hierarchical structure of components, and can be nested. One component or object can have another component and can be reused. Furthermore, the tool can assist the designer to construct design layout and behavior of each object. Hence, each attribute within S-Collections has its own functions and representation features which can be nested.

4.3.1.2 HM-Card Linker

Once the objects and collections are constructed using the editor, the author can use the linker to create navigational links. Based on the HM-Data model, the linker allows the author to compose the collections in four main kinds of ways, namely *envelope*, *folder*, *menu*, and *free* links. Each S-Collection has an entry collection called *head*. A collection can *zoom in* and *zoom out* the view or edit the attribute objects of that collection. These S-Collections can then be grouped into a set of collections called *Pages* to be linked together to form a hypermedia database and maintain the integrity of all links created. Each page or S-Collection can be complied to test the visual presentation or behavior of the objects. These S-Collections are similar to 'slices' in RMM. However, they are based on object oriented techniques and they can communicate using interface functions.

4.3.1.3 HM-Card Viewer

HM-Card viewer is a windows-based standalone viewer to view finished prototype applications. The viewer can be associate with the Web browser to view applications produced by HM-Card.

Therefore as a summary, the tool includes several important features for developing hypertext applications. They are:

- It includes an editor for creating object (i.e. text, graphics and animation).
- It supports reuse of constructed objects or collections.
- It can be stand-alone or in connection with the Web with plug-ins to Web browsers.
- It allows layered navigational links and cross-references to other material.

4.3.1.4 Limitations

Although objects and collections are reusable within the tool, they cannot be exported for other purposes. Most importantly, the design made with HM-Card cannot be integrated with current Web-based HTML documents. Since the tool is MS-Windows based program, it is platform dependent. However there is also Java based version which can be compiled under Unix platforms.

4.3.2 RMCase

RMCase is a CASE tool for designing hypermedia applications. It is based on the concept of RMM design method. The method is mainly based on Entity Relationship (ER) model and consists of entity, slices and relationship between entities. RMCase consists of the following features for the development of application prototyping.

4.3.2.1 GUI design interface

RMCase allows the designer to construct the entities and relationship using a graphical editor. It also provides the designer to construct a conceptual design by creating entities, slices, and attributes on the graphical editor. Therefore the designer can create, modify or remove objects using graphical canvas. Unfortunately though no hierarchical representation is supported. The tool provides different environments with separate windows to allow the user to go back and forth between different stages.

4.3.2.2 Design Stages

Since RMCase is based on RMM design method, it allows the user to construct the model using different design aspects of RMM, namely, ER Design, Slice Design, Navigational Design, User Interface Design, data population and simulation. Since it is only a prototype, not all of the design steps have been implemented in the current version.

4.3.2.3 ER Design

RMCase allows the user to create entities and their attributes using drag and drop facility. Relationships can also define according to the user needs. Hence, in general the tool allows the designer to build conceptual design based on Entity – Relationship ER model.

4.3.2.4 Slice Design

Once the Entity Diagrams are created, the user can construct Slice diagrams to group one or more attributes within an entity. Each entity has a head slice as an entry point for that entity. The user can zoom in and zoom out into each entity to see the detailed attributes, slices and relationships between slices.

4.3.2.5 Navigational Design

In RMCase, navigational links can be specified by indices, guided tours or indexed guided tours. The users can also specify the relationships between the entities by giving two selections as one-to-one or one-to-many. However, only one navigational model can be made based on a conceptual design.

4.3.2.6 User Interface Design

User interface design environment is not implemented in the version that is currently available.

4.3.2.7 Data Entry

The RMCase consists of an internal data entry forms to be entered the content of information. But, the relationship features are not properly implemented, and the designer has to synchronise manually.

4.3.2.8 Simulation

The designer can perform the trial run the prototype by using the internal simulation function provided by the tool.

4.3.2.9 Limitations

Since RMCase is only a prototype, it is not relevant when developing any large and complex applications. The tool is not fully functional and some of the critical functions, such as user interface design, forms, and object templates are not implemented yet. There is no connection between (i.e. export and import) between other objects and Web designers.

4.4 Requirements for a new CASE tool

Having analysed the currently available tools and design methods, we have proposed a list of requirements for the design of a new CASE tool. Firstly we identify the high level design requirements and detailed attributes of each high level design.

4.4.1 High level Design requirements

- The system should guide the user through different design stages. These stages are the necessary basic stages to construct hypermedia applications.

    - Conceptual Design: The design environment should support easy to use visual implementation of objects, including object creation, grouping and relationships.

    - Navigation Design: The user should be able to construct navigational structures easily and clearly.

    - User Interface Design: The tool shall have shall allow easy construction of interface design including forms, templates and object layouts.

    - Conversion and implementation: The tool should be able to export diagrams and generate CGI gateway scripts according to application requirements.

- A relational database should be created to store persistent objects, link information, classes structures, entities, templates as well as inheritance diagrams that can be reused in different stages.

- The program should generate HTML documents according to the design created by the designer's design specifications.

We have proposed a set of detailed criteria that should be considered for the development of hypermedia applications. They are as follows:

Range of application

The tool should facilitate the construction of applications ranging from the design of simple and small Web-sites (e.g. personal home pages) to large and complex Web based applications (e.g. organisational Web sites). The designer or user should be able to construct and edit wide range of applications in a straightforward manner using a set of stages with

GUI design environments.

Method

To construct different applications with different level of complexity, the tool should be flexible enough to support designing database oriented applications. Therefore according to user needs, complexity and nature of the application, the user should be able to follow methods relevant to these two main application types.

Design stages

The tool should be able to guide though the development stages that are presented in chapter two. It should also allow the users to choose different styles of development activities based on the application. The basic design stages are presented in Figure 21. The details of design environments for each development stage will be described later in the chapter.

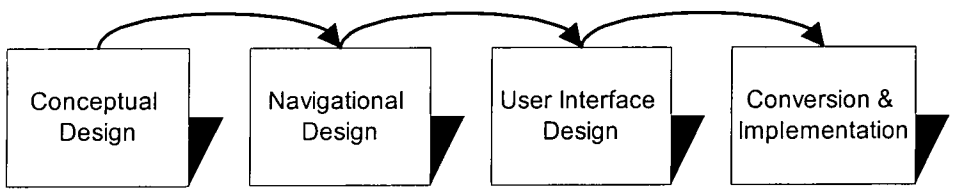


Figure 21 Hypermedia design stages

Data handling

Each project made by the user should be stored in a database that can handle a large amount of data including multimedia materials. The database will consist of two main parts, one for storing objects and one for storing navigational patterns. When generating HTML documents, the tool should allow the designer to give directory paths to point for images that are added.

Structure, hierarchy and Visual development environment

The tool should provide consistent structure and hierarchical design for all the objects across all development stages. The user should also be able to visualise all the relationships between the objects at every layer within the hierarchy. The tool shall also consist of structured graphical user interface design, which allows the user to manipulate visual objects easily.

Object grouping, forms and templates

The tool should provide attributes or objects grouping in a hierarchical manner. It should also allow the user to model high level entity templates or class templates, to allow the user to easily generate any instance of the domain oriented data composites having predefined semantics. For database instances, the user should able to customise the presentation of every attribute with forms and templates.

Types of links

During the construction of navigational links, the tool should facilitate the user to create any kind of link structures (i.e. index, guided tours, etc.,) and link information should be externally stored from data content.

Link structures

The tool should allow the user to construct links by layers and these links should be encapsulated within that layer within the semantic network. Objects and links are stored apart and linked according to their relationships. This idea is also presented in (Gronbæk, Bouvin, 1997), (Takahashi, 1997). Accordingly, in the final HTML documents, for example, there are only internal links embedded within the contents and external links are explicitly defined with other HTML frame or other part of the document.

Navigational structure & views

The tool should let the user to construct many navigational views based on a conceptual design to fit with different set of users. Hence, the designer can compose different structures for different user views. The detail description in regard to navigational designs will be made later in the chapter.

Storage Structure

The tool should allow the user to enter data (i.e. content entry or data population) into the database or as persistent objects using forms or templates.

Most importantly, the user should be able to fall back into conceptual design from any stage during the development lifecycle. The above requirements for the CASE Tool are focused on the development of large scale Web-based hypermedia applications.

4.5 New CASE tool design environment

Since the tool itself will be based on hypermedia design structure, all design environments will be linked, which allows the user to cross-reference their work between different stages. Its design structures or primitives are therefore based on the functionality of hypermedia.

Figure 22 shows the high level design contexts of CASE tool. The tool will consists of three main functions, namely projects, design environments, and database. In general, once the user creates a new project, the tool should provide a series of development environments through the different hypermedia design stages. Firstly, the user should be able to construct *conceptual design* for any prototype based on the application requirements. The many *navigational design* structures may be developed using the conceptual design from previous stage. Once the navigational design is completed, the tool should provide the design environment for user interface applications based on the functionality of HTML. As the user constructs along these different stages, the tool should store all design objects in the database for reuse. Therefore all objects from earlier stages can be applied in the later stages. The detailed description of how the tool should support the design process is described below.

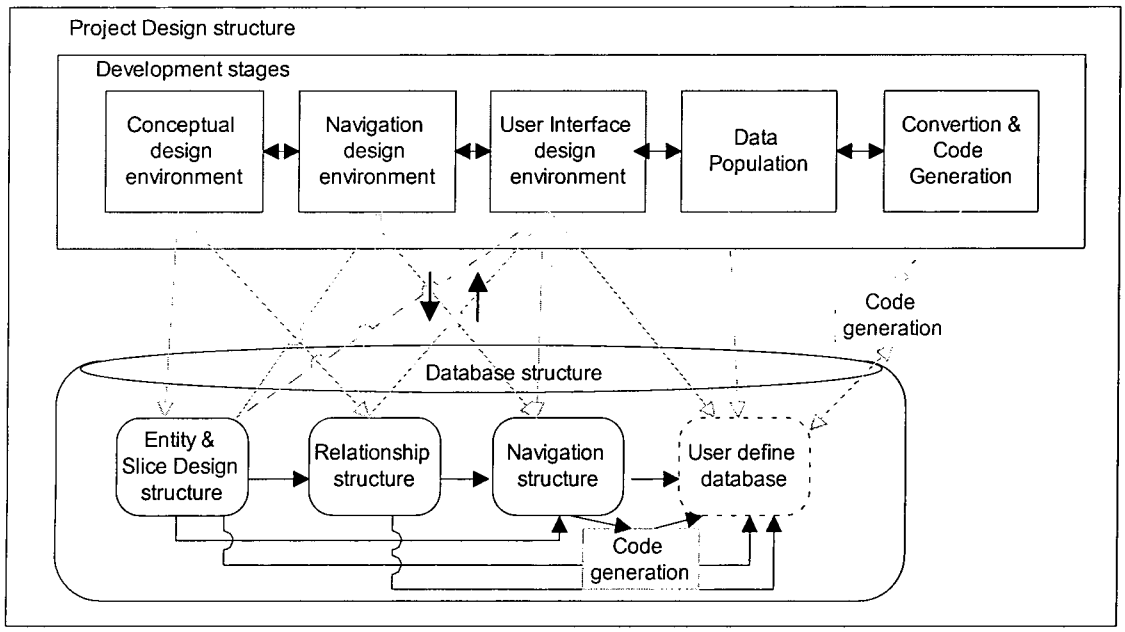


Figure 22 High level CASE tool design structure

4.5.1 Conceptual design environment

The tool facilitates the construction of conceptual design based on database design diagrams. The main objective to allow the user to capture the characteristics of the application domain by prototyping graphically.

The tool consists of a set of design primitives based on Entity Relationship (ER) model. The four main primitives of the conceptual design are entity, slice, attribute, and relationship.

Entity - The tool should allow the user to create, modify, remove and, most importantly, reuse the entities within the design environment. The tool should not only allow the user to visualise the entities on drawing canvas, but also should present with an entity tree hierarchy, so that all the reused entities can be clearly seen. Each entity that the designer creates should be stored in the database for further reuse in the later stages.

Attribute - When the designer creates a new entity, the designer should be able to create a set of attributes related to that entity. Attributes can be any media type such as text, graphics, sound, video, etc.

Slice - The designer can group the attributes by using slices. A slice can also contain other slices which may be re-used a multiple number of times. The tool should also support m-

slices (Isakowitz, Kamis, 1997) which can collect different slices from different owners (entities) by using matching keys. Each slice information should also be recorded in the database to be reused in the following stages.

Relation - Since the tool is based on database design model, the relationship between entities is crucial. The designer should be able to provide detailed information about the relationships such as, relationship types, cardinality, grouping, etc. Therefore these details can be reused as derived navigational links in the later stages of the design process.

Figure 23 shows a simple scenario of information structuring of a computer science department. Within the conceptual design environment, the designer should be able to construct a high level conceptual design including the details in each entity. During the development, the designer may zoom in and zoom out to manipulate each entity and relation.

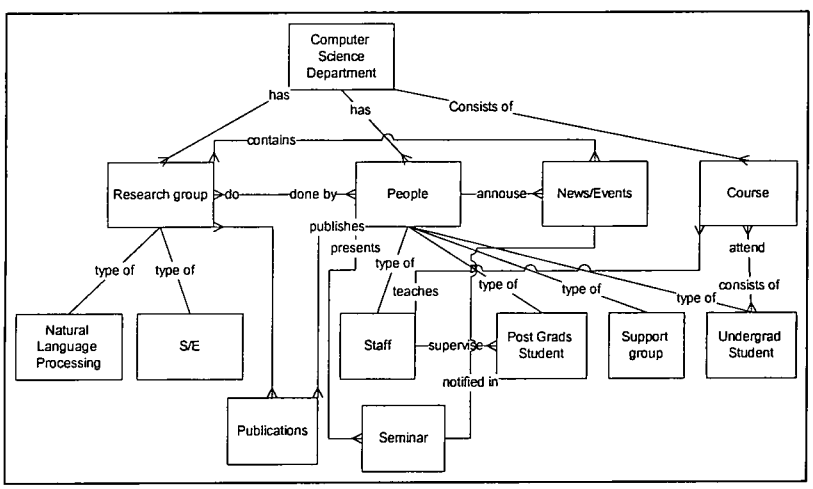


Figure 23 Simple conceptual design for Computer Science Department

The basic elements of the conceptual design environment include the following:

- Graphical editor - allows the development to model entities and their relations on a canvas. It also allows different editing features such as zoom in, zoom out, printing, exporting to other formats, etc.

- Database - records all design objects and editing settings during the development process. Once a project is created, a database and a set of tables are automatically constructed to record every design objects that the user makes. The tool should create all the necessary tables and views for every design context during the development.

For the conceptual design environment, five tables will be created, namely, entity, attribute, slice, relationships, and editor settings. The main purpose of creating these tables for each project is to allow the designer to store the elements of the conceptual design to be reused in other stages or other applications. The following tables show an instance of rows that may be generated during the development of conceptual design. Table 2A shows an instance of entities with entity ID and entity name. Table 2B shows a slice table. The first two slices are owned by the 'Person' entity and the third slice is owned by 'Course' entity. Table 2C shows slice and attributes columns. Since a slice can consist of another slice, 'Type' information is required to distinguish between attributes and slices. For example, slice ID 4 consists of another slice (ID 5) and the type is 'S', which denotes that it is a slice rather than an attribute. Table 2D shows an attribute table, which consists of name columns and various types. Table 2E shows a relationship table between two entities. All the relational information can be transformed into derived navigational links.

Entity ID	Entity Name
10	Person
11	Course

[A]

Slice ID	Slice Name	Owner ID
4	Person Info	10
5	Name	10
6	Course Info	11

[B]

Slice ID	Attribute ID	Type
4	2	A
4	5	S
5	1	A
5	3	A
....	.....	..

[C]

Attribute ID	Attribute Name	Owner ID	Type
1	First Name	10	Text
3	Last Name	10	Text
2	Address	10	Long Text
4	Picture	10	Graphic
..	..	....	..

[D]

Relation ID	Kind	From ID	From Key	To ID	To Key	Cardinality	Attributes
1	Teach	10	ID	11	Lec_ID	1..n	....

[E]

Table 2 Structure of database tables for conceptual design

- Tree Hierarchy - shows a hierarchy of entities and their attributes. This allows the designer to drag and drop existing attributes and slices to be reused. It also shows a clear hierarchical representation of all design objects within the diagrams. Figure 24 shows, a simple hierarchical tree of a *Person*. A *Person* consists of two child slices, *Name* and *Info*,

and each slice has its own attributes. The slice *Info* consists of another slice *'Name'* as an attribute. The tree view provides the overall architecture of the application that consists of all design objects and their relations.

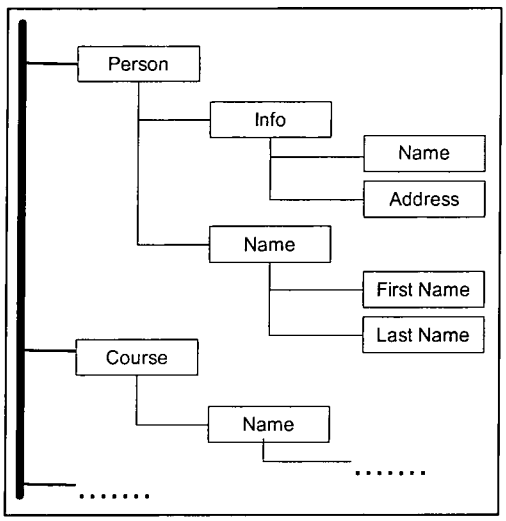


Figure 24 Hierarchy tree view

4.6 Navigation design environment

The second design environment that the tool should provide is navigational design. The tool should generate derived navigational links based on the relationships between entities from a previous stage. The designer can alter these derived links as needed. This alteration requires derived links if they are not relevant for navigation or the designer wants to add or remove additional navigational links.

The tool should also provide additional link features such as the same link from different attributes (i.e. same link is anchored from both a line text and a picture to is same destination). So the user can browse with different perspective.

The essential features that should be provided in the navigation design environment are:

- Database that stores all the navigational structures of design objects at each layer of interaction. All the link information must be stored explicitly outside of their contents in the separate tables.

- The tool should allow the designer to construct more than one navigational structure

(view) to suit different sets of users. For instance, the designer may create navigational patterns from students, lecturers or visitors perspective based on one conceptual design from the previous stage.

- The designer should be able to fall back to conceptual stages for any modifications. Nanard & Nanard (Nanard and Nanard, 1995) pointed out that fallback procedures between different stages are crucial for the development process.

- Like RMM, OOHDM and other methods, the tool should support the creation of different access structures including menus, maps, indices, and guided tours. Figure 25 show three main access structures.

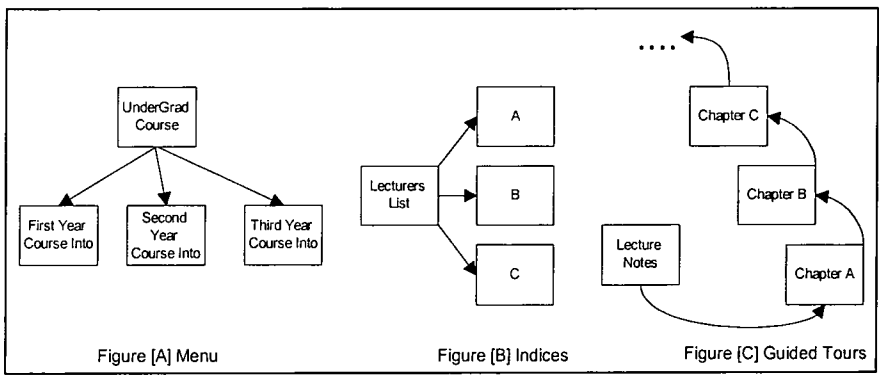


Figure 25 Access structures for navigation

- Most importantly, the designer should be capable of choosing both open and close links. The close links are encapsulated within each entity and can communicate only by relationships to other entities. For instance, the designer is limited to create interconnections of member slices within an entity and can only communicate to other member slices by relations, thereby preventing the the designer from creating unnecessary links, which cause user disorientation. In addition, open links can be added freely by the designer's choice. Adding open links allows the designer to present all relevant information that is necessary.

4.7 User Interface design environment

In this stage, the designer should be able to define layouts of each page. However, the layout and forms that this CASE tool will generate will be a skeleton which can then be fully edited by HTML-editors. This basic layout will provide uniform structures for large number of

pages. Currently, the tool comprises necessary HTML files with basic layouts. However the tool can be enhanced to be able to map each presentation unit (i.e. text, button, selection, input box, etc) and design objects that are stored in the database, and generating CGI scripts as necessary, if time is available. However, these features are supplementary functions to support for the design of hypermedia models. Figure 26 shows a simple layout that will be provided by the CASE tool. It consists of four frames. The 'Title' consists of page title, and may also consist of additional graphics. The 'link' frame contains all the necessary links that are derived from relationships with this page. The 'description' frame presents the content of the page. Finally the 'navigational bar' provides navigational assistance buttons such as 'home', 'index', 'next', 'up', etc.

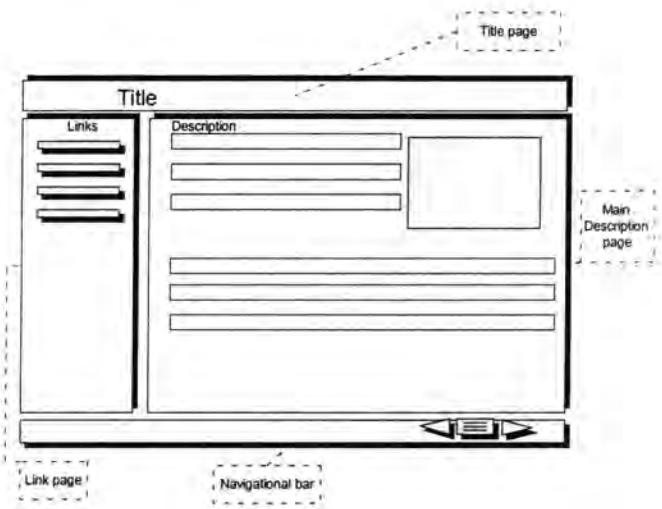


Figure 26 Sample HTML template with four frames

Although the tool would generate skeleton pages and HTML template files for the development, the designer still needs to use consistent graphics, text and layouts for the whole application framework. The designing visual graphics should also be emphasised equally within structure of the application. The designer should construct different styles of graphic elements to reflect the levels of importance, based on user needs and frequency of access.

4.7.1 Data entry

The tool should generate internal data entry forms to enter the contents of the information. These data entry forms should be dynamically based on the entities generated during conceptual design. However since the tables will be generated using Dbase or Paradox formal the designer can choose external applications.

4.7.2 Conversion and Implementation

Once the user interface design environment is constructed, the tool should be added with additional functions such as exporting diagrams so that the use can reuse that prototype in other authoring tools. The tool shall also be provided with the facility to generate skeleton HTML files that allows the designer to modify as his/her choice.

4.7.3 Programming language

The tool will be implemented using Borland Delphi using Paradox relational database. There are three main reasons for choosing this development tool. Firstly, Delphi includes necessary GUI features such as design canvas, tree view, etc. features that are required for this CASE tool development. Secondly, the tool requires the creation and modification of database structures. Delphi provides an easy to use database engine that is based on Paradox. Thirdly, Delphi is an object-oriented language, which allows the construction of design objects easily and efficiently.

4.8 Design Overview and summary of main points

As an overview, we have outlined the requirements and showed the detailed design for implementing the new CASE tool to develop Web based hypermedia applications. We have also discussed the development stages that require implementation while constructing the tool. In the next chapter the detailed features of the tool will be identified together with the description of design environments provided by the tool.

5. Tool Implementation

5.1 Chapter Abstract

The aim of this chapter is to describe what Structured Web Development tool (SWDT) can be provided to Web authors and the features provided by the tool. Since the development of a complex Web based application involves a variety of activities, such as data storage, navigation, and presentation units, this CASE tool is implemented differently from other tools which are intended to construct traditional software applications. This chapter shows the step by step design development stages provided by the tool, together with a sample application to illustrate the use of these design stages and functions that have been implemented. Throughout this chapter an example will be used to discuss various features included in the tool.

5.2 Introduction

As presented in chapter two, the development of structured Web-based hypermedia application requires one or more formal design methods, development stages, and many other activities such as feedback loops, incremental design process, etc,. (Nanard and Nanard, 1995) We discussed and proposed a set of basic development stages to construct hypermedia applications in chapter two. Later in this chapter we will describe how the features provided by the SWDT tool fit these basic development stages.

The SWDT tool is a CASE tool that consists of a set of computer-aided design environments to support the designer to design and develop Web applications using hypermedia design methods. The tool is based on development stages that are presented in chapter two. These stages will be discussed later in this chapter. After surveying different existing design methods such as HDM, RMM, HM-Data, SHDT, OOHM and OOHDM, the tool is added with several functions from these different methods. For instance, the use of Entity Relationship (ER) model, which is also based on RMM model to define nodes and their relationships. The tool also applies the different navigational views from one conceptual design to support different users, which is presented in OOHDM method. The detail functions will be discussed later in this chapter.

The tool focuses on the architecture and functions of the application, rather than on the

appearance of each information unit or page, such as graphics buttons or other text styles. In other words, the tool is constructed mainly to support information structuring and linking of the resources, with the concern of modularisation and reuse. One of the main objectives of the tool is to produce HTML codes as well as to export design specifications that can be reused further. Therefore, the outcomes on the tool can be applied with any Web browsers. It consists of a set of development stages that enable authors to build powerful and modularised applications visually. 'Visually' means that the author can visualise the whole structure of the application by zooming in and out during the construction. The following section presents detailed features of the tool, its functionality, design environments and instructions of how to work with the tool.

5.3 *Tool Overview*

As an overview, the Structured Wed Development Tool (SWDT) is a CASE tool for the design and construction of Web applications. It consists of a set graphical editor and methodological based development stages to be able to generate prototypes for the Web based applications. The followings are the main features of the SWDT CASE tool.

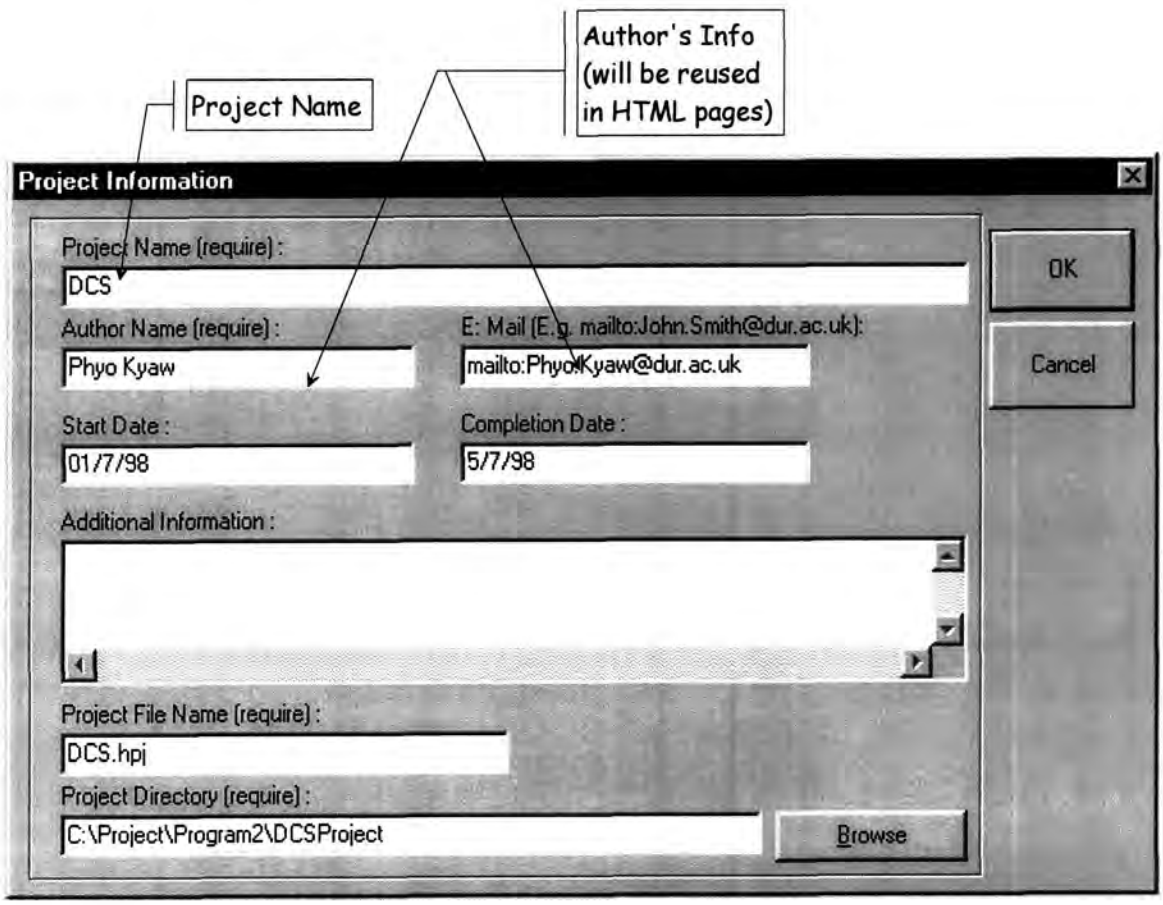


Figure 27 New project information dialog box

- The designer is able to create new, save, and open projects during the development of the application. As shown in Figure 27, once the designer creates a new project, he/she can enter project information. Some parts of this information, such as author name, e-mail, published date, etc., are reused during the generation of HTML codes. Accordingly, it is necessary for the designer to enter the information.

- The designer can follow the methodological based development stages and move back and forth as they progress in their tasks. The detailed development environments will be discussed later in the chapter.

- The prototyping features of the tool enable the design to easily conceptualise the application domain. Furthermore, it also includes a hierarchical tree view for different components within the design environments. This leads to structured and clear application prototypes.

- The tool uses HTML frames as templates to set the layout of the information nodes and their links. This gives structured and consistent Web pages. However, since the tool focuses only on information structuring, the layouts that are generated do not include any graphical details. Therefore the designer has to use other HTML editors or directly edit the HTML codes to enhance the user interface design.

5.4 Tool Architecture

From the methodological point of view, the tool is based on Relational Management Model RMM (Isakowitz, Strohr, 1995) with enhancements. As discussed on chapter 2, RMM is a model for the design and construction of hypermedia applications. It consists of seven steps and based on Entity-Relationship (ER) model (for a more detailed elaboration, we refer the reader to chapter 2). However, the tool consists of five necessary steps that are shown in Figure 28.

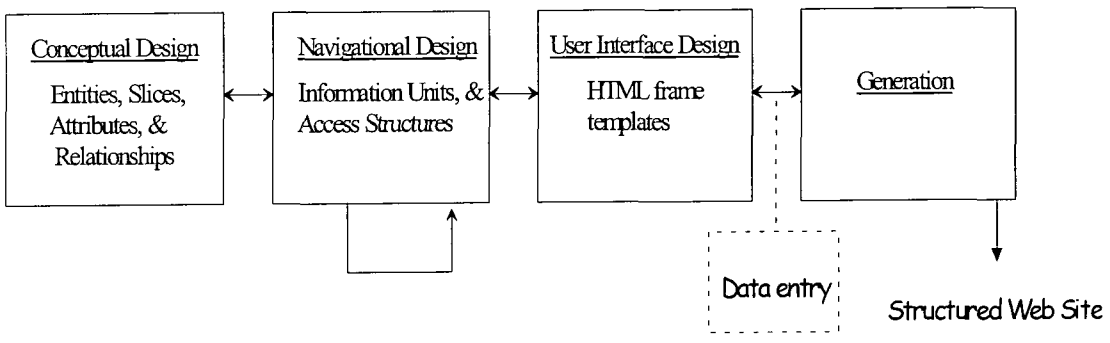


Figure 28 An overview of the tool architecture

The architecture of the tool is composed of three main activities. They are:

- To allow the user to easily follow the *formal design method*, which includes one or more methodological steps. In this way, the user can clearly distinguish between the steps and is able to move back and forth during the development. The formal method, during the design phase, helps structure the information domain and lead to clarity and easy information retrieval in the application. It also represents the concepts and notations for each design stage. Figure 29 shows the navigational bar from the SWDT tool. The designer can easily shift between different design stages with the design environment.

- To provide the designer with the guidance of the necessary design *process* through different stages. This means that specifying process guidelines helps authors to avoid inconsistencies and mistakes. The tool supports the designer's incremental activity during the construction of the applications and, most importantly, the feedback loops. The tool is added if different functions such as tree views, guided lines of the processes and design stages are needed to facilitate design process.

- To assist the designer for *organising* and *structuring* different elements of the design during the development. Structuring the elements of the design also lets the designer to evaluate, analyse and reuse design activities. As the application becomes larger, organising and communicating between different stages or different activities with the same stage is vital for the design process and the quality of the application structure.

The tool consists of an integrated design environment, which includes several design environments, for different development stages in the life cycle. It allows the user to draw a graphical representation of a data model, construct one or more navigational structures for different user views, selection of user interface templates, and data entry (data population) to develop a working prototype. In general, the development of Web applications involves a variety of people with different skills who are responsible for different aspects of the application. They use different standard and tools such as JAVA, databases, CGI programming, graphic tools and HTML assistant tools. The tool, however, is implemented as a prototype design and it only focuses on the structure of the application. Therefore one of the main objectives of the tool development is to assist the designers clarify complex

information domains and make it clear and accessible to users to construct structured navigational views.

Figure 29 Design Stages (Navigational bar)

5.5 A scenario for building a Web site

In chapter four, we have shown a sample data model of the Department of Computer Science (DCS) at the University of Durham as an example. This sample scenario will be used to illustrate the use of design environments in SWDT. The data model includes information related to departmental news, member of staffs, students, research groups, undergraduate courses, and other departmental information. There are also different user views such as visitors, students, lecturers and researchers that need to be considered when designing for a DCS Web site.

There are two sections to be discussed. The first section shows the use of design environments in SWDT with DCS a scenario. The second section describes the detailed step by step construction of DCS with the help of SWDT tool.

5.5.1 Conceptual Design Environment

This design environment includes different elements and functions to provide the designer to construct conceptual design. Since the conceptual design is based on Entity Relationship (ER) Model, it uses ER design primitives, namely: Entity, Slices, Attributes and relationships. The main objective of this stage is to conceptualise the data model of the application. Therefore a problem domain can be analysed to understand the domain and determine the scope of the application by ER analysis. During this step the tool is provided with a graphical editor to construct the data model. However other object-oriented approaches such as OOHDM, and OOHM can also be used.

As shown in the Figure 30, the designer can construct the prototype of the application in a structured manner using graphical canvas and hierarchical tree view. For instance, a 'member of staff' (i.e. professor or lecturers) teaches one or more 'undergraduate courses'. The designer can view the attributes of both staff and courses using the hierarchical tree view. The detailed description for the development of conceptual design on DCS will be discussed later in the chapter.

Although navigational views are not derived yet, the designer has to carefully consider different contexts required by different sets of users for navigation. This is mainly because different navigational views require different sets of information groups for different sets of users.

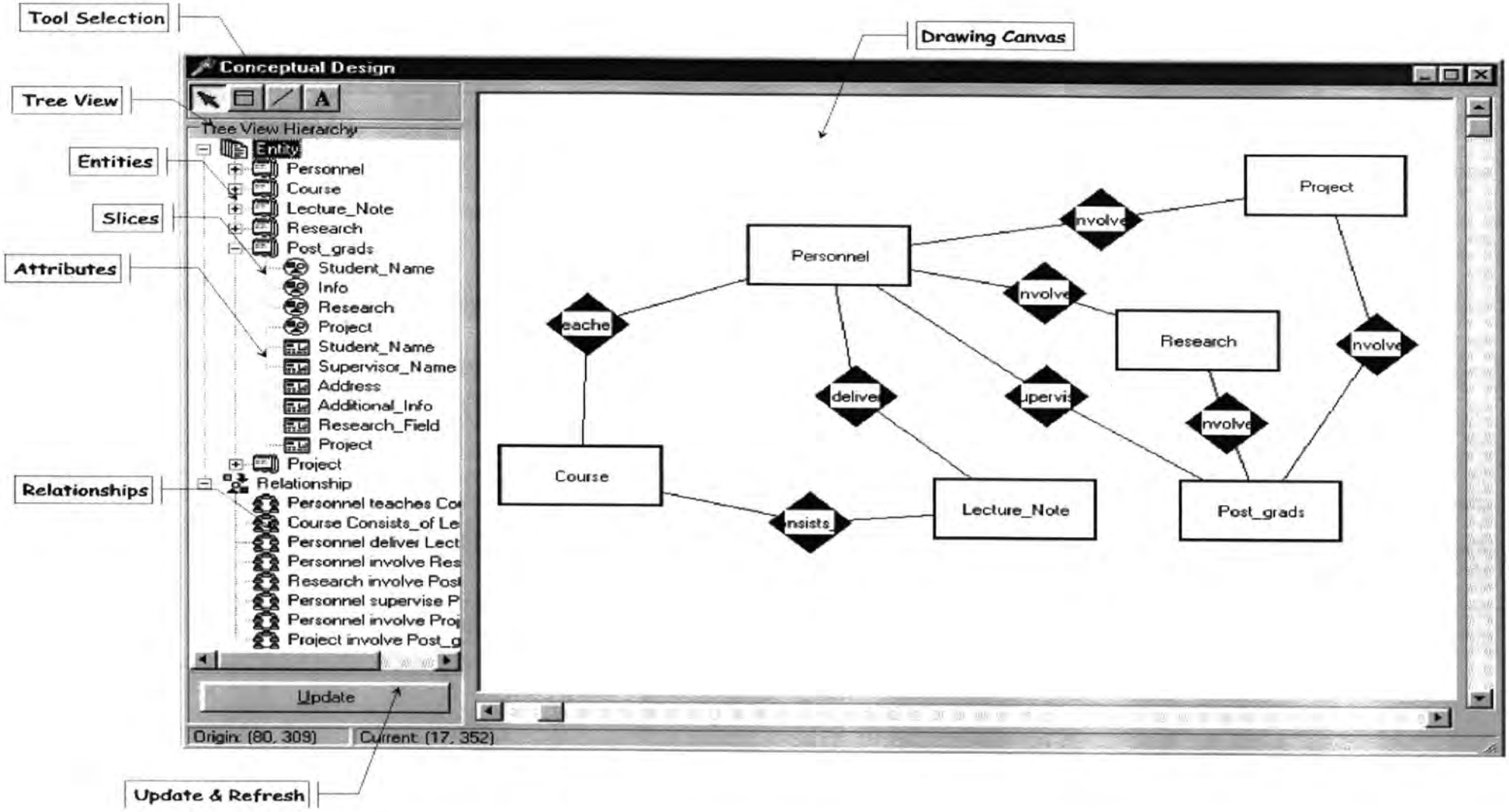


Figure 30 Conceptual Design Environment

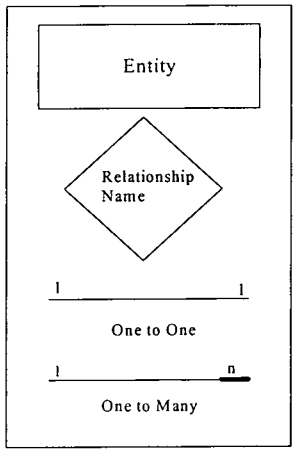


Figure 31 ER Design Primitives

5.5.2 Design primitives

Since the conceptual design environment is based on Entity Relationship (ER) model, it consists of ER based design primitives. Figure 31 show the ER design primitives that are used to construct data models. It consists of four main elements, namely: Entities, Information Slices, Attributes, and Relationships. In Figure 30, the entity 'Undergraduate Course' consists of many attributes such as module name, module code, and other descriptions. An attribute can be in the form of text, graphic, number, etc. Like RMM, one or more related attributes can be grouped into information slices. Therefore a slice is a collection of related attributes grouped together to present information. In SWDT, slices can be nested. Therefore a slice may contains attributes and other related slices. For example the 'name' slice may include 'first name' and 'last name' attributes and again the 'personal info' slice may consists of 'name' slice together with other slices and attributes. An Entity can communicate to one or more entities by using relationships. It can be one to one, one to many or many to many relationships. For example, a 'member of stuff' (or a lecturer) may teach one or more courses (modules) which causes one to many relationships. A relationship can also consist of one or more attributes.

5.5.3 Presentation of information

In SWDT, although the designer can create all entities, slices and attributes, only entities and relationships are shown on the canvas for clarity. Accordingly, the slices and attributes, which are the inner attributes of an entity, can be viewed as hierarchical trees. For the department of computer science example, only entities such as lecturers and courses are displayed on the graph and their attributes such as course name, lecturer name and other

information are presented with tree view just beside the graphical canvas. Additional features such as labelling, zoom in, zoom out, etc., are also provided in every development stages. Furthermore, creation, editing and deletion of entities and relationships can be performed instantly.

In SWDT conceptual design environment, once the designer has constructed the design of the conceptual data model, he/she can instantly enter the content of the data into the database. However the content is stored separately from the structure information, therefore can be reused in any other applications. Furthermore, since the content is independent from the structure, the changes made in the contents do not effect the structure of the application.

5.5.4 Building entities, attributes and relationships for DCS

With SWDT, the conceptual design can be built using entities, slices, attributes and relationships. As described earlier, boxes represent entities and diamonds represent relationships among entities. In DCS example, five main entities are defined : 'Personnel', 'Research', 'Course', 'Event', and 'Project'. Like other Entity-Relationship models, entities includes typed attributes. Relationships represent relation between entities, which can then derive navigational links. Since the relations are base links for navigational views, the relations should be explicitly defined for keys and cardinality constrains. These constrains are important for all kinds (i.e. one-to-one, one-to-many and man-to-many) relations. For example, within 'personnel' and 'courses' relationship, one lecturer may teach many courses (one-to-many). Therefore the key field from 'personnel' must be in every course record that are related. Furthermore, as shown in Figure 32, between 'Personnel' and 'Research' relationship, one person may involve in more than one research areas and one 'Research' may be involved by many persons. Therefore we have to explicitly define many-to-many (n..n) relationship and set two key attributes. Once many-to-many relationship and keys have been set, the tool automatically generates a table to store two key attributes and to link two entities. The SWDT provides with the graphical editor (or canvas) which allows the designer to create high-level structure of the application domain, thus reducing complexity and enhancing modularity.

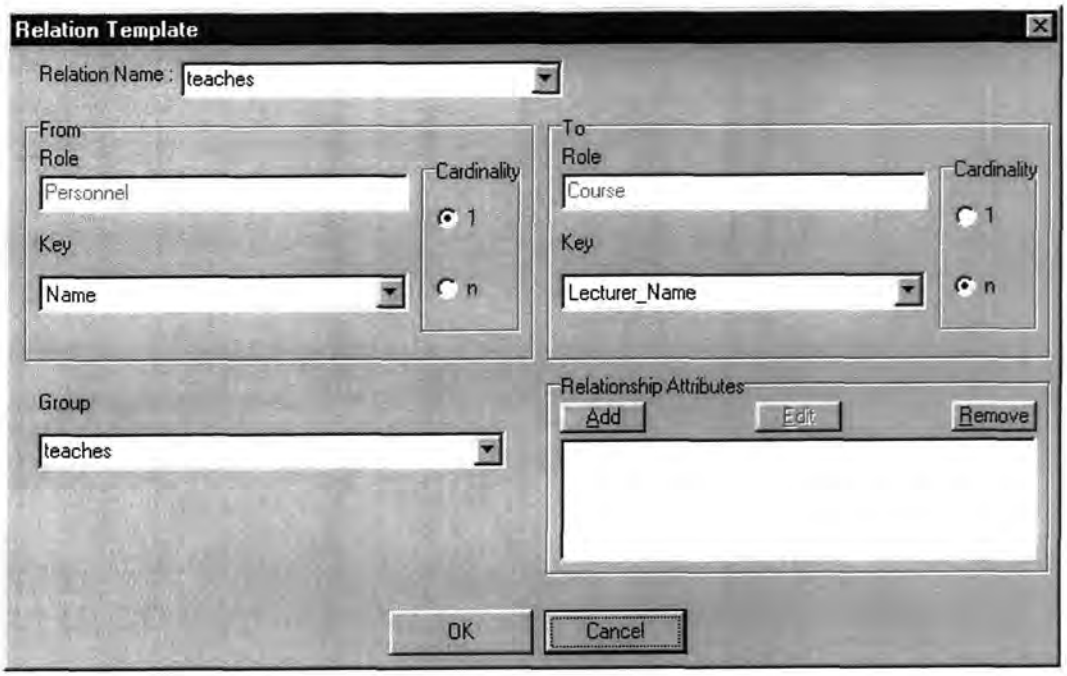


Figure 32 Relationship dialog-box

5.6 *Navigational Design Environment*

Following the conceptual design, navigational design environment is provided to construct navigational views. The designer can identify the different set of potential users and create different navigational views based the conceptual design from previous stage. Since the construction of conceptual design is based on traditional ER model, it is a well-accepted practice in software engineering field. Therefore the designer may easily conduct the analysis and create a prototype. However the construction of navigational design requires careful planning and organising of materials. Furthermore the designer also have to consider have cognitive aspects of the design as well as the need from different users. As Nanard & Nanard have pointed out, the important aspects of the design processes only includes the use of formal methods but also the need for evaluating, organising, structuring during the navigation process. The structure of the navigation affects the nature of the application in many ways. If the navigational structure is implemented with well-defined structures, the user can easily browse around among hypermedia documents.

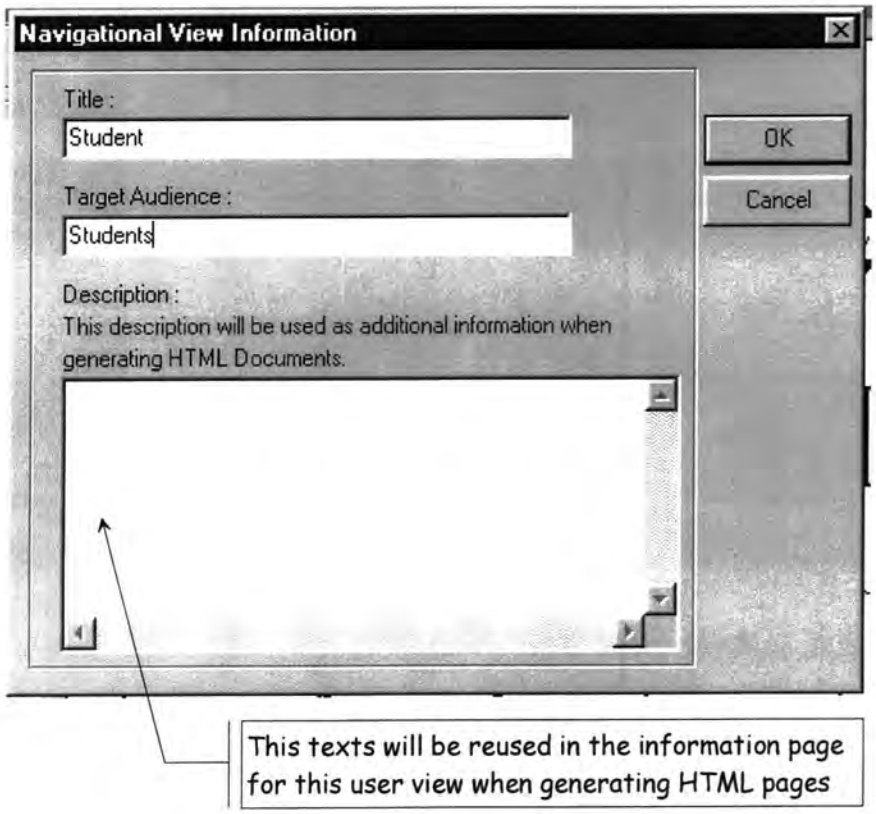


Figure 33 Information Dialog box for a New Navigational View

In SWDT, the navigational design environment provides the designer to construct the navigational patterns of an application. The designer can create one or more navigational views based on one conceptual design. Figure 33 shows the dialog box when the designer creates a new navigational view. Design primitives

As shown in Figure 34, the navigational environment includes three main design primitives. They are:

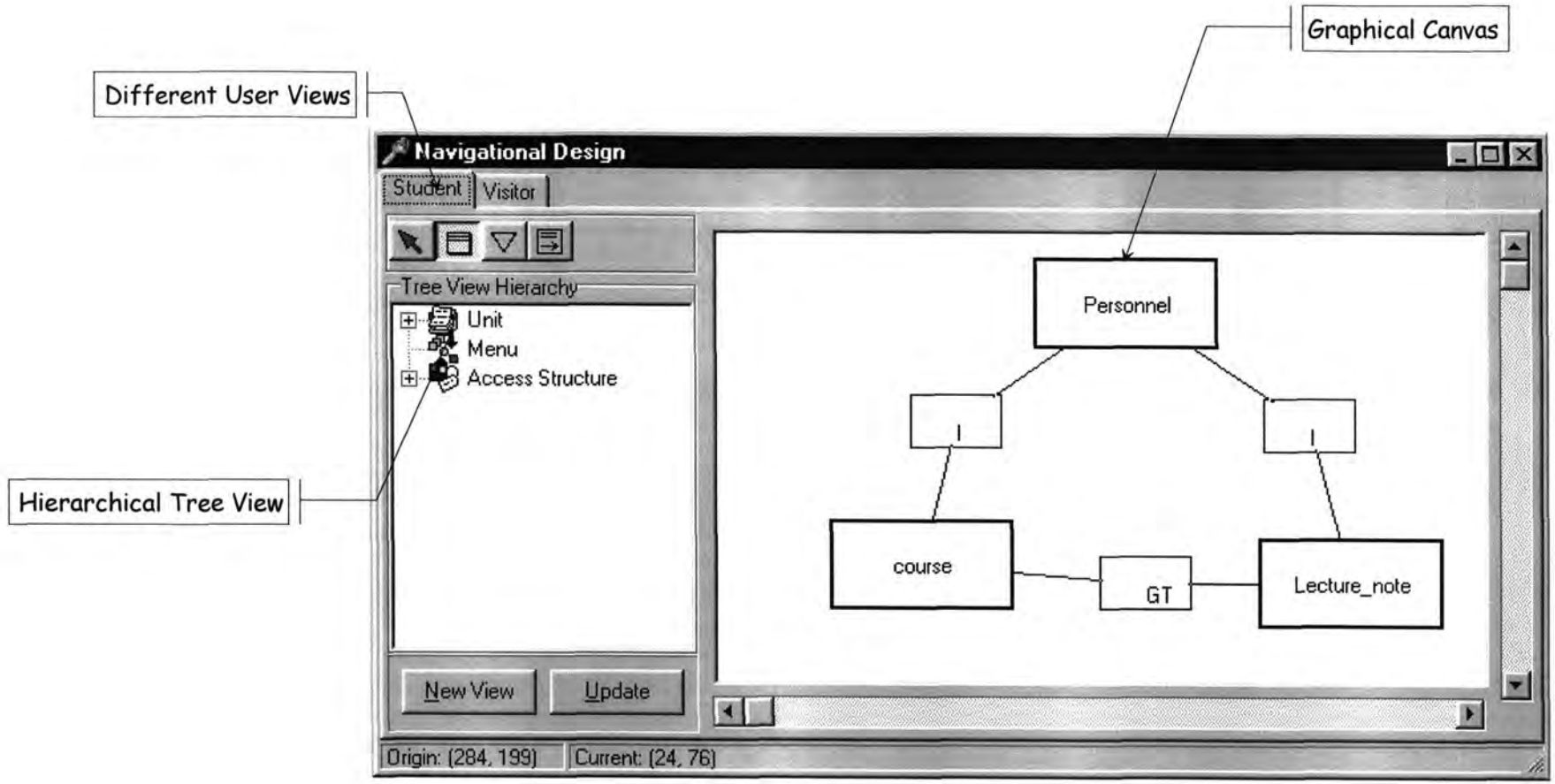


Figure 34 Navigational design environment

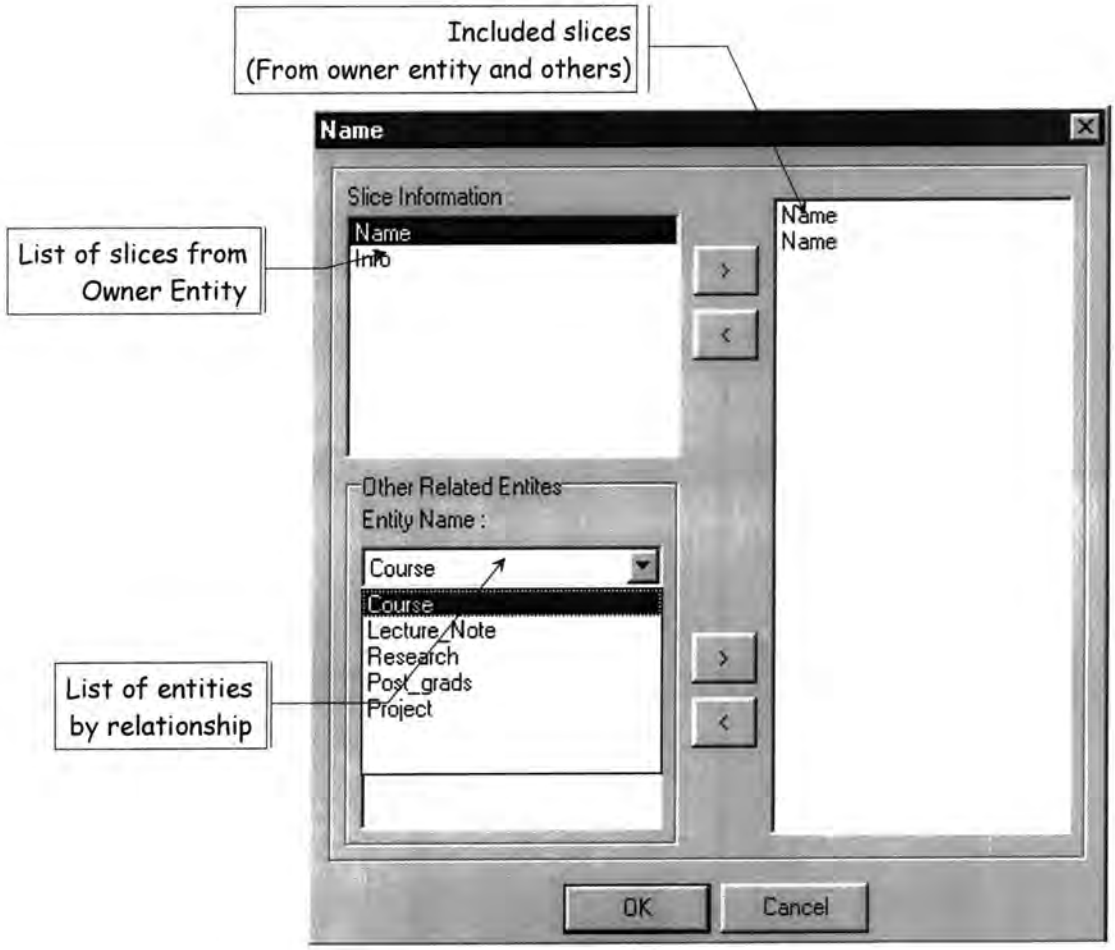


Figure 35 M-Slice Information (includes slices from owner entity and others)

Information Units – information units are similar to HTML pages and are also similar to Mslices in Extended RMM. (Isakowitz, Kamis, 1997) As shown in Figure 35, each unit is based on one owner entity, but may consist of one or more slices from other entities based one the relationship with the owner entity and the others. Therefore the designer can choose and present information that is useful for a particular view. As shown in Figure 34, information unit 'lecture notes information', for *student* viewpoint, consists of information attribute from lecture note entity together with lecturer name from lecturer entity. This lecturer name will be used as an anchor to link the lecture info unit when generating HTML documents. As shown in Figure 34, each information Unit is presented with a square box. The user can simply click the box to view the attributes of the unit or view from hierarchical tree view.

Menus – menus are the entry points of the navigation for a particular set of topics. In other words menus can be used to group the information in very top level of the design. For example, a menu can be applied to group information about the department, which includes

many items such as faculty, research, courses, etc. a menu is represented with a square box. As shown in Figure 36, the size of menu box is smaller than unit box.

Access Structures – Information units are linked with access structures. Access structures can be indices, guided tours or indexed guided tours. An index is a list of items to link directly to each listed hypermedia documents (pages). It can be also regarded as an anchor to other origin. A guided tour is a series of links, which allows the user to move back and forth on the path. For example, the students should be able to go though the lecture nodes one after another. An indexed guided tour includes both index and guided tour links. Figure 36 shows three graphical notations provided in the tool. In SWDT, the designer can create new access structures by easily connecting between two links and by selecting one the access structures. These links are based on previously defined relationships during the conceptual schema and provides different ways for easy access to information. Furthermore, since the designer is allowed to construct links to other documents based on the relationships, only relevant information will be anchored from the origin.

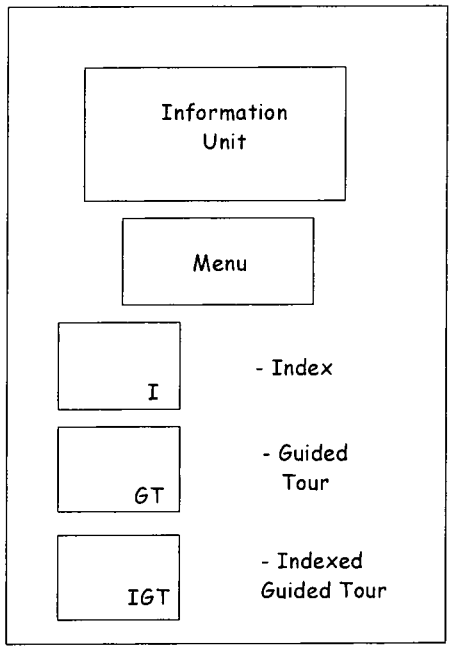


Figure 36 SWDT Navigational Design Primitives

5.6.1 Representation of information

One of the main features of the SWDT tool is its ability to offer different viewpoints, different ways of looking at the same information. For the above DCS example, Figure 34 shows the sample navigational view for *visitors* and *students*. For students' point of view, information such as lecture time and notes can be made accessible easily, while for visitors' point of view, information such as lecturers and available courses are important. The designer can construct a full navigational structure in each view for different sets of potential users. Each view represents as logical window for the combination of one or more entities, which are defined in the conceptual design stage. In SWDT tool, each tab sheet (i.e. each view) is provided with both hierarchical view and graph space (i.e. drawing canvas to construct navigational structures). Therefore the designer can work with both features.

5.7 User Interface Design selection (templates)

In (Schwabe, Rossi, 1996), Schwabe has presented the use of Abstract Data View (ADV) to describe the user interface design for hypermedia applications. These allow the designer to construct different interfaces for the same navigational design. Since it is based object oriented approach, theses views provide formal and structured interface prototypes for final implementation. They are most relevant to the presentation of objects for standalone hypermedia applications. However in SWDT tool, the main objective is to prototype HTML documents for the Web. Accordingly the support for interface design is implemented using HTML structure to represent the structure and relationships constructed in the earlier stages. To construct structured Web pages, the designer has to build efficient design and presentation for all objects within every pages and consistent navigation. In SWDT tool provides templates in order to enforce the usability of the whole Web site and consistent structure of information. The use of templates and structured layouts also enables the use to predict the potential layout of the information. The tool provides four types of *frames* in HTML and the content will be generated based on the use selection. This allows the users of the Web site to systematically browse the information. However, since the tool is focused only on the structure of information and omitted the support for building detailed graphical elements (such as buttons, backgrounds, and other animated details). Hence the product (i.e. HTML documents) of the tool would be skeleton frame structure with data elements from the database without any enhance graphics. Yet the designer can apply other HTML editing tools such as (MS front page, HOT METAL PRO, etc) to enhance the presentation of objects for different sets of audience.

As shown in Figure 37, the designer can choose four types of HTML document versions. Firstly, *no frame* presentation which contains all information on one HTML document. It includes page title, related links, description, and other navigational features in one document. Although this type of presentation is not recommended for clarity, the designer can still choose for browsers that do not support HTML frames. Secondly, *two frame presentation* which exclusively separate between document description and its related links. Therefore the left frame contains links and navigational features while body consists of title and body of the page. This separation between links and body text enhance the usability and accessibility of the information. This separation also provides easy updating the information. For instance, adding additional information to the description of the body does not affect the link presentation. Thirdly, *three-frame* based version as shown in Figure 37. The separation of title from the body of the text provides clear title area and also allows the designer to insert other logos as default template. Finally, *four-frame* based presentation, which enhance the

usability of the document and also the recommended layout. It contains the following four frame sets. A 'top frame' for main title area and other logos for the application. The title of the document is derived from the first slice of the based information unit related to that document. For example, when the designer sets the 'name' slice as the first slice within the 'lecturer' information unit, the name of the lecturer will be the title of page when presenting the page. Therefore the designer have to carefully set the first slice when creating information slices during the navigational stage. A 'body frame' contains the content of the information. This information is also based on slices in the information unit related to that page. The designer may add links to information slices within body. A 'left frame' consists of links to related documents. As described earlier, these links are derived from the relationships with the owner entity and others. During generation of HTML codes, these links are built upon the primary and foreign key attributes within entities. For instance, between 'lecturer' and 'course' relationship, each lecturer information may include the links to the courses that he/she teaches. Accordingly, each course information may include link to lecturer. Finally, a 'bottom frame' or global navigation frame, which include general indices, glossary, and guided tour for documents within the associated topic. The designer can also add additional links such as mail to and information about the site. Although four-frame based HTML presentation is recommended the designer may choose his/her preference.

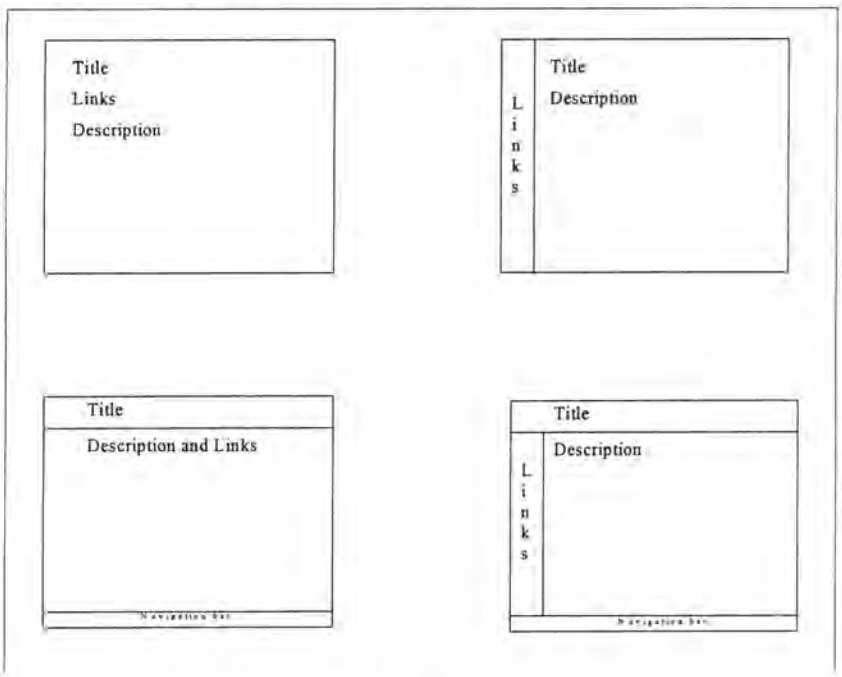


Figure 37 User Interface Template Selection

5.8 Data Entry (Data Population)

Data entry, also known as content creation or authoring, can be made after the conceptual design has been set. However the alterations such as changing and deleting of entities during the developing may affect the structure of database tables. This stage is done by simply filling the tables. These tables are derived from the entities and attributes of the conceptual design. Once the conceptual design, navigational patterns and interface design have been identified, the designer or subject experts can enter the data. Since the links are derived from the relationships among entities and attributes, the designers or content creators have to carefully enter the necessary primary and foreign keys. In SWDT tool, the designer has to simply click 'data entry' button and the tool automatically generates data entry forms to input, as shown Figure 38.

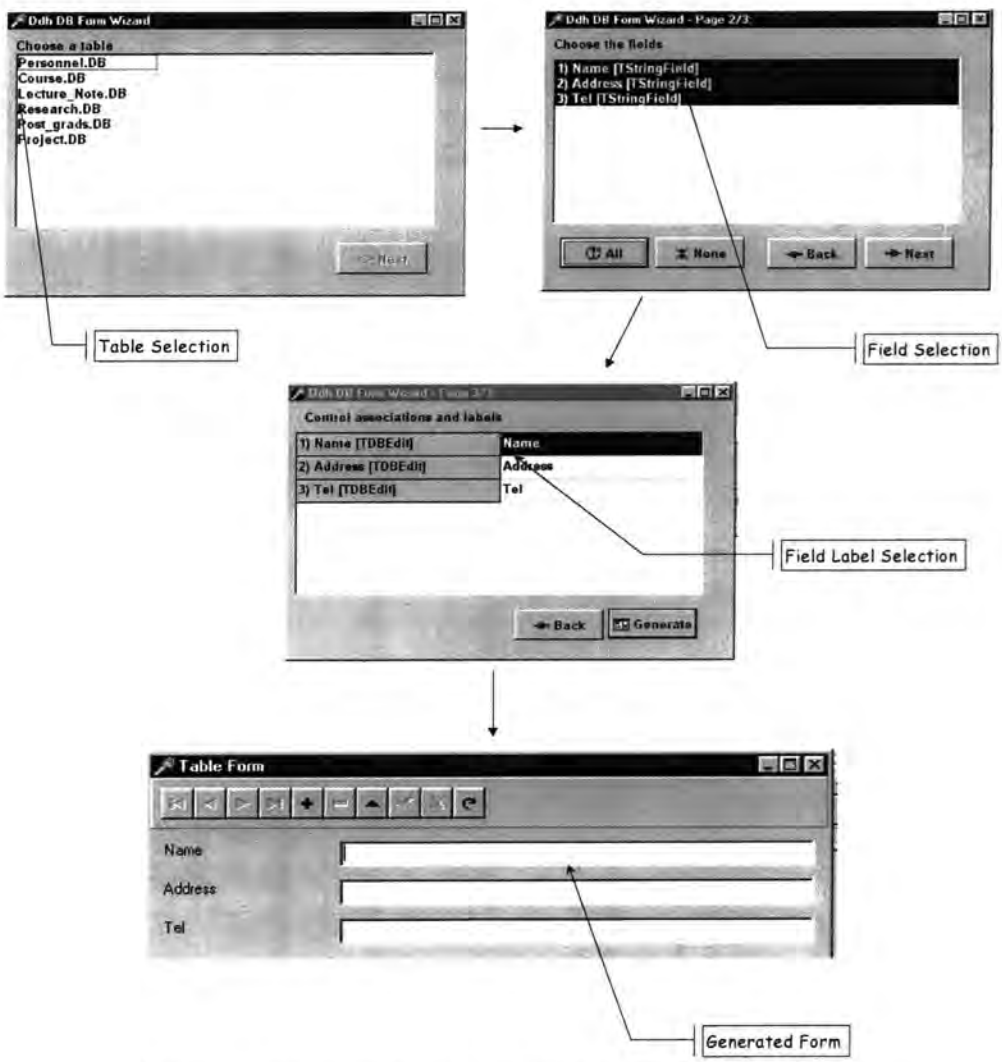


Figure 38 Data entry form (automatically generated by the tool)

5.9 HTML Generation

After identifying conceptual design, navigational design, user interface design and entering data into the database, the designer can simply click 'Generate' button to create HTML pages. Once the generation starts, the tool automatically creates different directories for each different navigational view. Since HTML documents produced by the tool uses standard templates selected by the designer during interface design, they allow the user to browse the documents with clear and consistent look and feel throughout all documents. However additional graphics such as buttons, logos, backgrounds, have to be added or updated manually by the designer.

5.10 Internal functions of the tool and the use of Database

The SWDT was developed using Borland Delphi 4 under MS Windows 95 environment. It uses Paradox database tables to store and manipulate the data including all necessary information about different design environments. Accordingly information from one design stage is reused in other stages. Furthermore every objects that the designer creates are persistently stored. Therefore the data can be reused in other applications. However, since the tool is only a prototype, there are still improvements to be made.

5.11 The tool implementation overview and summary of main points

As an overview, we have implemented a method based hypermedia design tool for constructing HTML developments. We have also identified the detailed features provided by the tool in each stage and discussed the important issues during the building process of the applications. Furthermore, we have also described the internal functions of the tool including database structures implemented within the tool.

In the next section we will describe the results and evaluate the tool, which has been implemented in this work. As describe earlier, the aim of the next section includes the description of developing the DCS Web site to test the CASE tool and analysing the outcomes. It examines the advantages of implementing and using the CASE tool and compares the results with other existing CASE tools. Then it is followed by a discussion of the ways in which this work could contribute towards developing and managing Web-based hypermedia documents.

6. The Evaluation of the CASE tool and Trial Usage

6.1 Chapter Abstract

Once the SWDT CASE tool is implemented, the main objectives of this chapter is to analyse whether the tool meets the requirements identified in chapter four. It also includes the description of the way the SWDT supports towards the development and maintenance of Web documents. We have also evaluated Web documents produced by the tool and showed the quality and structure of theses documents.

6.2 An overview of the tool and its evaluation

In chapter four, we have identified a list of requirements that have to be met. The following section shows different guidelines and features that have to be included in the SWDT tool and hypertext design environment.

As an overview the requirements are related to design environment and development stages provided by the tool. One of the main requirements of the tool in general is to provide a series of design environments to construct Web documents based on the basic model shown in chapter two. The following section describes the detailed requirements proposed and a number of features that are implemented during the development of the tool.

Currently, there are very few CASE tools that support the development of Web documents using structure model with explicitly defined development stages. Therefore as we identified in chapter 4, the tool must have different design and development stages that support a series of different context and guidelines proposed in the model. In detail, there are five context, namely : conceptual design, navigational design, user interface design, data entry, and generation of Web documents. For each design context there are guidelines and functionality that have to be met. Although there were time constraints, all the necessary design environments for different stages are implemented.

The need for graphical editor is essential for every CASE tool to construct prototypes. Accordingly, we have also identified the need for graphical editor, which allows the designer

to build an abstract prototype on the drawing canvas. It also must have ability to view the elements of different ways including displaying the attributes, adding, modifying, and deleting the entities and relationships on the drawing canvas. This requirement was also met as the tool consists of drawing canvas to support conceptual and navigational design. Furthermore, it allows the designer to instantly linking and creation of entities and attributes within an entity, which enhance the designer's productivity. In chapter four, we have addressed the use of hierarchical view to visualise the structure of entities, attributes and their relationships. According to the design guideline, we have implemented a hierarchical view for each design environment.

As Nanard & Nanard have described, fast feedback loops between the development stages are one of the key features of the design process. In other words, it facilities the designers to perform the design process with recursion or incremental approaches. It also allows the designer to evaluate his/her work in each design stage with improve the quality of the design. The tool also supports the facility so that the designer can easily move back and forth between different stages. Therefore the tool can be regard as it meets the requirement for fast feedback loops. Navard & Nanard have also outlined the requirement for prototyping the abstract model level of the application. The tool meets this requirement because it enables the designer to construct an abstract entity relationship model. Therefore the designer can visualise the abstract view of the structure of the documents.

One of the another requirements that we have addressed in chapter four was the use of relationship management and databases to store and manipulate the structure of the application. Based on the relationship management, the manipulation of entities, attributes and relationships are all supported by the tool to construct the conceptual model of the application. It also capable of generating the necessary database tables once the designer create a new project and these tables are used to store the structure of application in every development stages within the project. Although the tool provides the necessary facilities to design and development prototypes there are still other improvements can be made such as importing database and exporting design structures. For data entry step, which is one of the development stages, the designer or the content creator can populate the data by using internal forms that are automatically generated by the tool or by using external applications such as Borland Paradox which is capable of reading Paradox database tables. Accordingly the data populated by designer can be reused in other applications.

One of the guidelines that are addressed in the design requirements is the ability to generate HTML documents. By generating HTML documents, the design can instantly view the

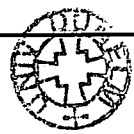
prototype by using an HTML viewer or Web browser. Thus facilitates designers to test various features of the design including its information structure, flow of the information and navigational structures. In SWDT tool, we have implemented the generator which allows the developer to instantly create a collection of HTML documents based on the structure of information and the database after data entry (i.e. data population). This functionality supports the designer to evaluate his/her work, thus improving the quality of the design. However due to the limitation of time and it is only a prototype, advance functions such as input forms, Java and Java scripts, and other object behaviors are not supported by the tool. Instead, the design and implementation of tool is focused on the development of structured design for information domain and navigational patterns.

As a summary, although there are time constraints, all the requirements were met. However for the tool implementation, some of the advanced features were impossible to development during the limited time, but all the necessary functions are implemented and can be regard as a satisfactory result. Therefore many improvements can be made to facilitate more functions to the tool to produce structured Web documents with advanced features.

6.3 Comparison between RMCase and SWDT

Since the ideas and functions in our tool are derived from RMM, there are similar functions and features with RMCase. The following tables show the comparison between two CASE tools. The first table shows different design environments (design context) between to CASE tools. The second table shows different features provided by the tool to assist the work of the designer.

SWDT	RMCase
Conceptual Design The designer can construct Conceptual design by using Entities, attributes and Relationships. It uses both graphical drawing canvas and hierarchical tree view to structure the information.	***Entity-Relationship Design and Slice Design*** The designer identifies entities, attributes, slices and relationships on the graphical canvas. Since both tools uses Entity Relationship (ER) Model, both design environments have similar functionality.
Navigational Design The designer can build one or more navigational views for different sets of users based on one conceptual design from previous stage. Each navigational view allows the designer to build access structures and information units depending on the requirements. Each view also consists of hierarchical tree view and drawing canvas.	***Navigational Design*** The tool provides only one navigational view with predefined information units. Therefore for each node, the designer cannot add additional information slices from other relation nodes and can only create access paths for navigation.
User Interface Template Selection The tool provides a selection of templates, based on HTML frames, that the designer can choose. However due to the lack of time, the support for the design of individual objects or units has not implemented.	***User Interface Design (Not Implemented Yet)*** There is no User Interface Design support in the RMCase version at the time of this research.



Data Entry The designer or content creators can add the data attributes to create content. Since the tool creates external database tables (Paradox tables), the designer can use internally generated formal for external application to fill the contents.	***Data Population*** The tool provides an internal form to fill the content.
HTML Generation The designer can generate HTML documents for each navigational view to test and evaluate the design structure.	***Simulation / Prototyping*** The designer can simulate the prototype after populating data. The tool uses internal simulator to simulate.

6.4 *Trial Usage*

The trial usage of the tool was performed through during and after implementation of the tool. The trial was carried out using the information structure of the Department of Computer Science (DCS). The guidelines are identified and implemented according to the guidelines. In chapter 5 we have shown the development of DCS Web documents as an example together with the implementation issues. In general, the tool is user friendly and easy to install. It is based on Windows 95 and all necessary setup files and database engines are included as a install package. The Web documents produced by the tool are consistent and easy to navigate without any disorientation and inconsistencies. However the Web documents produced by the tool consist of only one default style layout (i.e. background color, font size, color, etc) for each frame selection, since it is only a prototype. Therefore for further alterations (such as additional graphics, images, and other improvements), the designers have to manually edit the pages by editing the HTML code or by using other HTML editing tools.

During this trial, we have encountered a few important factors that the designer has to carefully conduct during the design process. Firstly, generation of HTML documents purely dependent on the structure of relationship between entities. Therefore the designer has to carefully add the primary and foreign keys when adding attributes for relationships. Accordingly any mistakes made by the designer during adding the contents or adding the keys can lead to inconsistent or broken links. Secondly, the designer has to add necessary information such as project 7information, authors e-mail address, and a brief description for each navigational view. Since this information is reused as Web pages when generating HTML documents, it is essential to fill these sets of information by the designer both for link integrity and from the maintenance point of view. Finally the tool is only a prototype and developed within very limited time, there are still many alterations and improvements (such as input validation, help files, additional user interface styles, and importing and exporting facilities, editing and drawing functions) to be made. However these trade-offs and limitations are expected. Because to design and develop fully functional CASE tool with integrated entity relationship design environment, Web authoring environment, WYSIWYG HTML editor requires a large technical support, time, and standardisation of design approaches. Although we have lack of these above supports, we have successfully implement a CASE tool as a prototype that can generate HTML documents.

6.4.1 Test Criteria

After the trial implementation of DCS HTML documents, a set of test criteria has been set to evaluate the consistency of these documents that were produced by the tool. They are as follows:

Actual document and link structure

After testing the content of each documents (pages), all the documents appeared to be consistent and structured since they use standardised templates (i.e. font size, color, paragraph format, etc.). Links used for external and internal relationships are also consistent and also avoid user disorientation. The tool does, however, pose the limitation that the tool only generates static HTML links with pre-formatted structure and links without the use of any dynamic linking (such as the generation of CGI scripts) particularly for searching and other queries. These advance features are limited in the tool mainly because, this kind of features require not only structured design and implementation, but also need time and technical supports.

General accessibility

The general accessibility of the documents is entirely depending to the designer's navigational design structure. For DCS example, since the nature of the domain is simple and the design of the application is structured, all the Web documents have high integrity and general accessibility. More fundamentally, it is the designer's task to provide different types of access structures (i.e. indices, guided tours, etc) wherever appropriate, hence the tool should generate structured HTML documents with high accessibility.

Link integrity

Form the designer's point of view, the tool produces structured links based on relationship information from hyperbase and data from database. The tool is implemented in the way that all HTML files are named and linked according to key attributes of the entities. Therefore as the designer makes links between entities, the tool should automatically generate links with high integrity and reliability. For the trial implementation with DCS example, all the links have high integrity and reliable.

User Interface Design

Since the tool only supports default template, all pages produced by the tool contains the same style and layout. More implementation time would be required in order to add additional templates and in order to integrate graphical editors, which allow the designer to set guidelines and layouts for each object.

Despite these limitations, the tool clearly provides support for the design and development of structured Web applications and documents using hypermedia design methods. Although the use of HTML documents is only to deliver information over the Web, it is essential to support the development of Web-based application for present and for the future. The developers have implemented many Web-based applications and HTML documents without applying any formal hypermedia design methods. Hence out work enforces the Web designers and developers with an aid when design Web documents. In recent years, researchers such as (Andrews, Maurer, 1993), (Díaz and Isakowitz, 1995) have presented with their formal design approach with supported CASE tools. However it would be useful if more commercialised tools, which integrate formal design methods together with other current technologies such as Java, Java Scripts, CGI Scripts, and HTML editors, but is beyond the scope of the work.

6.5 *Further development of the SWDT tool*

Further development is required to enhance the functionality of the tool and other editing and visualisation features. In more detail, the following features can be integrated to improve the tool.

- For analysis and design of the domain, the tool can be implemented in such the way that it can support both Entity-Relationship model and Object-Oriented model depending on designer's preference. For example, based on the problem domain, if the application requires features such as object behaviour and functions, rather than just for presenting the information, the designer can choose Oriented-Oriented approach for constructing the conceptual design of the application domain.

- For designing the user interface, a design environment can also be integrated for structuring and layout of each object, its appearance, behaviour (such as input validation, user response, etc.).

- The tool can also be added with features such as exporting and importing graphs. Furthermore, the designer should be able to import database tables or persistent classes from other existing applications.

- For generating and simulating of the prototype, the need for dynamic linking requires the generation of CGI scripts. Therefore the tool can be provided with the features for creating CGI scripts on the fly, if necessary.

- For testing the result application, the tool can be facilitated with maintenance features such as testing the link integrity, measuring the quality by using metrics.

6.6 The Assessment of the Tool

Although there are enhancements to be made, the tool meets the initial design objectives and requirements that are presented in chapter four. The development time, however, was very limited. As an overview, the implementation of this tool is the support for development of producing structured Web based applications using formal design methods. The tool also fulfils the guidelines and problems that are identified and discussed through the literature survey in chapter two. For this research the tool is only a prototype and never the less it is fully functional and achieves its goal.

6.7 Summary of Main Points

This chapter has examined the tool and evaluated the implementation of the tool. It has also outlined the difference between the SWDT tool with the existing RMCase. Furthermore, it has described the important issues from the resultant prototypes that are produced by the tool during the trial usage. This trial work showed that the tool provides support for the design and development Web documents with better quality and encourage the designers to apply structured methods for any Web development. It also identified further improvements to be done for the SWDT CASE tool and assessed our work.

7. Conclusion

7.1 Objectives of the Chapter

The main purpose on this chapter is to conclude the work by outlining the important issues and ideas that has been presented in the previous chapters. A brief description of the work that has been done will be given as a part of the conclusion. The criteria for success will also be described. The chapter will be concluded by identifying future development of the tool and future work on this research area.

7.2 An overview of the Work

As an overview this research was begun by examining the area of hypermedia and the World Wide Web. This area includes the way hypermedia is related to the Web and the development of Web applications using hypermedia design methods. The survey was started in chapter two by discussing hypermedia and its features. It also included the relations between the design of Web documents and hypermedia design methods. It described how these design methods could assist the designers for designing the Web applications in structured manner and encouraged the designers to apply structured methods. Later in the chapter two, different existing hypermedia design methods were identified and described three methods in detail, namely, RMM, HDM and OOHDM. The survey also went though each design methods and outlined the basic common design stages involved in each design methods. This particular survey also showed the implication of applying these design methods to Web developments.

In chapter three, a case study was used to reverse engineer an existing Web site, SEG World, to a newly structured Web site. The study was made to illustrate the ways to redesign existing Web sites to a new Web site by using hypermedia design methods. It also included the evaluation of each design method and discussed the implementation of applying these design methods to design structured Web sites.

After analysing existing design methods, the work followed on to implement a new CASE tool that allows the designer to construct structured Web sites using the basic design steps that were identified in chapter two. Chapter four examined the existing CASE tools that have been proposed by other researchers to support their methods and outlined their benefits and limitations. Hence, after analysing the current design methods from various perspectives, the requirements for a new CASE tool were defined in Chatper five.

After the tool was design and implemented, the tool was to be evaluated and prepared for trial usage. At this point, there were main features that have been implemented in the tool that need to be described. Therefore, in chapter five, a scenario was given to demonstrate the design environments and features provided by the tool. This scenario was designed and developed using the tool and ready to examine Web documents generated by the tool. In the following chapter, chapter six, the resultant documents from the trial usage of the tool were evaluated. Therefore the evaluation was made on two parts. The first part was the evaluation of resultant Web documents produced by the tool by using test criteria and the second part was the assessment of the tool itself. The comparison between the newly development CASE tool and existing RMCase tool was compared went through some strengths and weaknesses. Chapter six is concluded by outlining further improvements for the tool.

This work is concluded by assessing the success of the research and outlining the areas of future research.

7.3 Criteria for Success

The overall research can be regarded as a successful work. In the beginning of the thesis a list of criteria for success was stated. The first criterion is to understand the background of hypertext, hypermedia design methods and the way hypermedia is related to the development of Web-based applications. The area was gained in chapter 2 as a literature survey was made on hypermedia features, design methods and described the relation between different design methods and showed how they are derived from and described three methods in detail. Chapter two also showed the relation between deign methods and the development of Web based hypermedia applications. To obtain more understanding of hypermedia design methods, a case study has been used to reverse engineered an existing Web site to a new Web site in chapter 3.

The second criterion is to identify an approach to develop Web documents using hypermedia design methods, which consist on full development life cycle. This was done after analysing existing design methods and the basic development stages are identified in chapter two. Like Waterfall model in Software Engineering, these basic development stages are essential stages during the construction of Web-based hypermedia applications and are common in most existing design methods. The another criterion to be achieved is to design and implement a CASE tool to support these basic development stages that are identified. This was done, in chapter four, as the requirements and guidelines for the new CASE tool were defined, including visual design environments, and database structures. The tool was implemented based on the guidelines that are identified in chapter four. In chapter five, different features supported by the tool are presented including visual design environments and generating of HTML documents. Therefore the design and implementation of the tool can be regard as a successful work. The last criterion is to evaluate the tool and the resultant applications produced by the tool. This was done in chapter six as the evaluation on the trial usage was made. The evaluation was main by comparing with the another CASE tool and by assessing the quality of HTML documents produced by the tool using different test criteria.

7.4 Future research in this Area

There are many areas that need to be worked on further research. As the World-Wide-Web is growing day by day, many organisations and individuals are implementing new and complex Web sites to deliver information and Web applications for other purposes. Therefore further research can be made on new structured design methods, CASE tools, design and development guidelines and integration of current technology. These researches and developments, in fact, encourage the Web designer to apply structured methods rather than ad-hoc approaches. This will lead to better quality Web documents and lower maintenance cost for the long term. However, Web applications are doubling every day, the need for appropriate method and tools is urgent. When the Web Sites were beginning to develop, the aim only to deliver information, but as the technology changes rapidly the Web documents are used as other objectives such as remote transactions, banking, etc. However the lack of structured tools is one of the barrier for assisting end users and other organisational people to construct structured Web sites.

7.5 Summary

As a summary, the work done for this thesis is concluded in this chapter. It described the overview of the work that has been done for this research and showed the work completed the four criteria for success. Then followed the way to extend the work for the future. Since the World Wide Web is growing rapidly, the further work on the design and development of the Web is essential.

Appendix A

Tool locations

SWDT CASE tool – http://www.dur.ac.uk/~dcs3pk/SWDT/

RMCase CASE tool – http://www.stern.nyu.edu/~tisakowi/rmm/

HMCard CASE tool – http://www2.iicm.edu/hmcard/

Demo Applications

Department of Computer Science Example - http://www.dur.ac.uk/~dcs3pk/SWDT/demoapps/DCS/

Internet References

Tomas Isakowitz's Relationship Management Methodology (RMM)

http://www.stern.nyu.edu/~tisakowi/rmm/

Daniel Schwabe's (schwabe@inf.puc-rio.br) The Object-Oriented Hypermedia Design Model (OOHDM)

http://www.telemidia.puc-rio.br/oohdm/oohdm.html

Hyperwave Information Server

http://www.hyperwave.com/

Seventh ACM Conference on Hypertext Online Proceedings

http://www.cs.unc.edu/~barman/HT96/

Hyper-Proceedings of the WWW-6 CONFERENCE

http://proceedings.www6conf.org/

Jocelyne Nanard, Marc Nanard's Hypertext Design Environments and the Hypertext Design Process

http://www.cs.uct.ac.za/Courses/CS200W/Resources/HCI/nanard.html

Elements of Hypermedia Design

http://www.birkhauser.com/hypermedia/

W3DT (WWW Design Technique)

http://wicky.wu-wien.ac.at/w3dt/

International World Wide Web Conferences

http://www.w3.org/Conferences/Overview-WWW.html

References:

1. K. M. Anderson, *Extending User-Interface Toolkits with Hypermedia Functionality*, The 30th Annual Hawaii International Conference on System Sciences (IEEE, 1997), vol. 6.

2. K. Andrews, H. Maurer, N. Scherbakov, "Semantic Modeling in Hypermedia: An Object-Oriented Approach" (Graz University of Technology, 1993).

3. K. Andrews, F. Kappe, H. Maurer, "Serving Information to the Web with Hyper-G" (University of Graz, 1995).

4. V. Balasubramanian, A Systematic Approach to Designing a WWW application, *Communications of the ACM* **38** (1995).

5. V. Balasubramanian, A Systematic Approach to User Interface Design for Hypertext Systems, The 27th Annual Hawaii International Conference on System Sciences (IEEE, 1995).

6. V. Balasubramanian, A. Bashian, D. Porcher, A Large-Scale Hypermedia Application using Document Management and Web Technologies, Hypertext 97, Southampton, UK (ACM, 1997).

7. M. Bichler, S. Nusser, Modular Design of Complex Web-Applications with SHDT, ECIS (1996).

8. M. Bieber, F. Vitali, Some Hypermedia Ideas for the WWW, The 30th Annual Hawaii International Conference on System Sciences (IEEE, 1997).

9. M. Bieber, F. Vitali, Toward Support for Hypermedia on the World Wide Web, *Computer IEEE* **30**, 62-70 (1997).

10. H. Brown, Ed., *Hypermedia/Hypertext And Object-Oriented Database* (CHAPMAN HALL, London, 1991).

11. J. F. Buford, Evalating HyTime: An Examination and Implementation Experience,

Hypertext, 96, Washington D. C. (ACM, 1996).

12. V. Bush, As We May Think, *The Allantic Monthly* (1945).

13. J. E. Conlin, Hypertext: A Survey and Introduction, *IEEE Computer* **20**, 17-41 (1987).

14. D. D. Cowan, C. J. P. Lucena, Abstract Data Views, An Interface Specification Concept to Enhance Design for Reuse,*IEEE Transactions on Software Engineering* **21** (1995).

15. A. Díaz, T. Isakowitz, RMCASE: Computer-Aided Support for Hypermedia Design and Development, *International Workshop on Hypermedia Design* (1995).

16. F. Garzotto, P. Paolini, D. Schwabe, HDM - A Model-Based Approach to Hypertext Application Design, *ACM Transactions on Information Systems* **11**, 1-26 (1993).

17. F. Garzotto, L. Mainetti, P. Paolini, Hypermedia Design, Analysis, and Evaluation Issues, *Communications of the ACM* **38**, 74-85 (1995).

18. F. Garzotto, L. Mainetti, P. Paolini, Information Reuse in Hypermedia Applications, The Seventh ACM Conference on Hypertext, Washington D. C. (ACM, 1996).

19. K. Gronbæk, N. O. Bouvin, L. Sloth, Designing Dexter-based hypermedia services for the World Wide Web, Hypertext, 97, Southampton, UK (ACM, 1997).

20. L. Hardman, D. C. A. Bulterman, Using the Amsterdam Hypermedia Model for Abstracting Presentation Behavior, ACM Workshop on Effective Abstractions in Multimedia, San Francisco, California (ACM, 1995).

21. K. Hendrikx, H. Olivié, E. Duval, Object-Oriented Hypertext Design: Authoring for Reuse, The 30th Annual Hawaii International Conference on System Sciences (IEEE, 1997).

22. W. Horton, L. Taylor, A. Ignacio, *The Web Page Design Cookbook* (John Wiley & Sons, Inc., USA, ed. 1st, 1996).

23. T. Isakowitz, E. A. Strohr, P. Balasubramanian, RMM: A Methodology for Structured Hypermedia Design, *Communications of the ACM* **38**, 34-43 (1995).

24. T. Isakowitz, A. Kamis, M. Koufaris, Extending the capabilities of RMM: Russian Dools and Hypertext, The 30th Annual Hawaii International Conference on System Sciences (IEEE, 1997).

25. P. Kahn, Visual Cues for Local and Global Coherence in the WWW, *Communications of the ACM* **38**, 67-69 (1995).

26. P. Kyaw, C. Boldyreff, "A Survey of Hypermedia Design Methods and the World Wide Web" (University of Durham, 1998).

27. D. B. Lange, An Object-Oriented Design Method for Hypermedia Information Systems, The 27th Annual Hawaii International Conference on System Sciences (IEEE, 1994).

28. J. A. Lennon, *Hypermedia Systems and Applications: World Wide Web and Beyond* (Springer-Verlag Berlin Heidelberg, German, 1997).

29. H. Maurer, F. Kappe, N. Scherbakov, P. Srinivasan, Structured Browsing of Hypermedia Databases, VCHCI'93 (1993).

30. F. R. McFadden, J. A. Hoffer, *Modern Database Management* (The Benjamin/Cummings Publishing Company, INC., Canada, ed. Fourth, 1994).

31. M. E. S. Morris, R. J. Hinrichs, *Web Page Design: A Different Multimedia* (SunSoft Press, USA, 1996).

33. J. Nanard, M. Nanard, Hypertext Design Environments and The Hypertext Design Process, *Communications of the ACM* **38**, 49-56 (1995).

34. S. R. Newcomb, H. A. Kipp, V. T. Newcomb, HyTime: The hypermedia/time-based document structuring language, *Communications of the ACM* **34**, 67-83 (1991).

35. J. Nielsen, *Multimedia and Hypermedia: The Internet and Beyond* (AP Professional, London, 1994).

36. G. Rossi, D. Schwabe, C. J. P. d. Lucena, D. D. Cowan, An Object-Oriented Model for Designing the Human-Computer Interface of Hypermedia Applications, International Workshop of Hypermedia Design (1995).

37. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, *Object Oriented Modeling and Design* (Prentice Hall Inc., 1991).

38. M. Salampasis, J. Tait, "Hypertree: An Alternative approach to Web Authoring" (University of Sunderland, 1995).

39. M. Salampasis, Hypermedia Report, University of Sunderland (1995).

40. D. Sano, *Designing Large-Scale Web Sites: A Visual Design Methodology* (Wiley Computer Publishing, Canada, 1996).

41. W. Schuler, J. Hannemann, N. Streitz, *Designing User Interfaces for Hypermedia*, Research Reports ESPRIT (Springer-Verlag, 1995), vol. 1.

42. D. Schwabe, G. Rossi, Building Hypermedia Applications as Navigational Views of Information Models, 28th Annual Hawaii International Conference on System Sciences, Mavi (IEEE, 1995).

43. D. Schwabe, G. Rossi, S. D. J. Barbosa, Abstraction, Composition and Lay-out Definition Mechanisms on OOHDM, ACM Workshop on Effective Abstractions in Multimedia, San Francisco, California (ACM, 1995).

44. D. Schwabe, The Object-Oriented Hypermedia Design Model, *Communications of the ACM* **38**, 45-46 (1995).

45. D. Schwabe, G. Rossi, S. D. J. Barbosa, Systematic Hypermedia Application Design with OOHDM, Seventh ACM Conference on Hypertext, Washington DC (ACM, 1996).

46. O. Signore, Exploiting Navigation Capabilities in Hypertext/Hypermedia, The 29th Annual Hawaii International Conference on System Sciences (IEEE, 1996).

47. K. Takahashi, Analysis and Design of Web-based Information systems, Sixth International World Wide Web Conference (HyperNews, 1997).