

Personalisation of Web Information Search: An Agent Based Approach

A thesis submitted for the Degree of

Master of Applied Science in Information and Technological Sciences

In the

**School of Information Science and Engineering
UNIVERSITY OF CANBERRA**

Submitted by

**Ligon Gopinathan Leela
December 2005**

CERTIFICATE OF AUTHORSHIP OF THESIS

Except where indicated in footnotes, quotations and the bibliography, I certify that I am the sole author of the thesis submitted today entitled – **Personalisation Of Web Information Search: An Agent Based Approach** in terms of the Statement of Requirements for a Thesis issued by the University Higher Degrees Committee.

Signature of Candidate:

DATE:

ACKNOWLEDGMENT

The completion of this research degree would not have been possible without the support of many individuals and I would like to take this opportunity to acknowledge my gratitude to all of them.

I would like to acknowledge the enormous support and guidance offered to me by my supervisors, Dr Dharmendra Sharma and Dr Bala Balachandran. They were a constant source of encouragement, inspiration, prompting and, where needed, downright hassling. They were invaluable as both mentor and coach throughout the entire process, from proposal to completion. Words cannot fully convey my sincere appreciation.

I am indebted to my friends who helped me during the research, especially Benson Babu and Vijay K Sagar. Both of them have extended priceless help while coding the prototype and have shown extraordinary patience in resolving my technical doubts. I also acknowledge my thanks to Shilpa Yadav and Serena Chong for their support in arranging meetings with my supervisors and for other administrative help. Last, but by no means least, I thank my wife, Saritha, for her love, encouragement and practical assistance.

The Academic Skills program at the University of Canberra provided practical advice and support for which I am grateful. I would also like to thank Susan Prentice for editing the thesis prior to submission.

LIGON GL

07 December 2005

ABSTRACT

The main purpose of this research is to find an effective way to personalise information searching on the Internet using middleware search agents, namely, Personalised Search Agents (PSA). The PSA acts between users and search engines, and applies new and existing techniques to mine and exploit relevant and personalised information for users.

Much research has already been done in developing personalising filters, as a middleware technique which can act between user and search engines to deliver more personalised results. These personalising filters, apply one or more of the popular techniques for search result personalisation, such as the category concept, learning from user actions and using meta-search engines. By developing the PSA, these techniques have been investigated and incorporated to create an effective middleware agent for web search personalisation.

In this thesis, a conceptual model for the Personalised Search Agent is developed, implemented by developing a prototype and benchmarked the prototype against existing web search practices. System development methodology which has flexible and iterative procedures that switch between conceptual design and prototype development was adopted as the research methodology.

In the conceptual model of the PSA, a multi-layer client server architecture is used by applying generalisation-specialisation features. The client and the server are structurally the same, but differ in the level of generalisation and interface. The client handles personalising information regarding one user whereas the server effectively combines the personalising information of all the clients (i.e. its users) to generate a global profile. Both client and server apply the category concept where user selected URLs are mapped against categories. The PSA learns the *user relevant URLs* both by requesting explicit feedback and by implicitly capturing user actions (for instance the active time spent by the user on a URL). The PSA also employs a keyword-generating algorithm, and tries different combinations of words in a user search string by effectively combining them with the relevant category values.

The core functionalities of the conceptual model for the PSA, were implemented in a prototype, used to test the ideas in the real world. The result was benchmarked with the results from existing search engines to determine the efficiency of the PSA over conventional

searching. A comparison of the test results revealed that the PSA is more effective and efficient in finding relevant and personalised results for individual users and possesses a *unique user sense* rather than the *general user sense* of traditional search engines.

The PSA, is a novel architecture and contributes to the domain of knowledge *web information searching*, by delivering new ideas such as active time based user relevancy calculations, automatic generation of sensible search keyword combinations and the implementation of a multi-layer agent architecture. Moreover, the PSA has high potential for future extensions as well. Because it captures highly personalised data, data mining techniques which employ case-based reasoning make the PSA a more responsive, more accurate and more effective tool for personalised information searching.

TABLE OF CONTENTS

Chapter 1 Introduction

1.1 Background and motivation.....	2
1.2 Problem definition and research questions.....	3
1.3 Research aims and objectives.....	4
1.4 Organisation of the thesis.....	5

Chapter 2 Literature Review

2.1 Research Methodologies.....	6
2.1.1 Experimental research.....	6
2.1.2 Action research.....	7
2.1.3 System development research.....	7
2.2 Information searches on the Internet	9
2.2.1 Search engines and how they work.....	10
2.2.2 Popular search engines.....	14
2.2.3 Issues in information searching using search engines.....	16
2.3 What is personalisation in information searching.....	17
2.3.1 Strategies in information retrieval.....	19
2.3.2 Current work in personalised information searching.....	21
2.4 Agents and information searching.....	24
2.4.1 Software agents.....	24
2.4.2 Agent classifications.....	24
2.4.3 Software agents in information searching.....	25
2.5 General architecture for information searching systems.....	27
2.5.1 Client server architecture.....	27
2.5.2 Multi-layer architecture.....	27
2.6 Summary.....	29

Chapter 3 An agent based model for the personalising of web searching

3.1 PSA architecture.....	32
3.2 A multi layered approach to personalisation.	35
3.3 PSA functionalities.....	39
3.4 Rakings and algorithms.....	42
3.5 A sample scenario walkthrough	44
3.6 Summary.....	48

Chapter 4 Personalised Search Agent (PSA): Design and Development

4.1 Design and development standards for the PSA	49
4.1.1 Object oriented software design.....	49
4.1.2 Unified Modelling Language (UML).....	49
4.1.3 Implementation tools	50
4.2 Requirement analysis using UML.....	50
4.2.1 Use cases and system sequence diagrams for the PSA.....	51
4.3 The prototype design for the PSA.....	56
4.3.1 The class diagram	56
4.3.2 ER diagrams.....	57
4.3.3 Interface design.....	58

4.4 Prototype implantation of the PSA.....	62
4.5 Assumptions made.....	63
4.6 Summary.....	64
Chapter 5 Testing, Benchmarking and Findings	
5.1 Software testing standards for the PSA.....	65
5.2 Defining the test criteria: pre and post conditions	65
5.3 Test results of the PSA.....	66
5.4 Test results for the search engines.....	68
5.5 Bench marking.....	74
5.6 Summary	76
Chapter 6 Conclusions and Future Work	
6.1 Summary of research findings.....	78
6.2 Conclusions.....	78
6.3 Future work and extensions for the PSA.....	80
BIBLIOGRAPHY.....	82
APPENDIX A: Sample interaction screen shots of PSA.....	86
APPENDIX B: Source code of the PSA prototype.....	89
APPENDIX C: Paper published in the KES International conference.....	108

LIST OF FIGURES

- Figure 2.1 : System Development Research Method
- Figure 2.2 : Technology behind the web Search
- Figure 2.3 : User profile architecture by Giuseppe Amato and Umberto Straccia
- Figure 2.4 : User Profile Modelling in Content Personalisation
- Figure 2.5 : FOCI System Architecture
- Figure 2.6 : A graphical representation for agent classification.
- Figure 2.7 : Three tier distributed client/server architecture depiction
- Figure 3.1 : General framework for agent based personalised searching
- Figure 3.2 : A conceptual framework of the PSA
- Figure 3.3 : A simple illustration of the category concept
- Figure 3.4 : A sample XML Schema
- Figure 3.5 : A sample XML representation of category item values
- Figure 3.6 : Multi layer architecture of the PSA
- Figure 3.7 : Personalisation graph for multilayer PSA architecture
- Figure 3.8 : Sequence diagram depicting PSA functionalities
- Figure 4.1 : System sequence diagram; User request a search using a new keyword
- Figure 4.2 : System sequence diagram; PSA client side search using a keyword that has reference in the PSA database
- Figure 4.3 : System sequence diagram; Users browse the search results for desired information in the PSA client database
- Figure 4.4 : System sequence diagram; PSA client updates its category table
- Figure 4.5 : System sequence diagram; Client requests for a search that has no reference in server database
- Figure 4.6 : System sequence diagram; Client requests for a search that has reference in server database.

- Figure 4.7 : System sequence diagram; Server updates its category table
- Figure 4.8 : Class diagram for the whole PSA architecture.
- Figure 4.9 : ER-Diagram for databases in the PSA Client side
- Figure 4.10 : ER-Diagram for databases in the PSA Server side
- Figure 4.11 : PSA search interface page
- Figure 4.12 : Detailed search result view, either of a dynamic page created or of a new search
- Figure 4.13 : Interface for results in multiple categories or category items
- Figure 4.14 : Interface for the display of details of a single URL
- Figure 4.15 : Interface for a search, with a keyword that is not either in local or global search databases
- Figure 4.16 : Tree hierarchy view of the category structure
- Figure 4.17 : Detailed view interface for updating category details
- Figure 5.1 : Graph depicting the relevancy of the results and the response time for delivering relevant results comparing the PSA with other candidate search engines

LIST OF TABLES

- Table 3.1 : Representation of Local and Global Categories
- Table 3.2 : A sample representation of Global Preference Database
- Table 5.1 : Assumed preferences against test keywords
- Table 5.2 : Test results of the PSA for keyword ‘Job’
- Table 5.3 : Test results of the PSA for keyword ‘Fee’
- Table 5.4 : Test results of the PSA for keyword ‘Thesis’
- Table 5.5 : Test results of the PSA for keyword ‘Scholarships’
- Table 5.6 : Test results of google.com for keyword ‘Job’
- Table 5.7 : Test results of google.com for keyword ‘Fee’
- Table 5.8 : Test results of google.com for keyword ‘Thesis’
- Table 5.9 : Test results of google.com for keyword ‘Scholarships’
- Table 5.10 : Test results of AltaVista.com for keyword ‘Job’
- Table 5.11 : Test results of AltaVista.com for keyword ‘Fee’
- Table 5.12 : Test results of AltaVista.com for keyword ‘Thesis’
- Table 5.13 : Test results of AltaVista.com for keyword ‘Scholarships’
- Table 5.14 : Test results of Yahoo.com for keyword ‘Job’
- Table 5.15 : Test results of Yahoo.com for keyword ‘Fee’
- Table 5.16 : Test results of Yahoo.com for keyword ‘Thesis’
- Table 5.17 : Test results of Yahoo.com for keyword ‘Scholarships’
- Table 5.18 : Relevancy index of the PSA and the candidate search engines from a research student perspective
- Table 5.19 : Averages of the time taken by the PSA and the candidate search engines under various search criteria
- Table 5.20 : An effort to find new indicators for relevancy based on the relevancy factor and the time

LIST OF ACRONYMS

ASP	: Active Server Pages
CADCS	: Call Attempt Data Collection System
CASE	: Computer-aided Software Engineering
CLR	: Common Language Runtime
COBOL	: Common Business Orientated Language
DOM	: Document Object Model
DTD	: Document Type Definition
ECMA	: European Computer Manufactures Association
FOCI	: Flexible Organiser for Competitive Intelligence
GUI	: Graphical User Interface
HTML	: Hyper Text Markup Language
HTTP	: Hyper Text Transport Protocol
KGA	: Keyword Generating Algorithm
LIFO	: Last In First Out
LRA	: Link Ranking Algorithm
LSR	: Life Saving Rank
OOA	: Object-Oriented Analysis
OOSD	: Object-Oriented Software Design
PSA	: Personalised Search Agent
RDBMS	: Relational Database Management System
RPC	: Remote Procedure Call
SER	: Search Engine Ranking
SOAP	: Simple Object Access Protocol

SQL : Structured Query Language

TCP/IP : Transmission Control Protocol / Internet Protocol

UDDI : Universal Description Discovery and Integration

URL : Uniform Resource Locator

UML : Unified Modelling Language

VB : Visual Basic

WSDL : Web Services Description Language

XML : Extensible Markup Language

XSLT : Extensible Stylesheet Language Transformation

CHAPTER 1 INTRODUCTION

Search engines are essential to the success of the Web. To some they might appear to fit the description Shakespeare applies to Gratiano:

*“Gratiano speaks an infinite deal of nothing, more than any man in all Venice. His reasons are as two grains of wheat hid in two bushels of chaff: you shall seek all day ere you find them, and when you have them, they are not worth the search.”*¹

However, the truth is probably closer to John Dryden's words from the seventeenth century, especially for those who use search engines skilfully and with some persistence:

*“Errors, like straws, upon the surface flow; He who would search for pearls must dive below.”*²

Undoubtedly, the Internet is the most abundant source of information in the present world, covering most domains of knowledge. Moreover, the information available through the Internet is constantly growing. As a result, the search for information interesting to a user has become a time-consuming and laborious task since this activity involves the analysis and separation of interesting pages from a great set of candidate pages (Analia & Daniela, 2000). Search engines are the most widely used tools for information searching on the web. Users provide a set of keywords to these search engines and wait for a set of links to webpages that contain those words. This mechanism is solely based on words (keywords) supplied by the user to the search engine. Therefore, getting the desired information largely depends on framing suitable keywords. The simpler the keywords, the more general (less precise) the results are. This causes inexperienced/novice users to spend considerable amounts of time searching for specific information. In terms of today's “*economy sensitive*” business world, this is a waste of human resources as well as hardware and software resource, energy and time, making it a pullback force in productivity. Users can make effective keyword combinations, by trying different combinations of search string words for better results. However, this is not an efficient method because the activity takes time and effort. For example, imagine a user looking for information about ‘software agents’ using a search engine, keying the keyword ‘agents’.

¹ Quotes from Shakespeare's *The Merchant of Venice*, Act I. Scene I. Retrieved 2 March, 2005, from <http://www.bartleby.com/70/1911.html>

² Quotes from John Dryden's ‘*All For Love*. Prologue’. Retrieved 2 March, 2005, from <http://www.bartleby.com/100/191.76.html>

This would result in the search engine returning URLs with information about travel agents, marketing agents, software agents, insurance agents and so on, clearly, far too general for the user. If the user gives the keywords, 'software agents' more specific results may be produced from a search engine. However, with knowledge of the user preferences and search history, a personalised search technique could generate the relevant results (i.e. regarding 'software agents') via the general keyword, 'agent'.

Again, search engines have limitations in regards to the classification of information because of the need for human intervention in the process, especially, considering the enormous growth of information on the Internet. The technologies supporting the information on the Internet (like HTML, XML, etc.) have provisions to include meta-information (information about information) but the inclusion of the meta-information is not mandatory. If search engines or other search mechanisms on the Internet rely on meta-information, the number of search results from those services could be very low. Such services clearly have chances to bypass valuable information channels and resources, as the number of webpages without meta-information well exceeds those with meta-information.

This thesis is an effort to identify technological gaps in personalising information searches on the Internet and to suggest a solution, through the development of a conceptual model - Personalised Search Agent (PSA). PSA is an attempt to fill the gaps in the current theoretical framework in order to deliver a more personalised search experience to users, providing better responsiveness, efficiency and user-friendliness.

1.1 Background and Motivation

The current practice of information searching on the Internet, although widespread, still suffers from drawbacks in regard to personalisation and user adaptability. Since search engines are the tools used to seek information on various online resources over the Internet, it is quite interesting to see how and to what extent search engines support users by delivering personalised results. On the other hand, search engines cannot be blamed for delivering less personalised results, as it is important for search engines to maintain a general user perspective about the information search rather than recording each user perspective.

This situation obviously leaves to a gap in the current information search technologies, allowing room for personalisation of search engine results. Again it does not sound sensible to make a new search engine for each unique user perspective, because most of the current search engines do their job of finding information quite effectively. The problem is that the results returned from searches are too numerous, too general and largely irrelevant, though they do deliver relevant results scattered within a pool of general results. An attempt to create a new search engine with a unique user perspective would be *reinventing the wheel*, because the need is not for develop a new search engine but for a personalising technique.

Though many attempts have been made to personalise search engine results, some issues remain unresolved, needing a more refined personalising tool, which can act as an intermediary between the user and the search engines. This study is motivated by the possibility of filling the gaps in the current practice of personalising the search engine results.

1.2 Problem Definition and Research Questions

The major issue under consideration in this research work is, how to develop an effective technique to personalise search results so that users can achieve more responsive results, organised in a way users want to view it. Apparently, to be successful, one needs to clearly understand the inadequacies in the current system. Moreover, it is a research requirement to define and frame the research questions to indicate both the focus and the limitations of the research. Following is a list of problems in the current practice of information search on the Internet using search engines.

- Search Engines most often return the results we need but they are normally lost in a large pool of irrelevant results. The effect of this is that much user-time is wasted browsing through the URL pool to find the relevant results.
- Search Engines mostly allow users to indicate *what they need* (in the form of search strings or keywords). However, they are not very helpful in taking user input about what kind or type of information is needed.
- Little effort is made to learn user preferences by capturing user feedback.
- Search engines have limited support to assist the user in forming appropriate keywords for a search, normally a crucial factor in achieving the right results.

- Search Engines or other personalising tools which act on search engine results do not generally have a mechanism to store the URLs which the user has visited during a successful search, so that those URLs can be retrieved for a similar search.
- Though meta-search engine facilities are available with some search engines or search filtering tools, an effective way of combining multiple search engine results with other personalising techniques is still lacking.

In view of these problems and the current information search practices, the following research questions form the basis of the thesis.

- a. **What are effective ways of finding user relevant information from search engine results?**
- b. **What are possible ways of capturing user feedback, which can help improve the personalisation of the information search without compromising usability issues (like *ease of use* and *minimum number of clicks*)?**
- c. **How can data be captured efficiently from user search practices and how can user preferences be learned from the captured data? How can user preferences be remembered or retained, rather than users being required to recall their preferences each time?**
- d. **How can the user be assisted to form effective keywords, resulting in efficient information searches?**

1.3 Research Aims and Objectives

The major aims of this research work are to develop a conceptual model and implement it by developing a prototype and to validate the prototype against current information search practices. The principle objective is to develop a well-structured conceptual model, which easily fits into the current theoretical framework of information search practices on the Internet. However importance has also been given to the development and implementation of a prototype and to its testing against a variety of quality factors, so that the new idea can stand on the credibility of the test results. Moreover, prototyping seems to be the only way to achieve internal validity for a short-term information technology research project like this. Only core functionalities and features in the conceptual model are implemented in the prototype.

The conceptual model addresses all the issues raised in Section 1.2, but particularly focuses on the research questions. Some models have been developed with the similar intention of personalising information searches on the Internet by using search engines as a backend resource or by using their own techniques for web search. This research takes some of the concepts used by them, which are discussed in detail in Chapter 2, to model the Personalised Search Agent (PSA).

1.4 Organisation of the Thesis

This section outlines how the thesis is organised. The thesis is organised in six chapters as follows.

Chapter 2 covers the literature review. It documents the context, relevant research and the understanding of the research environment. This chapter concludes with a comprehensive summary. Chapter 3 proposes the agent based model for personalised search. The idea of PSA is discussed in this chapter with complete development of conceptual model for the PSA, along with the techniques used (such as ranking) which together form the conceptual model for the PSA. Chapter 4 designs and develops a prototype from the conceptual model of the PSA. This chapter also describes how the prototype is developed including requirement analysis and modelling using UML. Chapter 5 tests the prototype and benchmarks it against other existing technologies. This chapter also summarises the test results against standards and criteria. Chapter 6 concludes the thesis, summarising the achievements and experiences. It also summarises future expansion of the current work.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews relevant literature which involves identifying, locating, synthesising and analysing the conceptual literature, as well as completed research reports, articles, conference papers, books, theses, and other materials related to the research area. A literature review also presents a comparison of ideas and research findings. The theoretical framework for the proposed research study should emerge from the literature review. According to Marshall and Rossman (1995, p.28) “*a thoughtful and insightful discussion of the literature builds a logical framework for the research that sets it within a tradition of inquiry and a context of related studies*”. In other words, the literature review provides background and context for a study. It also assists the researcher in understanding the problem in its context. Very importantly, the literature, along with the findings of the research reported in the literature, can help in the choice of research methods for the study (Kristy, 2000).

2.1 Research methodologies

This section describes the choice of methods in relation to the problem definition. It also gives an insight into how the research has been conducted and in what manner the results have been reviewed and selected. Following are some candidate research methodologies considered.

2.1.1 Experimental research

Experimental research is undertaken when the research wishes to trace a cause-and-effect relationship among defined variables. There are three broad types of experimental research design; true experiment, pre-experimental research and quasi-experimental design. Experimental research tradition is based on hypothesis testing by a deductive process of logical inference, where reasoning proceeds from general principles to particular instances.

Due to the rigorous controls exercised and the ability to rule out rival explanations, true experiments have much higher internal validity than other research designs. However, true experiments have poor external validity as they are conducted in rigidly controlled laboratory conditions and hence their generalisability to other populations is very limited. Moreover, true experiments are always carried out in unnatural conditions. To incorporate external validity, a field study is often conducted after a true experiment, though a field study is not a cost

effective method. Pre-experimental research though a weak methodology, is acceptable for an exploratory study, where the research wishes to gain insights or gather ideas, and not to generalise to a wider population. The quasi-experimental method is selected where it is not possible to use a true experimental design, and where external validity is a more important factor than internal validity. Nevertheless, the internal validity of the quasi-experimental method still remains much lower than that of the true experiment (Kristy, 2000).

2.1.2 Action research

There are numerous definitions of action research. One of the most widely cited, however, is that of Rapoport, who defines action research in the following way: “Action research aims to contribute both to the practical concerns of people in an immediate problematic situation and to the goals of social science by joint collaboration within a mutually acceptable ethical framework” (Rapoport, 1970, p. 499). This definition draws attention to the collaborative aspect of action research and to possible ethical dilemmas, which arise from its use. It also makes clear, as Trist (1976) emphasises, that action research is concerned to enlarge the stock of knowledge of the social science community. It is this aspect of action research that distinguishes it from applied social science, where the goal is simply to apply social scientific knowledge but not to add to the body of knowledge. Action research has been accepted as a valid research method in applied fields such as organisation development and education. In information systems, however, action research was for a long time largely ignored, apart from one or two notable exceptions. More recently, there seems to be increasing interest in action research with respect to information systems.

2.1.3 System development research

The System Development method is a relatively new and not very popular method for conducting research on information systems. First proposed in the early 1990s, this is a systematic approach to information systems research, which includes some systems development. The systems development approach denotes a way to perform research through exploration and integration of available technologies to produce an artefact, system or system prototype. Systems development focuses on theory testing, more than theory building aspects of research, allowing a smooth progression from development to evaluation. It could be thought of as proof-by-demonstration (Nunamaker, Chen & Purdin, 1991). On the other hand,

it can be useful to consider as part of the exploratory stage of information systems study, when the aim is to observe and evaluate the implications or any other effects of introducing a particular new technology into the organisation. The system development research process is of an iterative nature, as illustrated in Figure 2.1.

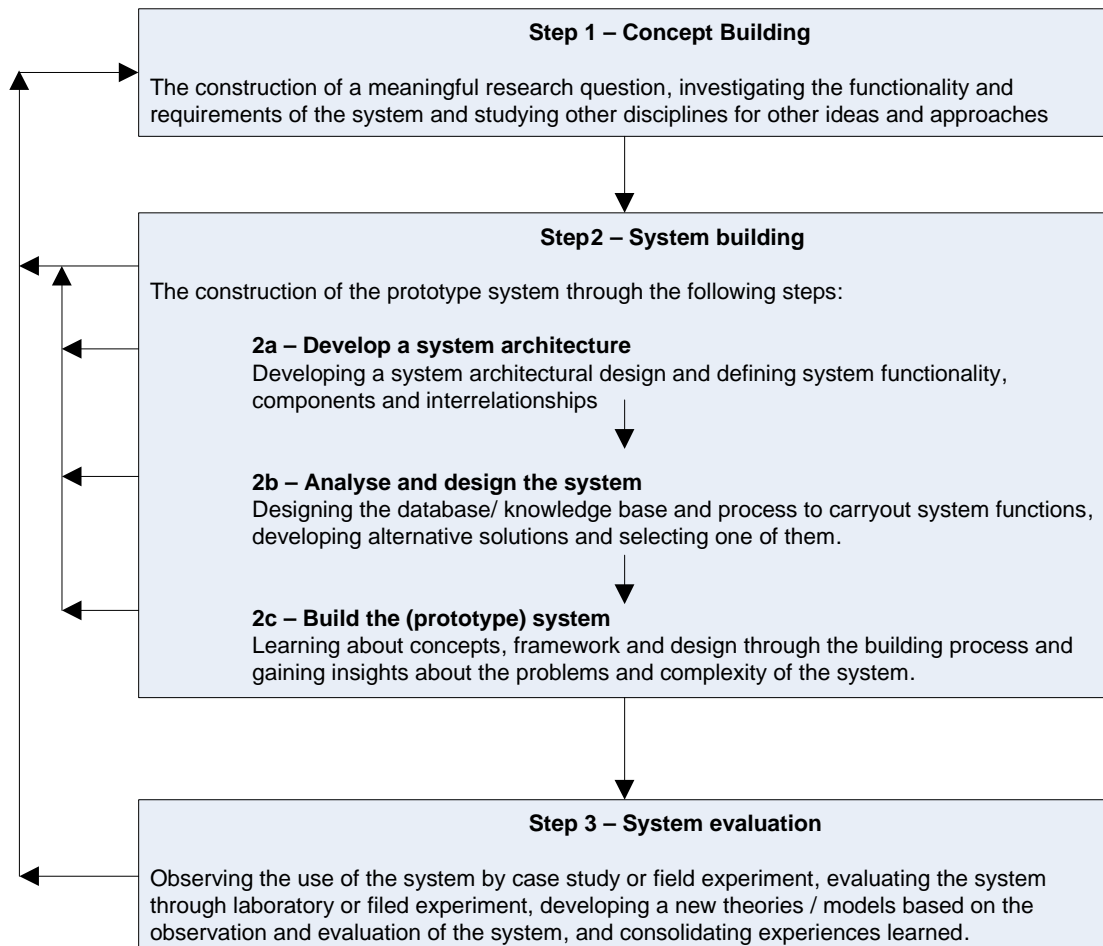


Figure 2.1: System Development Research Method (Kristy, 2000).

The System Development Research Method was chosen as the preferred method for this research because of the following reasons.

- Research in information systems has been criticised as being too conceptual
- Due to rather applied nature of the work, system development is essential to prove underlying theories.
- To be a valid research activity, the existing systems and concepts need to be compared against the proposed conceptual idea and apparently needs a prototype to proof and test the new concept.

The flaws in the new concept can only be identified and improved in the process of system development.

2.2 Information searches on the Internet

According to Convey (1992), the act of information searching depends on the information that has been stored. Both the storage and retrieval procedures consist of three operations.

Storage:

- Subject analysis of a document by an indexer.
- The translation of the subject analysed into the database's indexing language.
- The organisation of the files of which the database is comprised.

Retrieval:

- The analysis of the search question.
- The translation of the concepts contained in the question into the indexing language of the database.
- The formulation of the search statement, that is, the relationships among terms and the commands to be used that are usually expressed in "Boolean" logical statements.

According to Harter (1986), the online searching process can be represented as a set of discrete but interrelated steps, to be carried out approximately in the order given below:

- a. Understand the information need. Here, attention must be given to distinguishing between what the user needs as opposed to what he/she wants or says he/she wants.
- b. Formulate search objectives. What will the search attempt to accomplish? The listing of a comprehensive bibliography? The discovery of a fact?
- c. Select one or more databases and search systems.
- d. Identify major concepts or facts and indicate their logical relationships to one another.
- e. Select an overall approach or strategy for attacking the information problem.
- f. Identify a variety of ways to express the concepts in words, phrases, symbols, etc., expressed in natural language, descriptors, subject headings etc.
- g. Identify the fields of the records that will be searched in the databases selected.
- h. Translate decisions made in steps (b) to (g) into formal statements expressed in the command language of the search system.
- i. For each of the steps (b) to (g), consider and plan alternatives, in case initial attempts do not meet search objectives.

- j. Logon to the search system of choice and enter the initial search statement formulated in step (h).
- k. Evaluate the intermediate results against the search objectives.
- l. Iterate. That is, on the basis of the results of the evaluation obtained in step (k), and considering the alternatives planned in step (i), as well as new ideas obtained while searching. On the basis of system feedback, decide whether to print the results and stop, or keep going. The searcher might return to any of the steps, perhaps even to step (a). The process continues until satisfactory results are obtained.

A central issue to several of the steps in the search process is the concept of communication, whether this is binary communication between host computer and user terminal or verbal communication between end user and search specialist.

2.2.1 Search engines and how they work

In the WDG's glossary of terms (1997), the term search engine is defined as a system dedicated to the search and retrieval of information for the purpose of cataloguing the results, usually based on an index of several HTML documents, so that the document(s) being sought can be easily located. According to Cooke (1999), search engines are the most popular way of searching for information through the Web. Search Engines, sometimes also called spiders, robots or crawlers, are supported by machines, automatically generating databases with Web pages by constantly searching and visiting different sites and thus automatically indexing them. Remember that a Web page has to be indexed before the engine can "make a copy" of it and thus display it to the user. If a Web page changes, the search engine will find those changes, which may affect what the "hit-list" looks like since a search engine takes into consideration page titles, body copy and other elements. (Chignell et al., 1999) Indexed search engines are often called dummies by professional searchers because of the great amount of unstructured sites/pages with little or no relevance they present to the user after a search. It is also possible to program a Web site so that a user gets the URL to a site in which content has no relevance for the user.

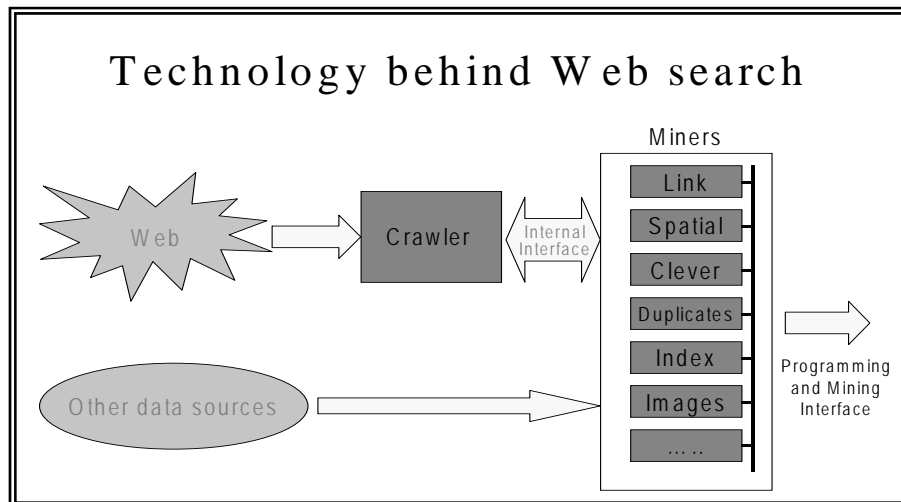


Figure 2.2: Technology behind the web Search (Linell & Rose, 2001)

Search engines often retrieve so much information that they grow into an impressive size. Their retrieval process is achieved by matching words in the pages from huge databases of Web page texts from the World Wide Web. Unfortunately, they retrieve everything they come across, whether it is good, bad, wanted or unwanted information and without prioritising or distinguishing between the quality in the found material. Because of this it is important to evaluate everything before using it.

Search engine architectures

There are three main parts that are perceptible in a search engine.

- The spider or crawler. The act of “crawling” is performed when the spider visits a Web page, reads it and then follows its links to other pages within the site. The spider also returns on a regular basis to the site looking for changes.
- The index or catalogue is the second part of a search engine. It works like a giant book, which lists a copy of every Web page found by the spider. The “book” is then updated whenever a Web page changes. Even though a page has been “*spidered*”, it will not be accessible until indexed. An example of index architecture is shown in Figure 2.2.
- Search engine software is the third component of a search engine. It is the program that sifts through all the millions of pages recorded in the index to find matches to a search which are then ranked into what is believed to be the most relevant order (Search engine architecture, n.a).

Subject directories

Subject directories are selections of Web pages (most of the time) already evaluated and organised into subject categories or catalogues that are maintained by people. A Web page can end up in a directory in two ways; either its creator submits the site and a brief description of it to the maintainer of the directory, or the maintainer can find good webpages that he/she wishes to add. When doing a search the engine looks for matches only in the description submitted (Chignell et al., 1999). It is possible to browse or search the contents of those directories in an organised manner through a hierarchy of subject headings, but the search is restricted to the pages that are part of the catalogues and does not extend to the entire Internet. As a result, changes made to a webpage do not alter the list. The content has already been reviewed which leads to more trustworthy content than found in the search engines. This is true for as long as the directory does not grow too large. If this happens, the selectivity and quality do generally get worse. Subject directories can list searchable databases and gateway pages. Professional users call these engines “intelligent engines”, since they are constantly maintained by people (Recommended subject directories, 2005).

Directories of searchable databases

Directories of searchable databases are organised into subject databases the same way as subject directories, are able to search and browse, and are built by humans who most of the time evaluate the content and give pointers on the use of each of the listed databases. Another feature these directories offer is search boxes and links to all kinds of databases covering the entire world. This makes it possible for to search (and find) information which cannot be found in general Web pages, and which is therefore impossible to find through search engines or subject directories. These specialised databases retrieve an enormous number of documents that are often called “invisible” or “deep” Web. Visible Web is the result received from a general search engine such as Google, AltaVista, Infoseek, FastSearch and NorthernLight. Invisible Web is what results from a search of a specialised searchable database. Unlike general Web pages found in search engines, database pages are dynamically generated for a certain search and are not stored anywhere. These dynamically generated pages are not found by engine spiders. For these pages it is necessary to go a search box belonging to that specific database (What is the "Invisible Web"?, 2004).

Directories of gateway pages

Directories of gateway pages described as collections of general Web pages of unique value. Basically, they are gateway pages or subject guides with many links covering subject areas, disciplines or fields. They are organised by subject and can be searched as well as browsed. Each directory is elaborated on by an “expert” who spends much time searching the Web and assembling guides to a field, subject, discipline, etc. Sometimes the division is done using a classical subdivision of the discipline and sometimes using the formats in which the information is found. Sometimes a unique organisation developed by the expert is used. There are several specialised directories or guides to these kinds of pages also compiled by experts (General Subject Directories, 2005). The directories are often made by libraries or non-profit organisations or by individuals. Some have a commercial approach and are crowded with advertisements and other things. The main intention, however, is always to make it easier for the user to see, understand and manage the content of the Internet. The value of directories of gateway pages can be perceived, for example, while doing academic research.

Meta Search engines

Meta search engines, meta-crawlers or multi search engines were developed to improve search performance by querying multiple search engines at the same time. They can simplify a search by selecting a subset of first-level search engines and digital libraries to submit queries to based on the characteristic of the user, the query topic, and the search strategy. The named selection can be guided by diagnostic knowledge of the type “which of the first-level search engines works best under what circumstances” (Chignell, Gwizdka & Bodner, 1999). All the results are brought together in a convenient display, often a homogeneous format and listing. (Björkman & Ohlsson, 2000) Meta search engines do not have a database of Web pages on their own but rely entirely on other search engines. When a search is performed, the keywords written in the search box are transmitted simultaneously to several different search engines and their databases of webpages. The listing of the results comprises the results from all the search engines queried.

In spite of being quite good for a first search meta search engines are not suitable for more serious and in-depth searches, since they do not have the same search power as other search

engines. They are mostly used for simple searches using only one or two terms and usually do not search the largest search engines. However, the meta search technology is constantly being improved, so this situation might soon change (Meta search engines, 2004). Three main factors, determine the usefulness of a meta search engine:

- a. The search engines to which the meta search engines send the user's search terms (size, content, number of search engines and user ability to choose the preferred search engines).
- b. How the meta search engines handle the user's search terms and search syntax (Boolean operators, phrases, and defaults they impose).
- c. How meta search engines display results (ranking; aggregated to one list, or with each search engine's results reported separately) (Meta search engines, 2004).

2.2.2 Popular search engines

The following section presents a detailed analysis of 3 popular search engines, namely, google, altavista and Yahoo. The analysis reviews the techniques followed by each of these search engines along with their strengths and weaknesses.

Google

Larry Page and Sergey Brin, two Stanford Ph. D. candidates who developed a technologically advanced method for finding information on the Internet, founded Google in 1998. Google is a privately held company, whose backers include Kleiner, Perkins, Caufield & Byers and Sequoia Capital. (Review of Google, 2004)

According to Google, the site focuses exclusively on delivering the best search experience on the World Wide Web. Through innovative advances in search technology, Google helps users find the information they are looking for quickly and effectively. The company delivers services through its own Web site at www.google.com, and by licensing its search technology to commercial sites. Google's page ranking relies on the uniquely democratic nature of the Web by using its vast link structure as an indicator of an individual page's value. In essence, Google interprets a link from page A to page B as a vote, by page A, for page B. But, Google looks at more than the sheer volume of votes, or links a page receives; it also analyses the

page that casts the vote. Votes cast by pages that are themselves "important" carry a greater weighting and help to make other pages "important".

Google combines Page Rank with sophisticated text-matching techniques to find pages that are both important and relevant to a search. Google goes far beyond the number of times a term appears on a page and examines all aspects of the page's content (and the content of the pages linking to it) to determine if the page is a good match for a particular query (Review of Google, 2004).

AltaVista

The development of AltaVista began in the summer of 1995 at Digital's Research Laboratories in Palo Alto, California, and was formally delivered to the Web on December 15, 1995 (Review of AltaVista, 2003). AltaVista indexes the full text of over 550 million full text Web pages with unspecified update frequencies. According to its documentation, AltaVista can fetch 2.5 million pages a day following the Robots Exclusion Standard, and index one gigabyte of text per hour. AltaVista supports Boolean searching, both term and phrase searching (i.e., proximity searching with the NEAR operator), field searching (e.g., title: steelhead; url: home.html), right-hand truncation with some restriction, and case-sensitive searching if only the first letter of a word is capitalised.

AltaVista is developing 'AltaVista Categories', which provide access to resources categorised into various subject areas. AltaVista was the first search engine to offer multilingual searching capabilities, including the ability to search in Chinese, Japanese and Korean, as well as European languages such as French, German, Italian, Russian and of course English. AltaVista provides three display options: compact, standard, and detailed, although the last two are the same. The display order or relevancy ranking of search results is determined by the location (e.g., in the title or the body of the text) of matching words, occurrence frequencies of matching words, and distance (i.e., how many words apart) between the matching words. However, only the first few words of a document found are displayed, which may limit users' ability to judge its relevancy without referring to the full version of the document. In addition, general search terms such as "computer" and "analysis" are automatically ignored in Alta Vista (Review of AltaVista, 2003).

Yahoo

Yahoo started as an idea, grew into a hobby and has lately turned into a full-time passion. The two developers of Yahoo, David Filo and Jerry Yang, Ph.D. candidates in Electrical Engineering at Stanford University, started their guide in April 1994 as a way of keeping track of their personal interests on the Internet. It did not take long before they found that their home-grown lists were becoming too long and unwieldy. Gradually they began to spend more and more time on Yahoo (Review of Yahoo search, 1999).

During 1994 they converted Yahoo into a customised database designed to serve the needs of the thousands of users that began to use the service through the closely bound Internet community. They developed customised software to help them efficiently locate, identify and edit material stored on the Internet. The name Yahoo is said to stand for "Yet Another Hierarchical Officious Oracle" but Filo and Yang insist they selected the name because they considered themselves yahoos. Yahoo first resided on Yang's student workstation ("akebono") while the search engine was lodged on Filo's computer "konishiki" (These machines were named after legendary Hawaiian sumo wrestlers).

In early 1995, Marc Andressen, co-founder of Netscape Communications in Mountain View, Ca., invited Filo and Yang to move their files over to larger computers housed at Netscape. Today, Yahoo contains organised information on tens of thousands of computers linked to the Web. Yahoo is an example of a subject directory. Indeed, Yahoo claims to be the oldest and largest directory, listing over 750,000 Web sites divided into more than 25,000 categories. Users suggest most of the sites, and each suggestion is examined and evaluated by a member of the Yahoo staff, whose job it is to decide where the site best belongs. In Yahoo it is possible both to search and browse (Cooke, 1999).

2.2.3 Issues in information searching using search engines

The enormous amount of information and web pages on the Internet apparently makes the search engines to return relatively high volumes of data. Following is a list of the most common problems with search engines based general observations:

- Difficulties in expressing the right query, or, finding the words that will give the best results.

- Too many search hits, most of which are not relevant, particularly when Yahoo is used.
- The existence of many “dead” links or links that are not updated on the Web, particularly AltaVista.
- Sometimes a search engine is too limited or simple, which makes combined searches difficult or impossible. One problem arises when a search engine does not work properly with Boolean operators (must use + and -), as in the case of Yahoo.
- Information is often presented in an unstructured way and is graphically confusing, making an overview difficult.
- Advertisements are disturbing and impossible to “shut off” (AltaVista and Yahoo).
- Help functions are too complicated or are of no help, particularly Yahoo Search guide.
- There is no historical resume (search path) listed so it is easy to get lost in cyberspace.
- Pictures and backgrounds take too long to download, because they might be too big.
- Web pages are seldom adapted to people with special needs.

2.3 What is personalisation in information searching?

As per the Oxford dictionary, word *personalisation* is a noun derivate of the verb personalise meaning, design or produce (something) to meet someone’s individual requirements.

Personalisation involves creating systems that are user-centric and have a 'unique user' sense. This means systems that can adapt to the user, learn about the user, and provide unique data views that are personal to that user. Clearly this encompasses different areas - including User Modelling, Machine Learning, Context Sensitivity, and in terms of the web, hypermedia adaptation, including information searching and retrieval (John, 2003). Web-based personalisation has been in existence almost as long as the World Wide Web itself. Moreover, most of the same problems that personalisation aims to solve were present at the outset, albeit to a much lesser extent. With more than 600 million people online, and at least 2.5 billion web pages and web services such as search engines are struggling to keep up (Trends & Statistics, 2005), effective personalisation is more important today than ever before .

There are two general approaches to personalisation of web searching, each with different levels of sophistication:

- Collaborative approaches in which personalised information is presented to users based on the actions of similar users in the past.
- Learning approaches, which learn user behaviour by observing the system interactions of users.

These approaches can be the most accurate and reliable for all users. However, it is a relatively new area and much research must be carried out to develop efficient techniques. Personalisation is the way forward; however, due to the enormous number of users and the volume of data involved, complex personalisation applications that require very deep user models are not feasible. Simpler models are efficient and can produce reasonably accurate results (John, 2003).

The common evaluation method applied in web search systems is *precision and recall*, which usually requires relevance feedback from users. However, obtaining relevance feedback explicitly from users for personalised web search systems is extremely challenging due to the large size of the Internet (Cyrus & Yi-Shin, 2003).

According to Cyrus & Yi-Shin (2003) other factors affect personalisation of web searching/searches. They are, personalised page importance, query refinement and meta-search capabilities.

- **Personalised Page Importance:** In addition to the traditional text matching techniques, modern web search engines also recognise the importance of page ranking in the search results. The most famous example is the Page Rank algorithm, which is the basis for all the web search tools of Google. In topic-sensitive Page Ranking, the system first pre-computes web pages based on the categories in Open Directory. By using these pre-computation results and the favourite pages, the system can retrieve “topic-sensitive” pages for users (Haveliwala, 2002).
- **Query Refinement:** Generally, the query refinement process consists of Obtaining User Profiles from User, Query Modification and Refinement of the results
- **Meta-search Systems:** It has been reported that search engine coverage decreases steadily as the estimated web size increases. In 1999, no search engine could index more than 16% of the total web pages (Lawrence & Giles, 1999). Consequently, data searching using only a single search engine could result in a very low retrieval rate. Meta-search systems can solve this problem by incorporating several search engines in

the search process and smartly combining the results from different search engines by user rankings.

2.3.1 Strategies in information retrieval

The two major pathways for information retrieval are search and browse. The remainder of this section will discuss these in detail along with the hybrid technique of effectively combining browsing and searching for information retrieval.

Browsing

In contrast to the formal analytical strategies developed by professional intermediaries, information seekers also use a variety of informal, heuristic strategies. These informal, interactive strategies are clustered together under the term *browsing strategies*. In general, *browsing* is an approach to information seeking that is informal and opportunistic and depends heavily on the information environment. Four browsing strategies are distinguished in this section: scanning, observing, navigating and monitoring. The term *browsing* according to Marchionini (1997) reflects the general behaviour that people exhibit as they seek information using any of these strategies.

Browsing offers significant challenges to information seekers and system designers. The challenge to the information seeker is to relate personal knowledge about the topic to what the system represents and the ways in which its representations are organised. The challenge for designers is to make clear the system's scope and organisation and to suggest entry points for the searcher. Furthermore, information must be examined and assessed during browsing. The challenge here for the information seeker is to provide flexible display facilities for examination and assessment.

Large, Tedd & Hartley (1999) indicate that browsing is an attractive proposition because it requires smaller cognitive loads than does an analytical search strategy. Browsing allows information seekers to devote their full cognitive resources to problem definition and system manipulation for ill-defined or complex tasks. Humans are better able to recognize something than to generate a description of it.

Browsing strategies

Some browsing strategies can be identified as part of the information seeking-process. Scanning and navigation are most often used in systematic browsing in which the objects are well defined in the information seeker's mind and environment. Observation and monitoring, on the other hand, are most often used in opportunistic browsing when the goals are exploration, learning or accretion of knowledge; when the objects are complex and vaguely defined or when the systems are unfamiliar or unstructured (Marchionini, 1997).

Scanning: Scanning is the most basic browsing strategy. Scanning implies that the information seeker compares sets of well-defined objects with an object that is clearly represented in the seeker's mind. Scanning is applicable to highly organised environments that provide clear and concise representations. It can proceed sequentially according to some structural feature of the content or through some sampling method.

Monitoring: Monitoring is most similar to scanning except that it tolerates poorly structured environments. For example, while reading text related to a specific topic, a monitor browsing strategy "listens" for concepts related to another topic of interest. Monitor strategies focus on attributes of interest to the information seeker and are less dependent on stimuli in the environment than are observational and navigation strategies.

Observation: Observational strategies are the most general of all browsing strategies since they have minimal thresholds for all the browsing dimensions except for cognitive effort. Browsers who use these strategies assume that they are in a promising neighbourhood and react to stimuli from that neighbourhood. Observational strategies depend on a great deal of parallel input. Like scanning, observational strategies are rooted in our physiological instincts. Observation does require interpretation and reflection to make sense of what is observed and to relate it to information-seeking objectives. Observations may lead to interesting discoveries but yield most initiation control of the environment.

Navigation: The navigation strategy balances the influence of the user and the environment. The environment constrains browsing by providing possible routes and the user exercises some control by selecting which routes to follow. Relatively high thresholds for all the browsing dimensions define navigation. Objects must be specifiable. Moreover, information seekers must know what they are seeking. They must also actively interact with the

environment and regularly reflect and make decisions about progress. Navigation as a browsing strategy refers to the ongoing observation of environmental attributes and adjustments to the mental problem representation based on searching that proceeds incrementally based on feedback from the information system. One critical factor in navigation is the way in which the system provides feedback.

Combination of search and browse strategies

Both searching and browsing strategies have their own role to play in ensuring that information seekers can locate relevant information as effectively and effortlessly as possible. According to Large et al. (Large et al., 1999) few information systems will fail to provide access to both these strategies. Conventional retrieval systems, utilising analytical search strategies, can offer at least minimal browsing capabilities.

2.3.2 Current work on personalised information searching

There have been some attempts to personalise web search results incorporating one or more of the techniques discussed above. The following three applications are chosen for discussion because of their relevance to the theoretical framework of this research.

User profile modelling and applications to digital libraries

Giuseppe & Umberto (1999) describe personalising information access in digital libraries through user profiles. They identify data categories as an efficient and essential tool in user profiling. They also discuss various ways to gather data categories and methods to capture user preferences, suggesting three unique ways, namely, the document content category, the document structure category and the document source category. Their study also covers how to specify user preferences by classifying the categories into two distinct subcategories, each addressing orthogonal delivering dimensions. The paper also explains a profile schema and architecture depicted in the Figure 2.3.

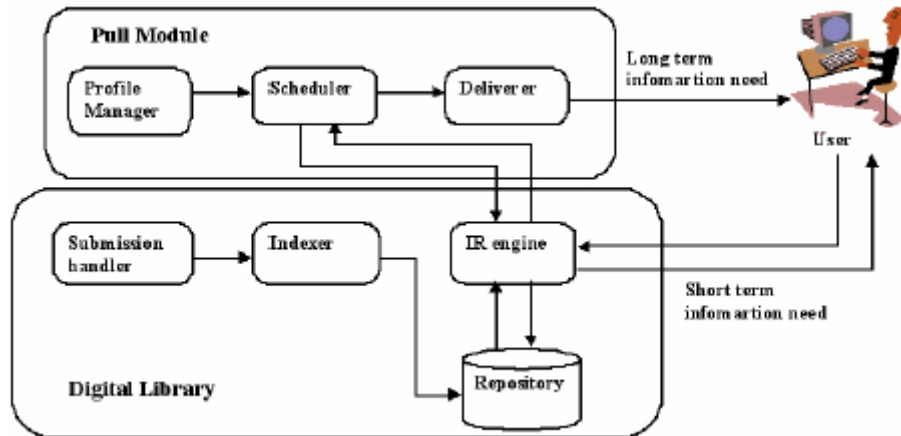


Figure 2.3: User profile architecture by Giuseppe & Umberto (1999).

This work clearly identifies data classifications into categories as an important step towards personalisation of information. Though the work deals primarily with personalising digital libraries, it is also relevant to personalising web searching.

User profiling for content personalisation in information retrieval

Boris, Parisch, Paul & Mícheál (2004) describe the importance of information categorisation and user profiles in web information search personalisation. They have done in-depth analysis of the existing standards for user profile modelling. They also suggest generic user profile modelling as depicted in Figure 2.4.

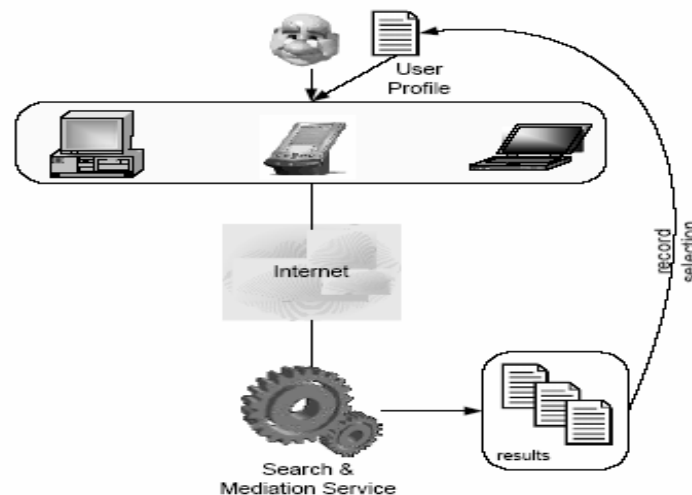


Figure 2.4: User Profile Modelling in Content Personalisation (Boris, Parisch, Paul & Mícheál, 2004).

They also emphasize the use of XML as a data structure for user profiles and have provided extensive examples of how to map user profiles with XML, XSLT and XML Schema.

A personalised web intelligence system

Ah-Hwee, Hwee-Leng, Hong Pan, & Qiu-Xiang (2001) introduce a system known as Flexible Organiser for Competitive Intelligence (FOCI) that provides an integrated platform for gathering, organising, tracking, and dissemination of competitive information on the web. The system enables users to build information portfolios by gathering and organising on-line information according to their needs and preferences. Through a method called user configurable clustering, users can personalise their portfolios in terms of the content and the information structure. The personalised portfolios can be constantly updated by tracking relevant information and new information can be organised into appropriate folders of the portfolios automatically. The personalised portfolios thus function as "living reports" that can then be published and shared by other users.

FOCI system employees a clustering engine to map the keyword against the category. It uses a complex algorithm based on fuzzy logic in its clustering engine, which helps the user to create and manage personal information portfolios. An architectural diagram of the FOCI system is shown below in Figure 2.5.

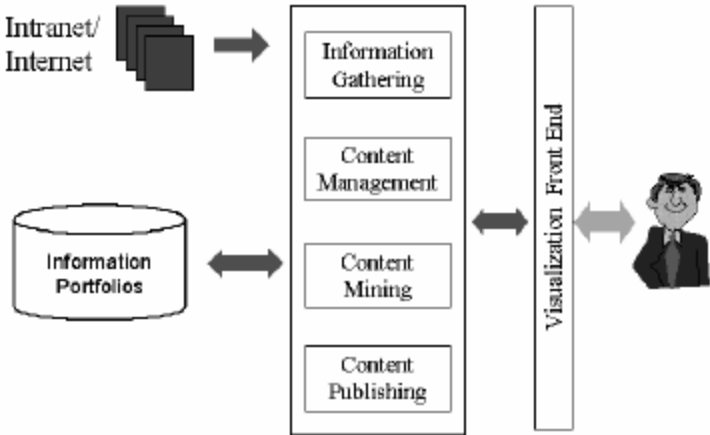


Figure 2.5: FOCI System Architecture (Ah-Hwee, Hwee-Leng, Hong Pan & Qiu-Xiang, 2001)

The first two information search personalising systems discussed above, namely *user profile modelling and applications to digital libraries* and *user profiling for content personalisation in information retrieval*, have the category concept and the user profile modelling applied in order to achieve personalisation. The third system discussed above, i.e. the FOCI system, is more advanced in terms of the techniques it applies for achieving personalisation in information search. The techniques that FOCI system follows are, category concept, user profiling and learning from user search practise. All the three systems discussed above take input either from the Internet directly, or take the input from a search engine. However they do not employ meta search engine techniques. Hence an effective combination of category concept, user learning approach and meta search engine approach is still lacking in the current systems for personalising web information search. An approach to overcome these limitations is described in chapter 3.

2.4 Agents and information searching

This section studies software agent techniques in detail and analyses how effectively they could be employed in information searching on the Internet.

2.4.1 Software agents

The word “agent” has become a buzzword among IT researchers over the recent past and is widely used among heterogeneous research domains. According to Stan & Art (1996), “A *software agent is an autonomous system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future*”. Software agents are different from computer programmes. Computer programmes are those which run in computer hardware take user inputs, process information and make necessary outputs. They are invoked by a user and controlled by a user. A user has to give necessary inputs for a programme to run smoothly and do its task. Software agents are also programmes and are always part of a particular environment. An agent senses its environment and acts autonomously upon it. No other entity (such as a user) is required to feed it input, or to interpret and use its output. Each agent acts in pursuit of its own agenda or pursuing goals designed by other software agents. Agents act so that their current actions may effect their later sensing, that is their actions affect their environments. Agents also act continuously over a period of time.

2.4.2 Agent classifications

Agents can be classified by their mobility that is their ability to move around the networks. Agents can also be classified by the nature of an action that is either deliberate or reactive. Deliberative agents derive from the deliberative thinking paradigm: the agents possess an internal reasoning model and they engage in planning and negotiation in order to have active coordination with other agents. Reactive Agents, in contrast, do not have any internal, reasoning model of their environment, and they act using a stimulus/response type of behaviour by responding to the present state of the environment in which they are embedded (Brooks, 1991).

Finally, agents can be classified according to several ideal and primary attributes which agents should exhibit, namely, autonomy, learning and cooperation. Autonomy refers to the principle that agents can operate on their own without the need of human guidance, even though this would sometimes be invaluable. Agents have individual internal states and goals, and they act in such a manner as to meet goals on behalf of the user. The key element of their autonomy is that they are proactive, that is, they are able to take the initiative rather than acting simply in response to their environment. Cooperation with other agents is paramount; it is the reason for having multiple agents in the first place rather than just one. In order to cooperate, agents need to possess a social ability, i.e. the ability to interact with other agents and possibly humans via some language. *“For agents to be smart they need to learn as they react or interact with the external environment. Learning makes the agents intelligent and also delivers increased performance over time”* (Wooldridge & Jennings, 1995). Based on the characteristic explained above, agents can be broadly classified as collaborative agents, learning agents, interface agents and smart agents. Figure 2.6 depicts these differences.

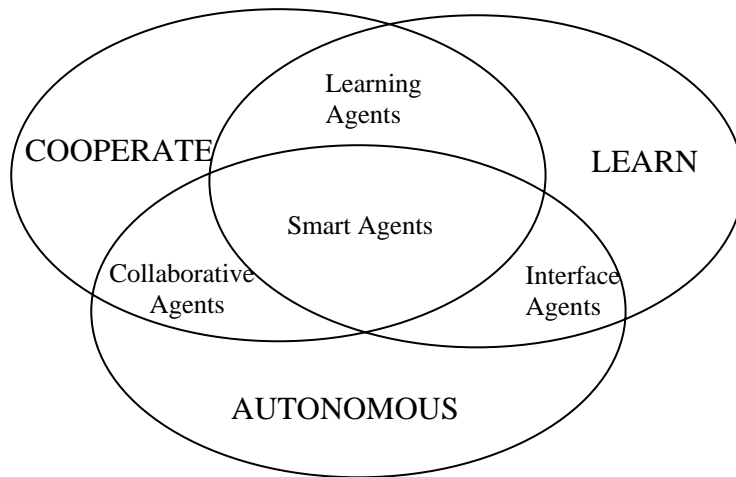


Figure 2.6: A graphical representation for agent classification. (Hyacinth, 1996)

2.4.3 Software agents in information searching

Because of the dynamic nature of the Internet and conventional methods of mining information on the Internet, using search engines may not always be quite fruitful. Information and information services on the Internet are very heterogeneous offering, in many ways, different kinds of formats.

When looking for information, the use of software agents has certain advantages compared to current search practices. Information searching is done based on one or more keywords given by a user. Agents are capable of searching for information more intelligently, because tools (such as a thesaurus) enable them to search also on related terms, or even, concepts. Individual user agents can create their own knowledge base about available information sources on the Internet, which is updated and expanded after every search. When information has moved to another location, agents will be able to find, and update their knowledge base accordingly.

Furthermore, in the future, agents will be able to communicate and cooperate with other agents. This will enable them to perform tasks such as information searches more quickly and efficiently while reducing network traffic. They will also be able to perform tasks directly at the source/service, leading to a further decrease in network traffic (Björn, 1997).

Agents can relieve the human user of the need to worry about *where* and *how* certain information may be found, leaving the user to worry only about *what exactly* is being sought. The client side agents (thick programmes) can greatly reduce the access limitations (like internet) because they can cache the previous search history and hence allowing the user to

search for information (limited though) offline. An agent can perform multiple tasks simultaneously in parallel. A good example would be the meta search agents which normally search two or more search engines simultaneously for a given user query. This saves a considerable amount of user time and effort.

Software agents will be able to search for information based on contexts. They will access this context from user information or by using other services, such as a thesaurus service. The agents can adjust themselves to the preferences and wishes of the individual users. Ideally this will lead to agents that will increasingly adjust themselves to what users want, and what they are usually looking for, by learning from performed tasks (i.e. searches) and the user reaction to the results.

In short, software agents are extremely useful tools both in information searching and in the personalisation of information searching. However, it does not seem to be reasonable to make an agent from scratch which is capable of searching for information directly from the Internet. Instead, it is more sensible to use currently successful search engines as a backend information search resource and use the agents to personalise the information from the search engines.

2.5 General architecture for information searching systems

This section describes general architectures for information search systems in a distributed environment, particularly client server architecture and multi-layer architecture.

2.5.1 Client server architecture

The term client/server was first used in the 1980s in reference to personal computers (PCs) on a network. The actual client/server model started gaining acceptance in the late 1980s. The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve *usability*, *flexibility*, *interoperability* and *scalability* as compared to centralised, mainframe, time sharing computing.

A client is defined as a requester of services and a server is defined as the provider of services. A single machine can be both a client and a server depending on the software configuration (Edelstein, 1994).

As a result of the limitations of file sharing architectures, the client/server architecture emerged. This approach introduced a database server to replace the file server. Using a Relational Database Management System (RDBMS), user queries could be answered directly. The client/server architecture reduces network traffic by providing a query response rather than total file transfer. It improves multi-user updating through a Graphical User Interface (GUI) front end to a shared database. In client/server architectures, Remote Procedure Calls (RPCs) or Structured Query Language (SQL) statements are typically used to communicate between the client and the server (Newell & Machura, 1995).

2.5.2 Multi-layer architecture

The three-tier software architecture (a.k.a. three layer architecture) emerged in the 1990s and an effective multi-layer architecture for distributed systems. The third layer (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with two tier architecture) by providing functions such as queuing, application execution, and database staging. The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability and scalability, while hiding the complexity of distributed processing from the user.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems. A three tier distributed client/server architecture, as shown in Figure 2.7, includes a user system interface top tier where user services (such as session, text input, dialog, and display management) reside.

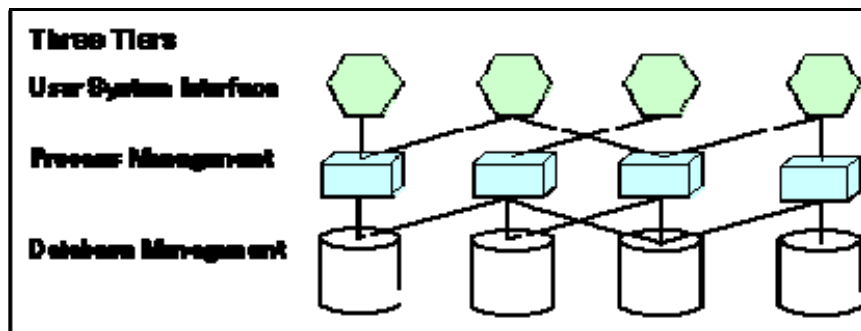


Figure 2.7: Three tier distributed client/server architecture depiction (Three Tier Software Architectures, 2004)

The third tier provides database management functionality and is dedicated to data and file services that can be optimised without using any proprietary database management system languages. The data management component ensures that the data is consistent throughout the distributed environment through the use of features such as data locking, consistency, and replication. It should be noted that connectivity between tiers could be dynamically changed depending upon the user's request for data and services.

The middle tier provides process management services (such as process development, process enactment, process monitoring, and process resourcing) that are shared by multiple applications. The middle tier server (also referred to as the application server) improves performance, flexibility, maintainability, reusability, and scalability by centralising process logic. Centralised process logic makes administration and change management easier by localising system functionality so that changes need only be written once and placed on the middle tier server to be available throughout the systems. With other architectural designs, a change to a function (service) would need to be written into every application (Eckerson, 1995).

Sometimes, the middle tier is divided in two or more units with different functions. In these cases the architecture is often referred to as multi-layer. This is the case, for example, with some Internet applications. These applications typically have light clients written in HTML and application servers written in C++ or Java. The gap between these two layers is too great to link them together. Instead, there is an intermediate layer (web server) implemented in a scripting language. This layer receives requests from Internet clients and generates html using the services provided by the business layer. This additional layer provides further isolation

between the application layout and the application logic (Three-tier architecture for client server technology, 2004).

2.6 Summary

This chapter has presented the context and the theoretical framework of the research along with insights into other studies relevant to the research. It streamlines and integrates the information gathered. The problem area and research objectives were defined followed by selecting a suitable method to conduct a research on information search personalisation. The candidate research methodologies considered were the Experimental Research Method, the Action Research Method and the System Development Research Method. After a detailed analysis of all the candidate methods, the System Development Research Method was chosen as the preferred method for this research.

To build up a rich background of the topic, a detailed study has been done understanding the current search practices in information search. As the popular method to find information online, is by using search engines, a detailed study has therefore been done, understanding their functionalities, architecture and underlying technologies. Also, various techniques to improve current search practices through search engines were studied, emphasising on directory or category concept, Meta Search Engine Systems and Browse & Search strategy. This enabled understanding the theoretical framework of the research. Similar works for personalisation of web search were also perused giving emphasis on how they enabled personalisation by combining the category and browse & search technology. Also, software agent systems were studied in detail analysing how effective they are in distributed systems especially in web search and personalising web search. Finally, different types of architectures for distributed were studied emphasis on client server architecture and multi-layer (3-tier) architectures of software design. The next chapter will detail the conceptual model for the personalised searching agent.

CHAPTER 3 AN AGENT BASED MODEL FOR THE PERSONALISATION OF WEB SEARCHING

The concept of a personalised search agent is discussed in Chapter 2. In this chapter an agent conceptual model is developed for personalisation of web searching to act between the user and the search engines. A general overview of the conceptual model is depicted in Figure 3.1.

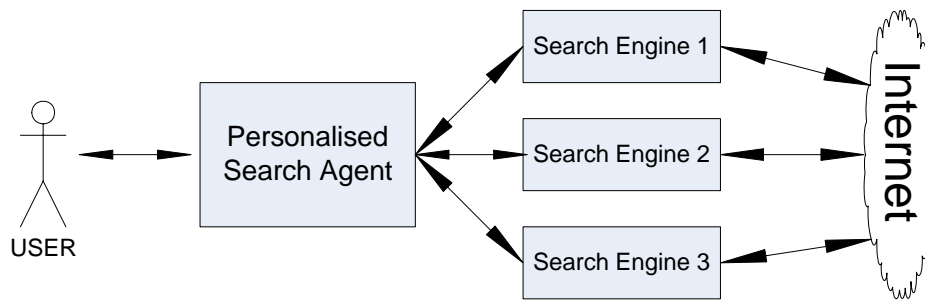


Figure 3.1: General framework for agent based personalised searching

The following techniques are applied to achieve personalisation:

- Using the category or directory approach of organising similar information in a hierarchy
- Using the browse and search approach
- Learning from experience and user feedback
- Using multiple search engines (meta search approach)

In addition, new concepts are also introduced as follows:

- Creating dynamic pages containing the links (URLs) that the user has visited during a search
- Ranking the links based on the time spent on and other parameters
- Ranking the search engines based on the successful links returned to the user

3.1 PSA Architecture

The general architecture of the Personalised Search Agent (PSA) is shown in Figure 3.2.

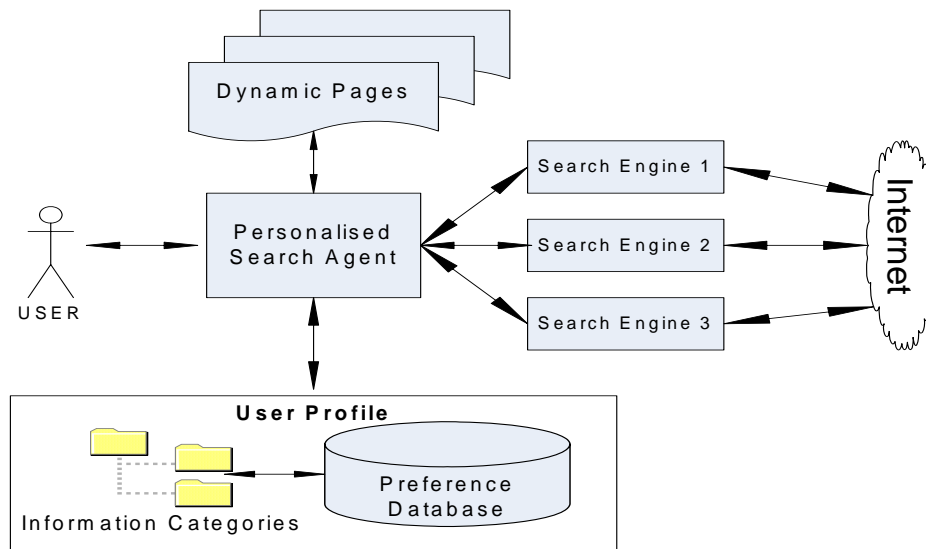


Figure 3.2: A conceptual framework of the PSA

Following is a detailed description of the components depicted in the conceptual framework for the PSA in Figure 3.2 along with the associated techniques.

Information categories

The concept of information categories, introduced in Chapter 2 is a technique used to organise information in a structured hierarchical order. Yahoo directory is a popular example. It would be a difficult task to pre-build all the available category items. Instead it is preferable to start with a minimum set of category items leaving it open for the user to update it later. Initial values of the categories can be gathered by giving a request form to the user.

Once an initial category is ready, whenever the user initialises a search, the search keyword is mapped against an existing category or a new category (categories can be added or updated by the user as needed). As the user keeps searching in this way, the category keeps building, adding new items and values. In the PSA these categories are useful in two ways. Firstly the categories help to enhance the user-entered keywords, by effectively combining the keywords with category values. For example, if the user selected category is *sports* and it has a value *badminton* and if the user entered keyword is *racket* then it is easy to generate a new keyword “*badminton racket*” by simply combining the category item value and the user entered

keyword. If there is more than one value for a selected category (eg. Badminton, tennis and squash) either the user can be prompted for a selection, or the selection can be done automatically by the number of previous searches for each value. If the selection is prompted by the user, it is less user friendly but has a high degree of accuracy. If the selection is decided automatically by the number of pervious searches, it is more user friendly but has a probability risk. For example based on the previous searches made by the user, only the results for “Badminton racket” will be shown. But there is a probability that user wished results for tennis racket this time. The second use of categories in PSA is that categories can be used to remember the user preference rather than asking the user to recall the preference. If the user entered keyword has a previous reference, under one or more categories, then the user is prompted to make a selection before starting the actual search.

Using XML for the Categories

Extensible Markup Language (XML) is a simple, flexible text format to store data. XML was originally designed to meet the challenges of large-scale electronic publishing (Extensible Markup Language, 2002). XML is currently used to improve the functionality of the web by providing more flexible and adaptable information identification. It is called extensible because unlike HTML it does not have a fixed format. Instead, XML is actually a metalanguage that allows individuals to customise using their own tags (UCC Glossary, 2003).

As discussed in Chapter 2, XML is an excellent model for building the data structure for the category or directory concept. With PSA, there are categories and sub-categories arranged as an inverted tree hierarchy as well as values for each category. A simple example is depicted in Figure 3.3.

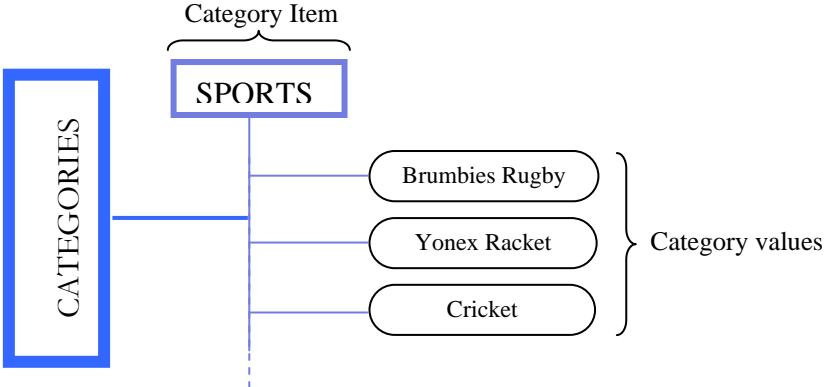


Figure 3.3: A simple illustration of the category concept

XML Schema offers exactly the same structure for data representation. Nevertheless, XML is a widely accepted standard, compatible with most of the platforms and easy to create or edit dynamically using web technologies. Moreover, as indicated in section 2.3.2, there is a detailed description of how XML can be used for creating user profiles and directories. Figure 3.4 is a representation of the category shown in figure 3.3 using XML Schema

```
<xs:element name="Categories">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Sports" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 3.4: A sample XML Schema

An XML representation (XML Schema with data) of the category shown in Figure 3.3 is illustrated in Figure 3.5.

```
...
<Categories>
<Sports> Brumbies Rugby </Sports>
</Categories>

<Categories>
<Sports> Yonex Racket </Sports>
</Categories>

<Categories>
<Sports> Cricket </Sports>
</Categories>
...
```

Figure 3.5: A sample XML representation of category item values

Preference databases

In PSA, preference databases are the actual personalisation database. All the search details are stored in this database. A successful search will always have at least the following details:

- User keyword
- Preferred category value
- Generated keyword
- URLs (links) opened
- Search engines from which each link was retrieved.

Again a successful search is determined by positive user feedback. The mapping of the keywords with the category is done in the preference database and stored with other relevant information.

User profile

The term user profile is more a conceptual or abstract term with regards to PSA. User profile is the effective combination of the categories and the preference database. After a certain period of use the database and the categories become quite uniquely customised to the user. The user profile form is a data entry form designed to capture an initial Local Category structure with a minimal set of category items and values. Though it may not be user-friendly to ask the user to fill in a form, the form could be really useful when the PSA has only very few clients and the global category does not have many items and values.

3.2 A multi layered approach to personalisation

One problem with the PSA conceptual model explained earlier in this chapter is the lack of multi-user support. Introducing client-server architecture, explained in Chapter 2, was found to be a simple and effective solution for the PSA framework. There are two options for incorporating this feature:

- A thick client model that runs in the user machine and an online server part that could also be a web service.

- A client-server model in which both client and server are online web applications / web services.

There are advantages and disadvantages in both of the above approaches. In the first approach the advantages are, since the client is a thick programme running in the user machine, it does not have much limitation for the local storage. This means the size of the database would probably not be a problem. Moreover, even without internet access the user would still be able to review the offline links or previous search history if necessary. The disadvantage of this approach is the accessibility. Since the thick client model runs in the user machine, the cached details would not be available if the user changes machine or the machine breaks down. Furthermore, users have to install the client programme on all the machines they use, making it difficult to synchronise the profile. In the second approach the advantages include good accessibility, since both the client and the server are web services the user can virtually access from anywhere through the Internet. The client-server model also greatly reduces the risk of losing the user profile details due to local system failure. One disadvantage of this approach would be with the size of the user profiles and the database which could be a burden on the web server if the size exceeds certain limits. Another disadvantage with this approach is that the user would not be able to review the history links offline because the Internet is needed to access the client.

After considering the pros and cons of both of the above approaches it was decided to go with the second approach considering that this is experimental research and would not have many users putting an additional burden on the server. Accessibility was also given higher priority. The client server architecture not only solves the multi-user issue, but also gives great enhancements to the personalisation itself. The multi-layer approach consists of two exactly similar units, one on the client side and other on the server side. On the client side the preference database and the dynamic pages would be specific to a particular user whereas on the server side they are more general. This is a generalisation specialisation approach. The server side profile database and the categories are the union of all the clients (users). Each successful search that a client makes therefore will have an impact not only in updating the local details but also in affecting the server details. This results in the server acting as a personalised information source or a global database of personalised structured information after a period of use. All the clients request the information from the server; the server then checks its global profile for relevant information and in the absence of any, conducts a real search. Once the server has enough links in its global profile, the search request from a client

can be returned almost instantaneously. A client-server based multi-layer architecture of the PSA is illustrated in Figure 3.6.

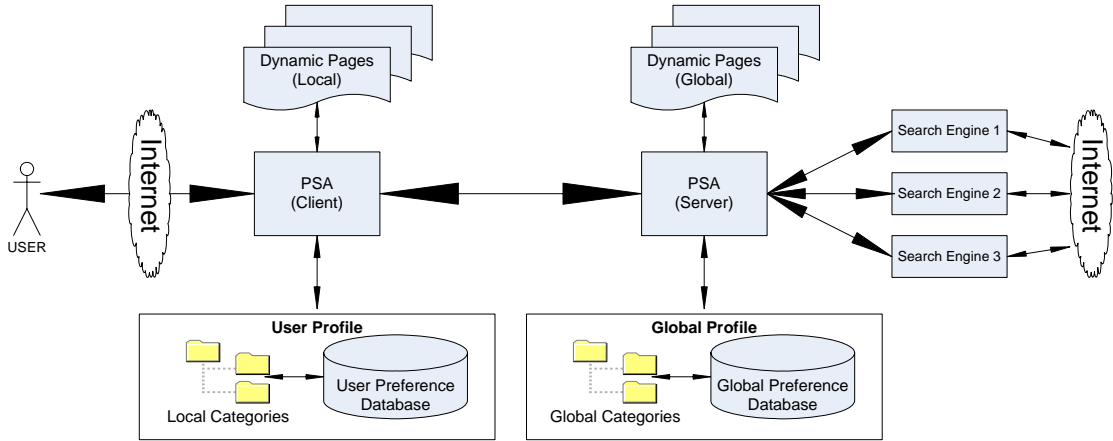


Figure 3.6: Multi-layer architecture of the PSA

This architecture enables a two level personalisation. Figure 3.7 illustrates how the level of personalisation varies from the Internet to the user through the PSA. This shows how the PSA reduces the amount of information by filtering out irrelevant information at the server and client level, giving a smaller number of highly personalised results to the user.

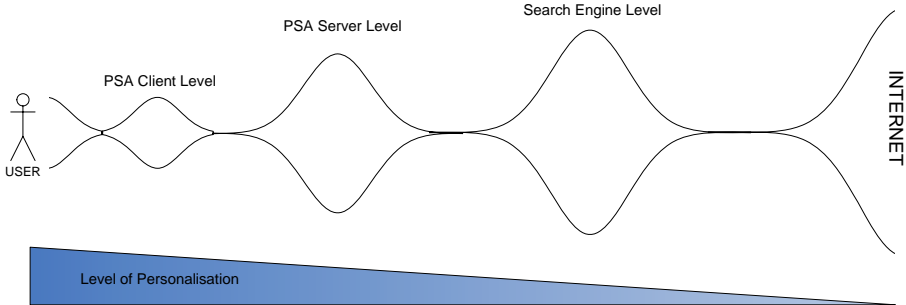


Figure 3.7: Personalisation graph for multi-layer PSA architecture

Global categories

Structurally, the global categories and the local categories are the same. They differ, however, in the level of profiling and in the number and variety of their content. Global categories are maintained in the server side of the PSA. The global category is a collection of all the client

items. With a considerable number of clients for the PSA, the global category becomes enriched so that new clients can actually import a copy of the global category and customise it for their needs. For Example, if client A and client B are the only two clients for the PSA, then the category structures are as illustrated in Table 3.1.

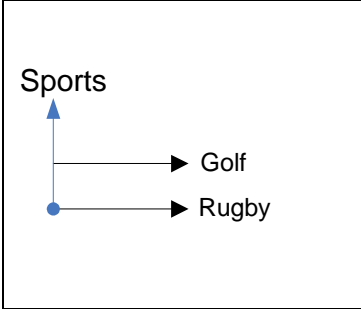
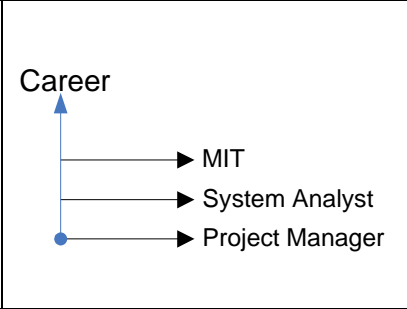
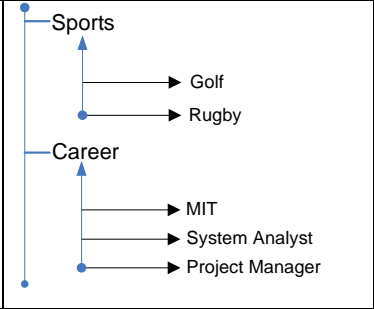
Client A (Local Category)	Client B (Local Category)	Server (Global Category)
		

Table 3.1: Representation of local and global categories

Hence, if new clients join the PSA, they can build their initial local category either by filling in a user preference form or by importing the global category from the server and then customising it if necessary. The former method is good when the PSA has only a few clients (in the beginning) or it has very limited entries in the global category, whereas the latter method is recommended when there is a larger number of clients in the PSA and the global category has grown substantially. The PSA server takes care of the global category, avoiding any potential redundancy of either the category items (eg: ‘sports’) or the category values within an item (eg: ‘Rugby’).

Global preference database

The global preference database is a union of all the local preference databases of all the clients in the PSA and is maintained by the PSA Server. It has largely the same structure as of a local preference database except that it has additional data fields to capture information about the clients and general trends. The global preference database is potentially a great resource for the PSA clients as it keeps record of the previous search history of all the clients in the PSA. With a good number of clients and substantial search history, the global preference database would contain almost every keyword combination a user might use, allowing the PSA to return the results internally without going for an actual web search. A sample database structure for the global preference database is illustrated in Table 3.2.

Keyword	KeyGen	Category	ClientID	Dynamic Page
---------	--------	----------	----------	--------------	-------

Table 3.2: A sample representation of the global preference database

Global profile

Like the user profile, the global profile is an effective combination of the global preference database and the global categories and is rather an abstract term to represent the ‘generalised personalisation’ of all the clients in the PSA. The global profile can be useful in predicting general trends and in determining search engine efficiency with respect to the category and the types of user.

Dynamic pages

Dynamic pages are the result pages created by the PSA based on the cached relevant results from the database. On the client side they are real webpages whereas, on the server side they are a package of results generated as per the client request and passed on to the client after creation. The client side dynamic pages can be a well-structured report having a small screen shot of each URL, the related category and other retrieval details.

3.3 PSA Functionalities

This section describes the major functionalities of the PSA starting from accepting inputs from the user to giving personalised results back to the user. The informed search throws light on the importance of the ‘browse and search’ approach explained in section 2.2. Every search using the PSA is done in relation to a category item. That is, either the PSA or the user has to select a related category before accepting the search keyword/s. This indicates in which area of information the search needs to be done. The browse and search approach is well known for its highly relevant output.

Generating keywords

The idea behind keyword generation is quite simple. During conventional information search practice, we normally shuffle the keywords by changing the order or replacing one keyword with another one to achieve the desired result. Obviously, this takes user time and effort. The

keyword generation feature of the PSA automates this process. Basically, the PSA accepts keyword/s from the user along with a category name. The PSA then makes different combinations of 'keywords' and 'category values' based on a keyword-generating algorithm. The actual search (with the search engines) is done using all the 'keyword sets' generated and presented to the user in a predefined order until the user gives positive feedback. It can save a considerable amount of user time and effort.

Sensing and decision making

Sensing and decision making happens on both the client and the server sides of the PSA although both are more user dependent on the client side and more user independent on the server side. This user independency makes the server act like an agent.

Sensing on the client side includes matching the user-entered keyword with the entries in the user preference database. Decisions are then made based on the matching results. If a match is found then the PSA searches for the related dynamic page. If the match is found in more than one category (like the word *racket* in both *sports* and *narcotics*) then the PSA presents the results under most accessed category and also lists the categories in the order of last access, and prompts the user to select a category. A detailed analysis of the client side decision-making is explained later in this chapter under section four.

As stated earlier the server is more independent of user action in sensing & decision-making. The PSA server continuously watches its environment sensing the client requests and making decisions based on the request parameters. For example, if the client requests a set of keywords the PSA searches its global preference database for a similar entry and retrieves the URLs based on the *search engine ranking* for that client. For example, if the *search engine ranking* of that client shows Google has been most successful in retrieving information for that client, the URLs originally gathered from Google are given higher priority while retrieving the URLs from the *global dynamic page*. Apart from this, the server continuously organises the keyword by mapping it against the global category items. New category items are added to the global category as the clients make changes in their local category structure. The PSA also senses any duplication in both URLs and keyword patterns and makes sure that redundancy is avoided to the maximum possible level.

Generating Dynamic pages from a successful search

Dynamic pages have been well explained in section 3.1.4 of this chapter. Basically, there are dynamic pages maintained by both the server and the client, named global and local dynamic pages respectively. A local dynamic page is created from a successful search on a new keyword, or an existing keyword with a new category value. Once the dynamic page is created, all the successful URLs associated with that search are stored in that page. A successful URL is determined by measuring the active time the user spends on that page (URL). Again *active time* means the time taken by the user while keeping that page in focus (for example, selecting text, clicking, copying, or simply while the window is active). However, if the URL window is opened but is out of user-focus or is minimised, that time will not be calculated as active time. If the active time crosses the threshold, the link is given a rank and added to the dynamic page against both the keyword and the category. All the links that score one or more value will find a seat in the dynamic page. The URLs are sorted based on rank (although the default sorting is done using the LIFO algorithm). The URLs in the dynamic page are again sorted using *Search Engine Ranking*. The sorting precedence would be as follows:

- Sort Level 1 : Link Ranking
- Sort Level 2 : Search Engine Ranking
- Sort Level 3 : LIFO (Last in first out)

If there is more than one category value associated with a single keyword (not confusing the category value with category item; an example of same category different values is, *badminton* and *basketball* under *Sports*), the PSA still maintains two different dynamic pages. But the PSA also understands that these dynamic pages are related to one another. So, if the search results of a category value receive negative feedback from the user, the PSA respond with links from dynamic pages of other category values in the same category item (again the default precedence rule is LIFO).

For example, let the category values for the category item *Sports* be *Badminton* and *Tennis*. The user-entered keyword is *racket* which has two entries in the *user preference database*, one against category value tennis and the other against category value badminton. The PSA senses the LIFO values and decides to choose tennis as it is the latest addition to the database.

If the result receives negative feedback from the user then the PSA retrieves the URLs from the dynamic page for badminton (the next category value).

After each successful search the client returns the *Local Dynamic Page* to the server along with the user preference database values and the server then matches the entries against its database. If no match is found the server then saves the URLs in the Local Dynamic Page to the *Global Dynamic Page*. If the server finds duplications in the URL, then the existing URL is given a higher *Global Link Rank*. The non-duplicated entries are added and database entries are updated accordingly. Links (URLs) in the global dynamic page are dropped if they fail to maintain the *Life Saving Rank*. Life Saving Rank will be explained in detail later in this chapter.

Managing the Preference Database

Preference databases which are the most important components of PSA conceptual model have already been explained earlier in this chapter. They contain the base data for all the ranking calculations both on the client and on the server side. They normally contain all the possible attributes of a successful search such as the keyword pattern, the category item, the category value and the search engine ranking for that search, as well as the name and location of the associated dynamic page. Almost all the non-derivable information of a successful search are stored in the preference database, some of which is not used currently by the proposed PSA conceptual model, but is potentially useful later as more and more data-mining and personalisation ideas are incorporated with the PSA.

3.4 Rankings and Algorithms

In order to determine the relevant information from the captured user details four ranking approaches are used. The algorithms of these approaches are described below.

The Keyword Generating Algorithm

It is an effort to simply assist the user in generating different keyword combinations based on a simple algorithm explained below. By doing this the PSA not only eases user effort but also substantially saves user time. If 'A' be the category of preference, B_1 to B_n be the words in the user keyword combination then the sequence would be:

“A + (B₁...B_n)”

A + “(B₁...B_n)”

A + B₁ + “(B₂...B_n)”,

Ideally, this sequence will stop either when the user gives positive feedback or when all the probable combinations of words are exhausted. However, in actual practice, proper limits are applied in the number of probable combinations.

The Link Ranking Algorithm

Link ranking is a new approach to find personalisation at its grass roots level. Basically, link ranking works by calculating the active time the user spends on each URL in a successful search result. A successful result is defined by user feedback. User actions will be continuously monitored once the results are submitted but only considered for processing, if the user gives positive feedback on the result. The link ranking algorithm can simply be explained as follows:

Let ‘t’ be the number of seconds spent by the user on a link (or URL), ‘s’ be the search engine ranking for that user and ‘n’ be the number of times the link (or URL) is viewed. Then

$$\mathbf{Link\ Rank = (t + s + n)}$$

Note:- The links or URLs are organised in the cached page as per the descending order of the link ranking algorithm.

The Search Engine Ranking

The search engine ranking is given to each dynamic page both on the client side and on the server side. Ranking is calculated by grouping all the URLs in the page by the search engines (which returned the link) and then taking the sum of the link rankings of the URLs in each group.

If two or more search engine have the same ranking, then their relative ranking is recalculated by counting the number of URLs returned by each, and furthermore with the LIFO algorithm.

The Life Saving Rank

Let the link ranking for the URL be LR and T be the total number of seconds elapsed after storing that URL to the dynamic page, then

The life saving rank is = T/LR

If the life saving rank value (T/LR) is above 100, the link will be dropped from the dynamic page. The life saving ranking is applied in both local and global dynamic pages and is an essential measure to make sure that only relevant links are in the dynamic pages and a minimum amount of space is used from the server storage resources.

3.5 A Sample Scenario Walkthrough

This section describes how the PSA works with the help of a scenario, which demonstrates the functionalities explained above. The detailed sequence diagram shown in Figure 3.8 below is a graphical representation of all three scenarios explained thereafter.

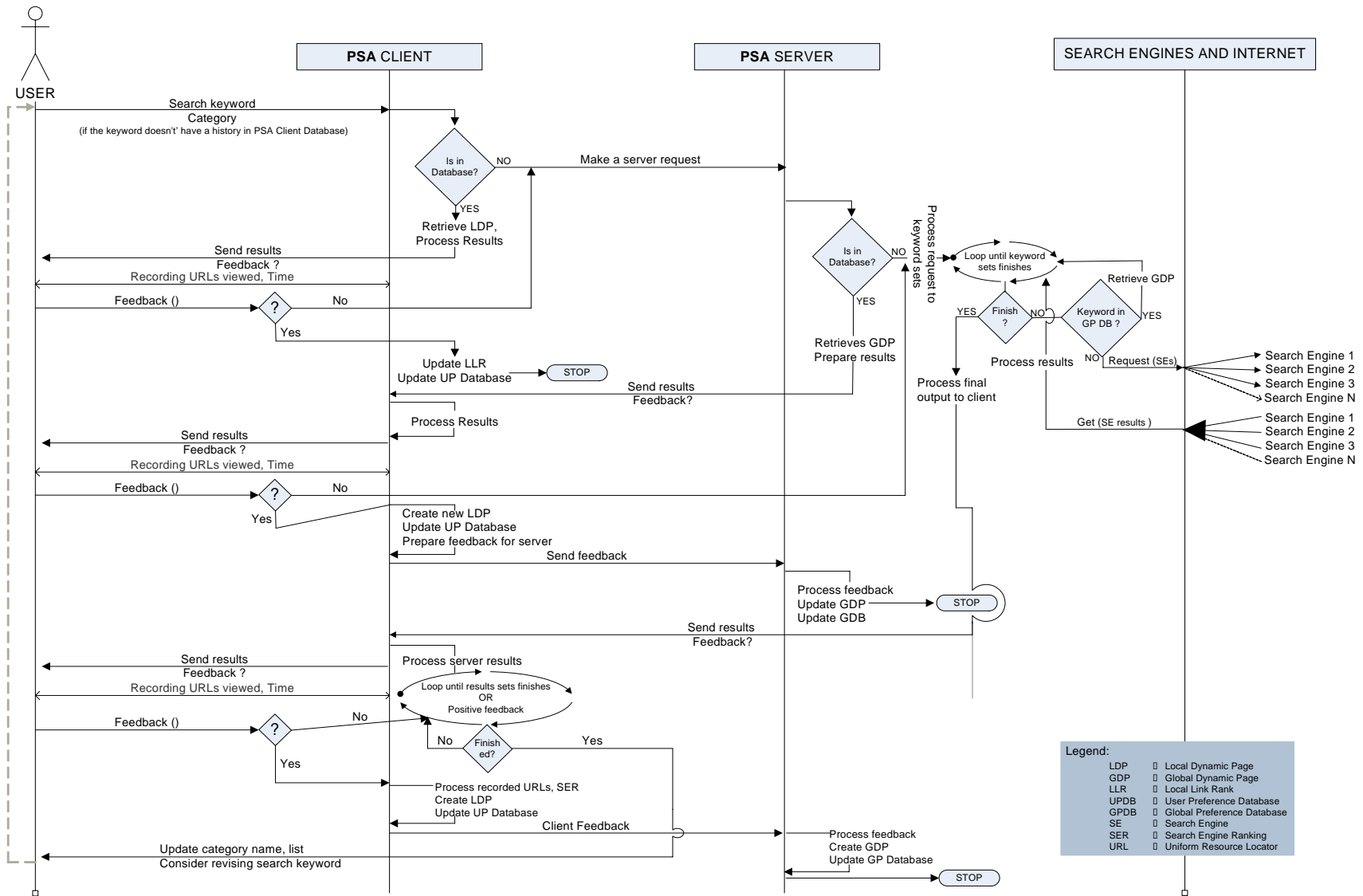


Figure 3.8: Sequence diagram depicting PSA functionalities

Scenario1: *The user performs a new search, which has no history either on the client side or on the server side of the PSA.*

The sequence of steps that the user would take and the related actions performed by the PSA are detailed as steps in the order of occurrence:

- a. The user inputs a search string to the PSA client
- b. The PSA client searches its user preference database for an entry for that keyword.
- c. If not found then the client asks the user to select a relevant category from the Local Categories.
- d. Once the user selects the Local Category item for the search string, the PSA client generates a set of search keywords by the different combinations of category value and words in the search string based on the search keyword generating algorithm (explained in this chapter in section 3.4.1).
- e. Makes a request to the server. The request basically contains the category, the original search string submitted by the user and all the keyword sets generated.
- f. The PSA server senses the client request and searches its Global Preference Database for an exact entry for the original search string under the given category.
- g. If the server finds no matches, the server then extracts the keyword sets from the client requests and again matches each keyword set with its global preference database. If a match is found then the related global dynamic page is fetched and the contents are read to a temporary space.
- h. For all those keyword sets, which have no match in the global preference database, a meta-search engine search will be chosen. The results from all the search engines are sorted based on the search engine ranking for the particular category and then saved in a temporary space.
- i. Once all the keyword sets are finished, the server then generates a results page (with a pre-defined format) for the client and sends it over to the client
- j. The client receives the results from the server, processes it and extracts the results for the first keyword set and presents them to the user.
- k. While the user is viewing the results, client continuously monitors the URLs opened by the user along with other activities and the time spent on each link.
- l. The results are presented with a feedback request. If the user gives positive feedback, the client generates a local dynamic page, stores the URLs opened by the user during the monitoring time and sorts the URLs (or Links) based on a Link Ranking Algorithm

(explained in this chapter under section 3.4.2). Also, the user preference database is updated (which includes mapping the user entered keyword with the category).

- m. On receiving negative feedback from the user, the PSA client fetches the results for the next keyword and presents them to the user along with the feedback request. On continuous negative feedback from the user, the PSA client repeats the process until the entire search results from the server are finished.
- n. If the user gives negative feedback for all the search results displayed, then the PSA client suggests the user revise the keyword entered or edit the category item/name.

Scenario 2: The user performs a search that has no history on the client side but has history on the server side

This scenario is another pathway within scenario 1. The sequences are explained below.

- a. The steps 'a' to 'e' are repeated from scenario 1.
- b. The server finds a match (obviously one or more other clients have already made the same search) and retrieves the relevant global dynamic page.
- c. The server prepares a result for the client using the URLs from the page and sends it over to client.
- d. The client receives the result, processes it and displays it to the user with a feedback request.
- e. While the user is viewing the results the client continuously monitors the URLs opened by the user along with other activities and the time spent on each link.
- f. The results are presented with a feedback request. If the user gives positive feedback, the client generates a local dynamic page, stores the URLs opened by the user during the monitoring time and sorts the URLs (or Links) based on a link ranking algorithm (explained in this chapter under section 3.4.2). Also the user preference database is updated (which includes mapping the user entered keyword with the category)
- g. If the feedback is negative, the PSA Client goes through scenario 1.

Scenario 3: The user performs a search that has client side history

This scenario is another pathway within scenario 1. The sequences are explained below.

- a. The user inputs a search string to the PSA client
- b. The PSA client searches its user preference database for an entry for that keyword.
- c. The client finds a similar entry (in this case) and retrieves the relevant local dynamic page and presents it to the user with a feedback request.
- d. While the user is viewing the results, the client continuously monitors the URLs opened by the user along with other activities and the time spent on each link.
- e. On receiving positive feedback from the user, the PSA client updates the URL order in the local dynamic page based on the change in link ranking and updates the user preference database.
- f. On receiving negative feedback from the user the PSA client follows the steps explained scenario 2 above.

3.6 Summary

This chapter has described the conceptual model of a personalised search agent, as a solution to the problems identified in Chapter 1, filling in the gaps in the current theoretical framework of personalised information searching. The conceptual model of the PSA is well illustrated with diagrams showing the PSA architecture. The components in the conceptual model along with the different algorithms and calculations used have also been explained in detail. The chapter ends with a scenario walkthrough, which reveals the functionalities of the PSA and illustrates the information flow and the process routines of the PSA conceptual model. The next chapter will illustrate the implementation of the PSA conceptual model, through the design and development of a prototype for the PSA.

CHAPTER 4 PSA PROTOTYPE DESIGN AND DEVELOPMENT

To test the architecture developed for the PSA in Chapter 3, a prototype is necessary. It is also a research requirement in order to achieve internal validity. The analysis and design was done considering all the functionalities and features originally developed in the conceptual model. However, not all of them were considered because they are out of scope of this research project and only a set of core features was considered for implementation.

4.1 Design and development standards for the PSA

This section reviews software design and development standards which were used in the development of the PSA prototype.

4.1.1 Object-oriented software design

Object-oriented software design (OOSD) is concerned with developing an object-oriented model of a software system to implement identified requirements. Many OOSD methods have been described since the late 1980s (Baudoin & Hollowell, 1996). OOSD can yield the following benefits: maintainability through simplified mapping to the problem domain, which provides for less analysis effort, less complexity in system design, and easier verification by the user; reusability of the design artefacts, which saves time and costs; and productivity gains through direct mapping to features of Object-Oriented Programming Languages (Baudoin & Hollowell, 1996). The maximum impact from OOSD is achieved when used with the goal of designing reusable software systems (Maring, 1996). For implementation of the conceptual model of the PSA to a prototype, OOSD was chosen because of its clear advantage over other conventional software development methods.

4.1.2 Unified Modelling Language (UML)

Modelling is the designing of software applications before coding. Modelling is an Essential Part of large software projects, and helpful to medium and even small projects as well. A model plays the analogous role in software development that blueprints and other plans (site maps, elevations, physical models) play in the building of a skyscraper. Using a model can assure that required functionality is complete and correct, end-user needs are met, and program design supports requirements for scalability, robustness, security, extendibility, and other characteristics, *before* implementation in code (UML Resource Page, 2005). UML is a natural selection, when the development method chosen is OOSD. UML is used for software

analysis and design while converting the PSA conceptual model from Chapter 3 into a software artefact.

4.1.3 Implementation tools

The following tools and technologies were used for PSA prototype implementation, following the software analyses and design. Microsoft technologies were used because of their flexibility and resource availability.

.NET framework

.NET is the Microsoft Web services strategy to connect information, people, systems, and devices through software. Integrated across the Microsoft platform, .NET technology provides the ability to quickly build, deploy, manage and use connected, security-enhanced solutions with Web services (What is .NET, 2005). ASP.NET is a technology for creating dynamic Web applications. It is part of the .NET Framework; providing better performance than with scripting languages. Web Forms allow the building of powerful forms-based Web pages. When building these pages, it is possible to use ASP.NET server controls to create common user interface elements, and program them for common tasks. These controls allow the rapid building of a Web Form out of reusable built-in or custom components, simplifying the code of a page (Getting Started with ASP.NET, 2005). ASP.NET was chosen for the PSA considering these qualities of it, which helped the implementation of the PSA functionalities easy and quick within the timeframe.

C # ("C sharp"), a modern, object-oriented programming language built from the scratch for exploiting the power of XML-based Web services on the .NET platform. Based on modern, object-oriented principles, C # has been crafted to help developers accomplish more with fewer lines of code, and with fewer opportunities for error (Microsoft Introduces..., 2005). With its elegant object-oriented design, C # will be a great choice for developing the PSA and building its various functionalities and components.

4.2 Requirement analysis using UML

A detailed analysis has already been done of the concept and the ideas which emerged on personalisation of web search practice, which eventually resulted in the development of a conceptual model in Chapter 3. However all the work done at that stage, was done from a

theoretical perspective, and hence was named the conceptual model for the PSA. But building a prototype demands a new analysis and design of the problem and the solution from a software development perspective. Object-oriented software development was chosen for this purpose, because of its clear advantages over the conventional systems and many other features, including reusability, maintainability and upgradeability (Khawar & Cary, 2002) which are quite relevant in this context as the implementation will be done with only limited features of the theoretical schema and apparent potential refinements.

As it was decided to follow object-oriented software analysis and design, UML was used as a standard tool to employ re-analysis and re-design of the PSA conceptual model from the perspective of software development. Since the PSA conceptual model is based on client-server architecture, to avoid complexity during the requirements analysis for the PSA prototype, the client and server sides are separated and hence the use cases are identified separately for both sides.

4.2.1 Use cases and system sequence diagrams for the PSA

Use cases and system sequence diagrams for each are given for a better understanding of how the PSA works. Use cases and sequence diagrams are explained separately for client and server side events.

Client side use cases: - (Actors: User, Client)

- a. The user requests a search using a new keyword (i.e. which has no reference in the PSA database)
- b. The user requests a search using a keyword that has a reference in the PSA database
- c. The user browses the search results for desired information.
- d. The Client updates its category table

Sequence diagrams for the PSA client

Use Case: - User requests a search using a new keyword (i.e. a keyword which has no reference in the PSA database)

- The user gives the search keyword
- The system checks the database and finds no match
- The system asks the user to select a category and category values (or enter a new category value)

- The system shuffles keywords with category values and generates one or more keyword sets.
- The system passes the keyword set to the server and gets results from the server.
- The system shows the results for the first keyword set to the user and asks for feedback.
- The system reads user feedback and if positive then displays results for the next set of keywords.

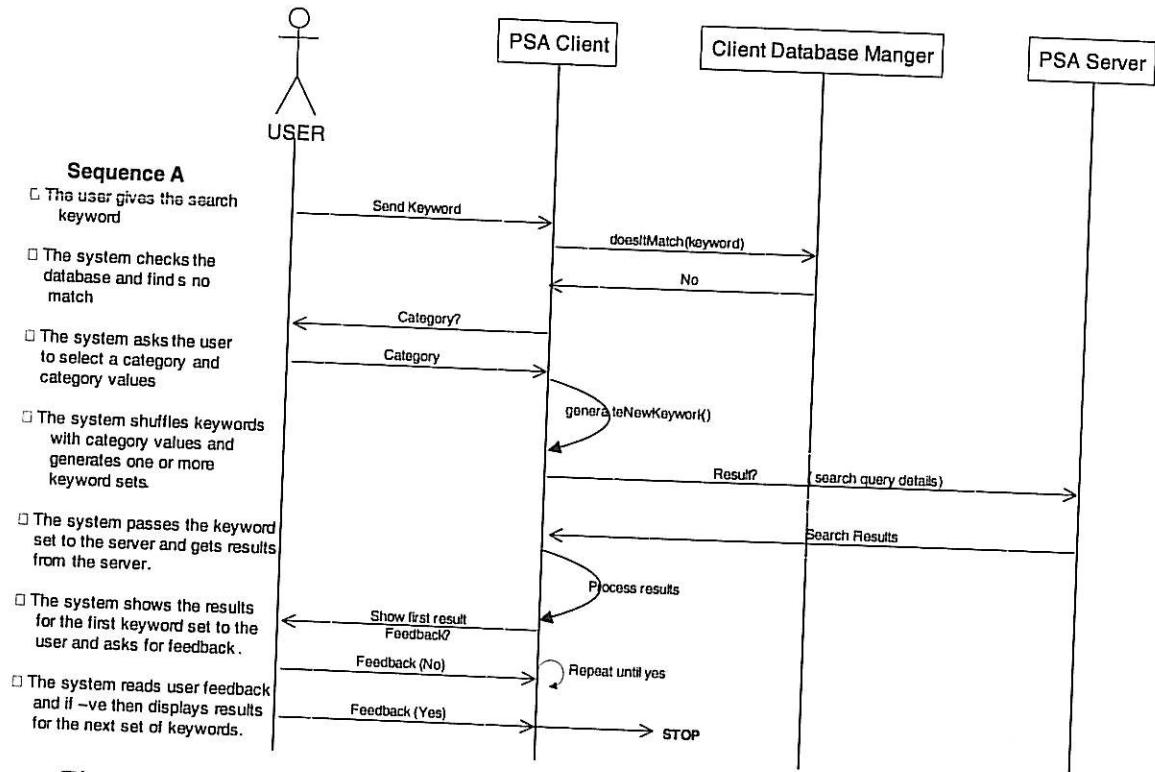


Figure 4.1: System sequence diagram: User requests a search using a new keyword

Use Case: - the user requests a search using a keyword that has a reference in the PSA database

- The user gives the search keyword
- The system checks for a match in the PSA Client database
- The system finds a match and retrieves the URLs, sorts them as per the link ranking and displays them to the user

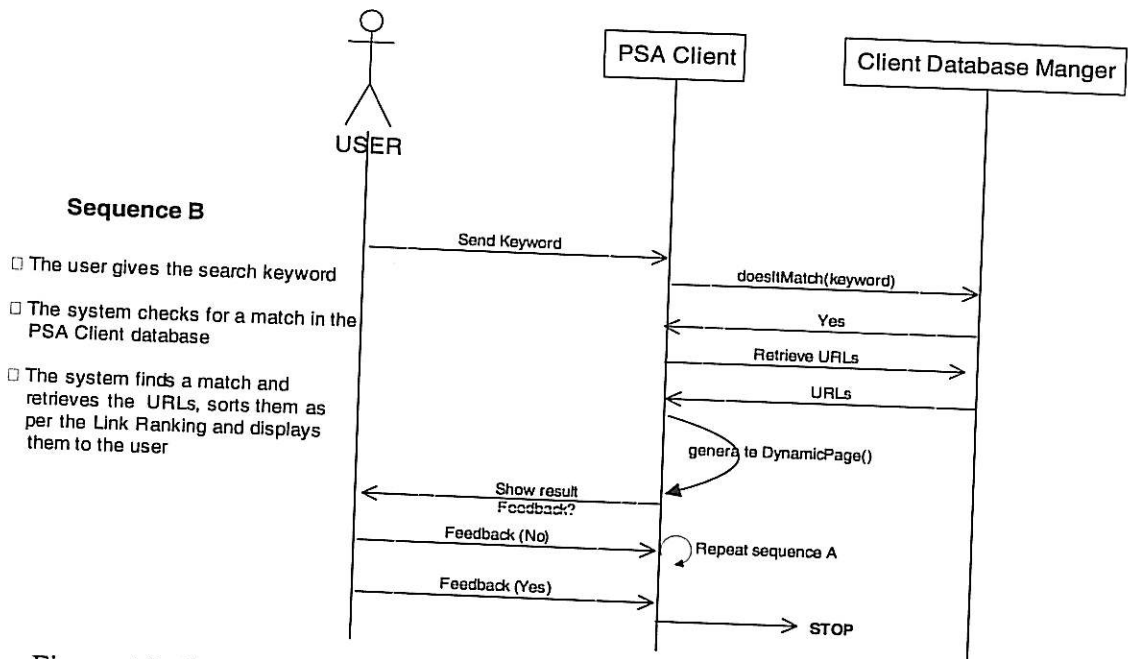


Figure 4.2: System sequence diagram: PSA client side search using a keyword that has a reference in the PSA database

Use Case: - The user browses the search results for desired information

- The user clicks a link from the results
- The link opens and active time is started
- The system asks if the link was useful
- On positive user feedback the link is stored in the database against other parameters (such as the link rank)
- If the user opens the same link again, only its link ranking is updated.

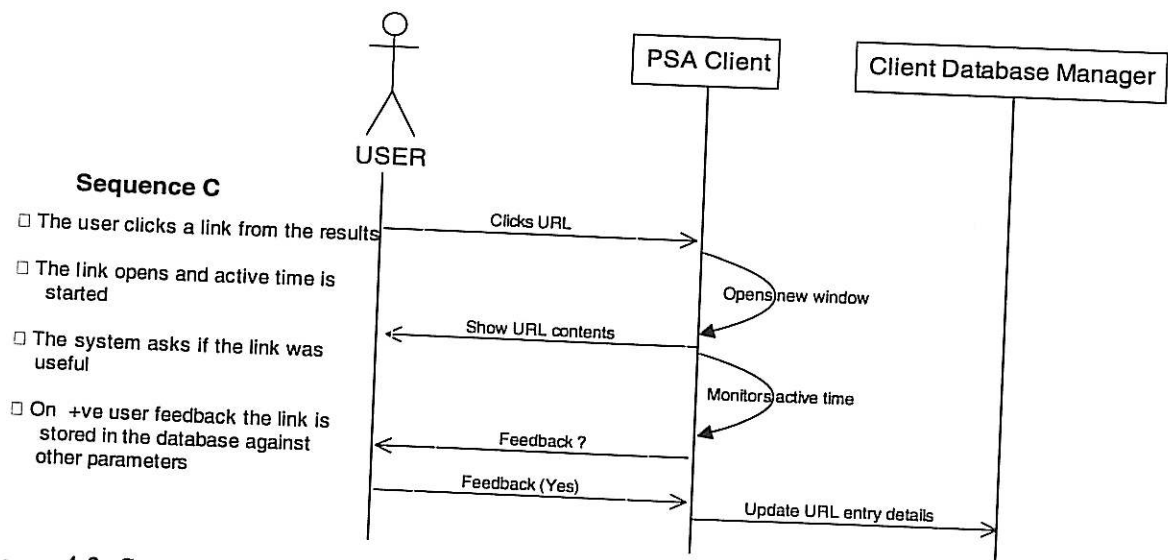


Figure 4.3: System sequence diagram: The users browses the search results for desired information in the PSA client database

Use Case: - The client updates its category table

- The user adds a new category item /category value or edits an existing category name / category value
- The system updates its category database (XML)

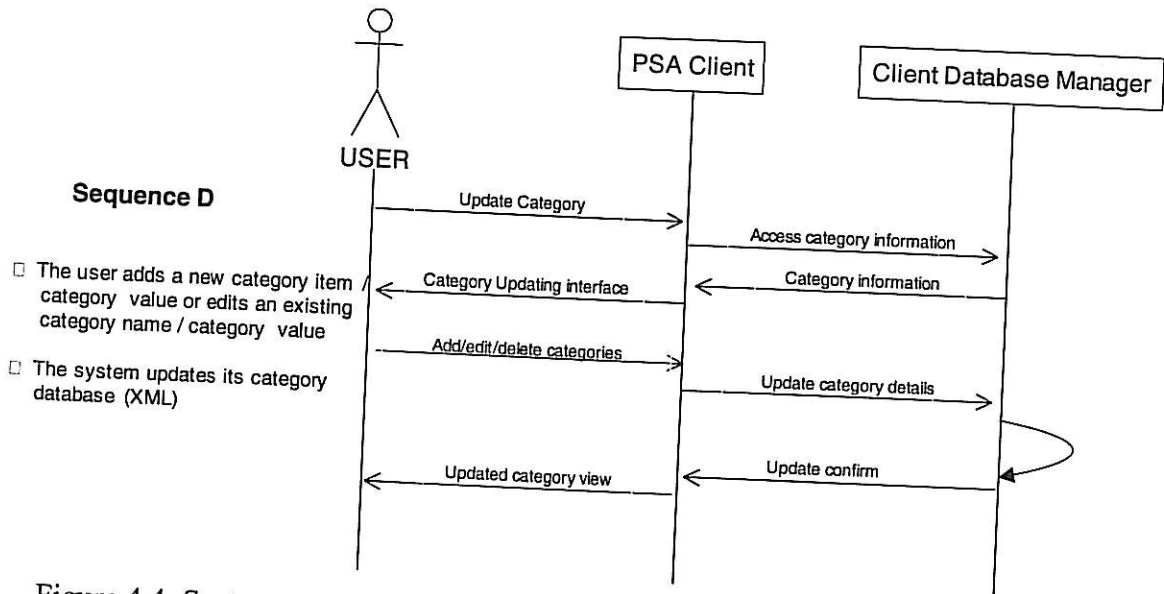


Figure 4.4: System sequence diagram: The PSA client updates its category table

Server side use cases: - (Actors: Client, Server)

- a. The client requests a search that has no reference in the server database.
- b. The client requests a search that has a reference in the server database.
- c. The server updates its category table

Sequence diagrams for the PSA server

Use Case: - Client requests a search that has no reference in the server database

- The server receives a search request from the client
- The server searches its database and finds no related record
- The server extracts keyword sets from the client requests
- The server initiates a search for each keyword from all available search engines
- The server receives the results for each keyword set from the search engines and combines them following the search engine ranking for that user.
- The server returns the results to the client

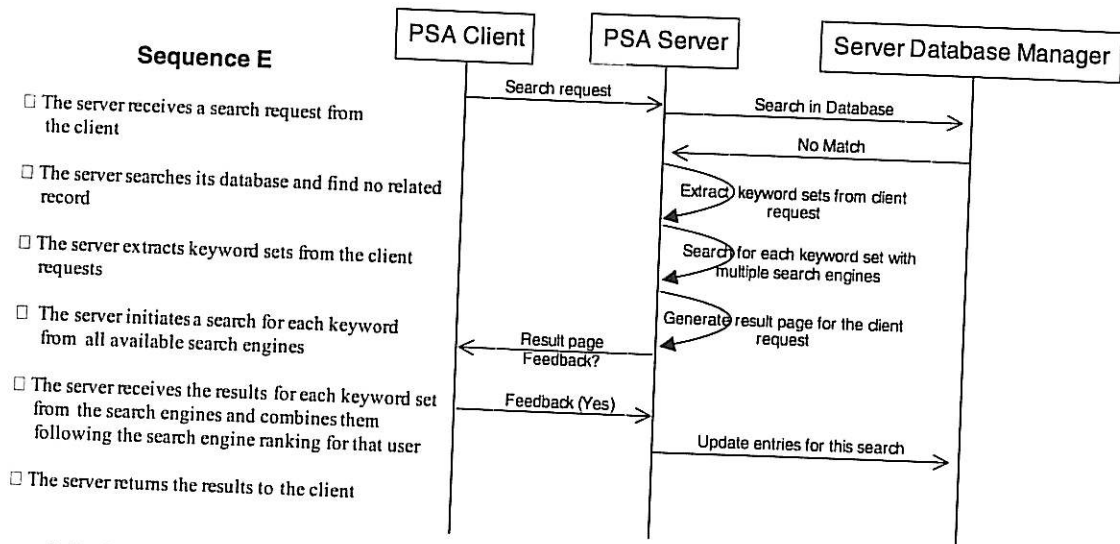


Figure 4.5: System sequence diagram: The Client requests a search that has no reference in the server database

Use Case: - The client requests a search that has a reference in the server database.

- The server receives a search request from the client
- The server searches its database and finds the related record
- The server retrieves the URLs based on the keyword, category and category values followed by creating a dynamic page with all those URLs sorted in order of their Link Rank.
- The server sends the dynamic page to the client as the results of the search query.

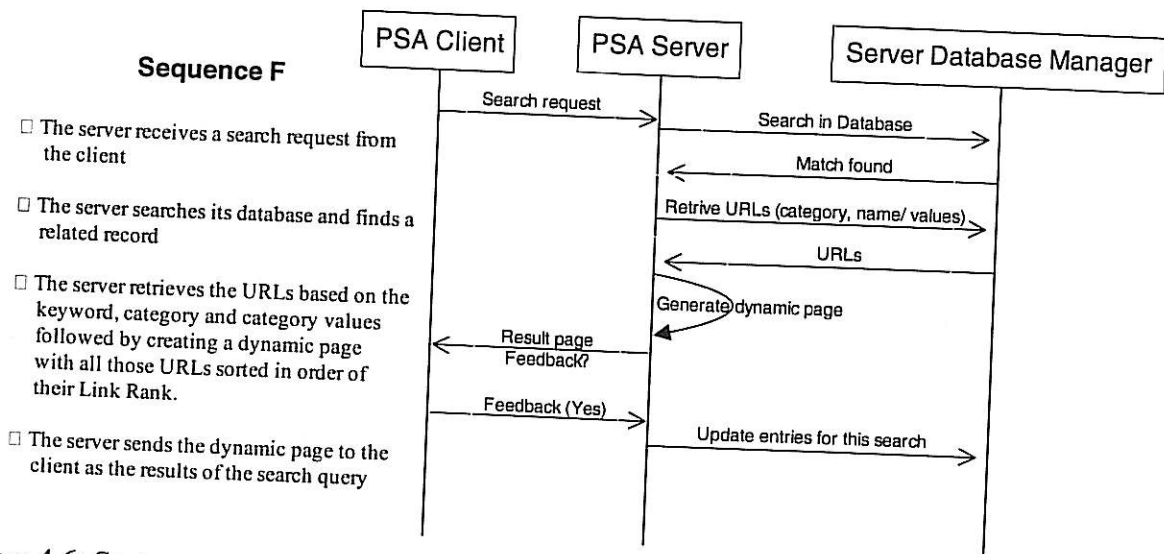


Figure 4.6: System sequence diagram: The client requests a search that has a reference in the server database.

Use Case: - The server updates its category table

- The client reports a category table update (New category item, category item name change, a new value added or an existing value name change)
- The server updates its category table (XML) by adding new entries and updates (please note, the server only adds. It does not change a category name or value)

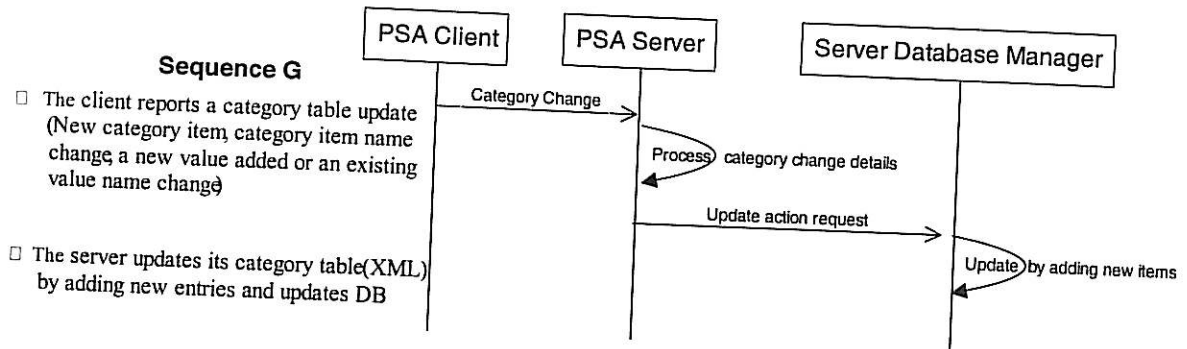


Figure 4.7: System sequence diagram: The server updates its category table

4.3 The Prototype Design for the PSA

The prototype design for the PSA based on the requirement analysis above was done by developing a collaboration diagram or class diagram for the whole architecture, that is covering both the client and the server components of the PSA. Also, Entity Relation diagrams (ER Diagrams) were developed as part of the database design for user and the global preference database, and for the dynamic pages. Finally, a few interface designs (GUI) were developed, based on the user interaction analysis. Moreover, an XML schema was also developed to represent the categories (both local and global).

4.3.1 The class diagram

The class diagram depicting all the classes developed for the client server architecture of the PSA is illustrated in Figure 4.8 below.

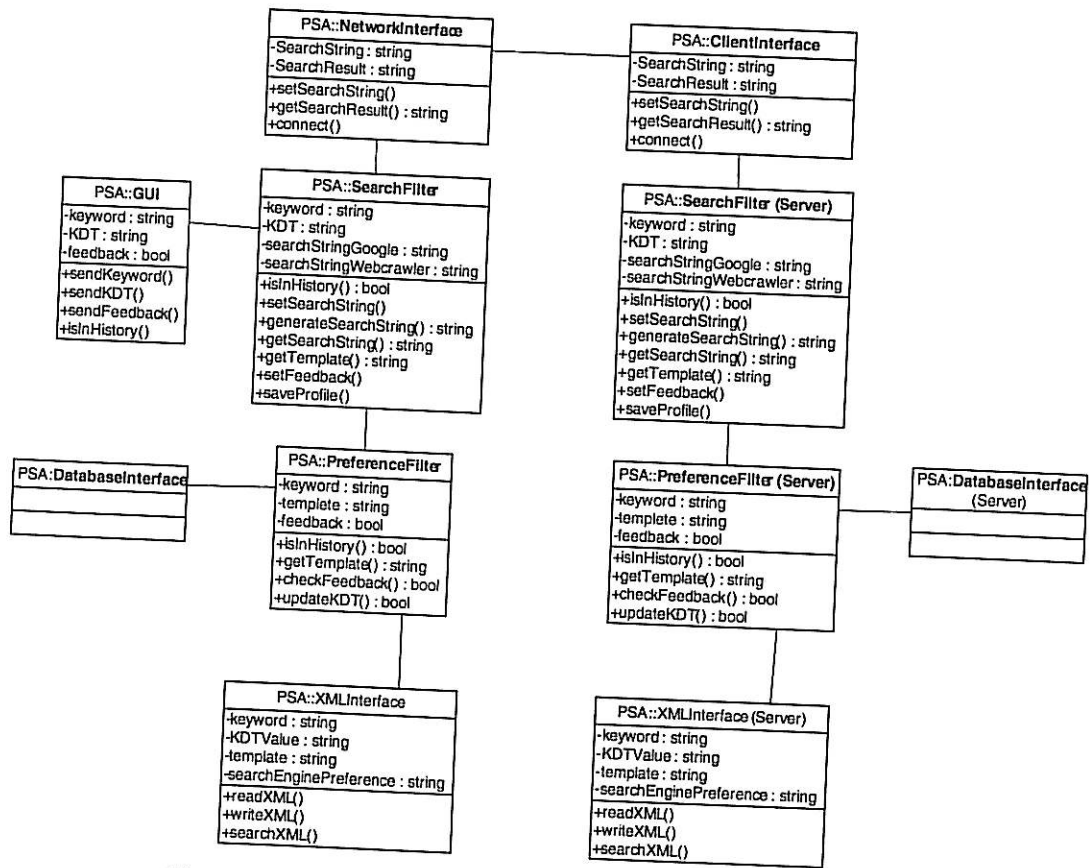


Figure 4.8: Class diagram for the whole PSA architecture

4.3.2 ER diagrams

The entity relation diagrams were developed to address the database requirements of the PSA. During the requirement analysis six major data storage issues were identified, namely, local categories, global categories, user preference databases, global preference databases, local dynamic pages and global dynamic pages. Later, however, it was decided to implement the local and global categories using XML rather than using a database hence the local and global categories were excluded from the ER Diagrams represented in figures 4.9 and 4.10.

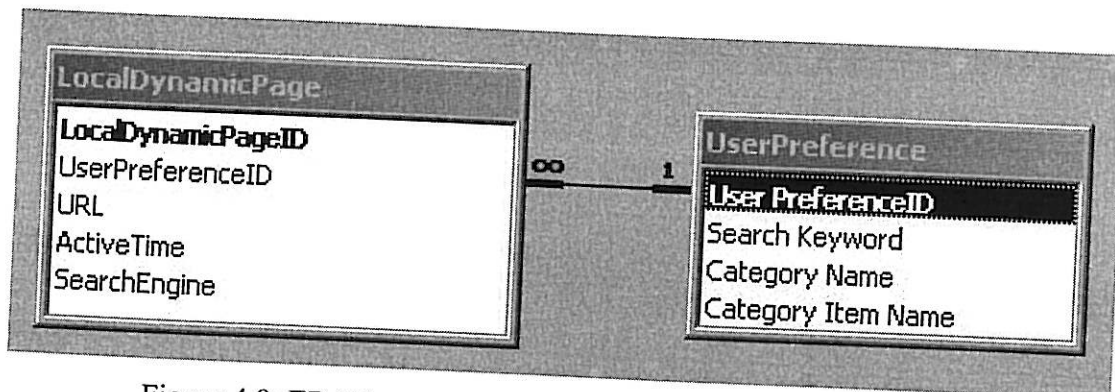


Figure 4.9: ER-Diagram for the databases on the PSA Client side

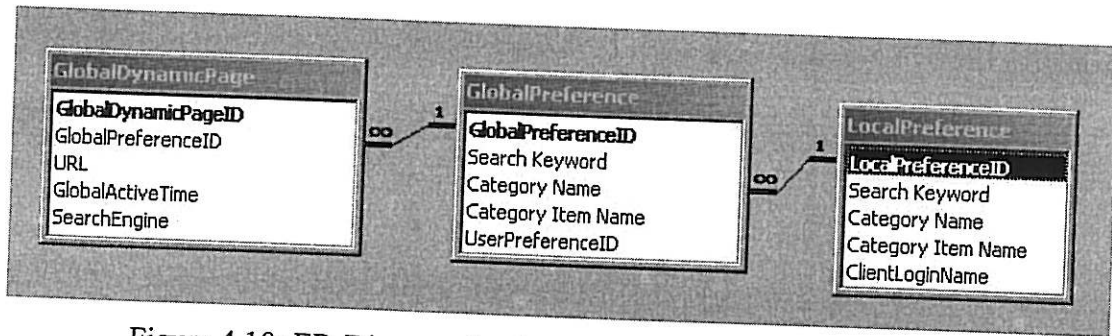


Figure 4.10: ER-Diagram for the databases on the PSA Server side

4.3.3 Interface design

Following are the interface designs to suit the use cases illustrated below. Care was taken to follow it on actual implementation. However, for technical reasons it may not be exactly similar. Figure 4.11 shows the first GUI, that is the PSA search interface page which simply accepts the keyword from the user.

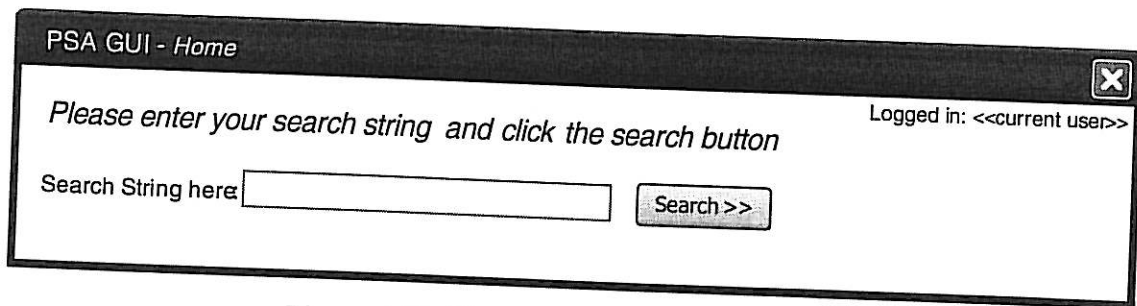


Figure 4.11: The PSA search interface page.

The PSA then searches its local and global databases for a similar entry, as explained in the scenarios under section 1 of this chapter. If the PSA finds a match in its database for a related search it then extracts the URLs and creates a dynamic page as shown in Figure 4.12.

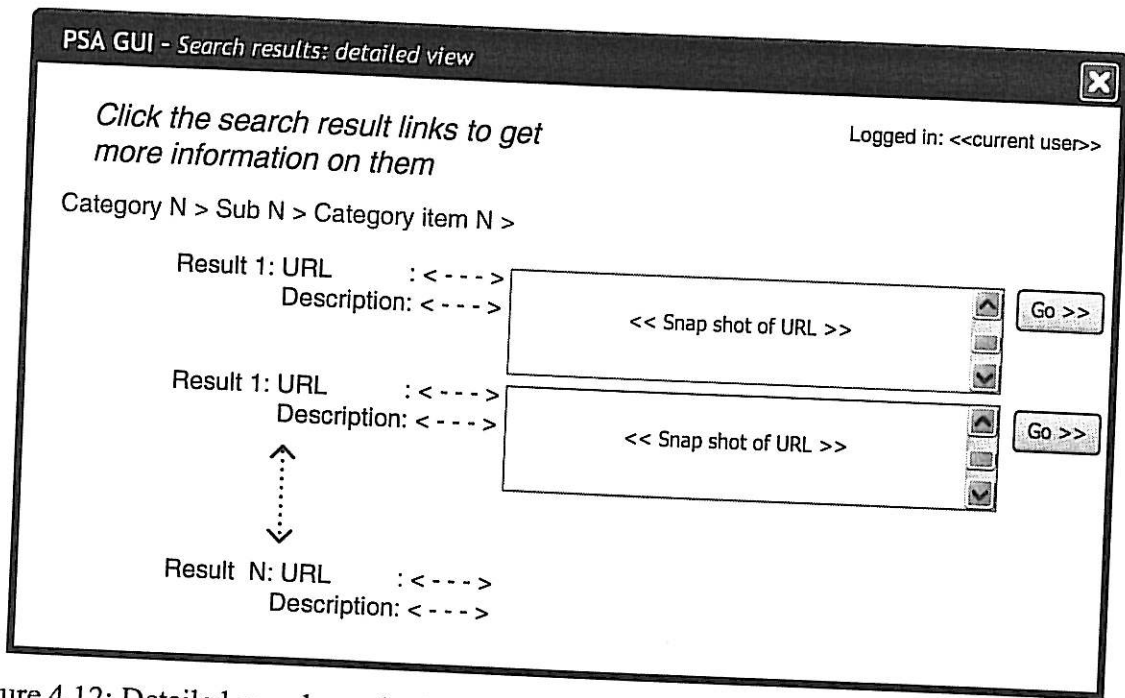


Figure 4.12: Detailed search result view, either of a dynamic page created or of a new search.

If the PSA finds result for the keyword in more than one category it then displays the relevant URLs category wise as shown in Figure 4.13 for the user to make a choice.

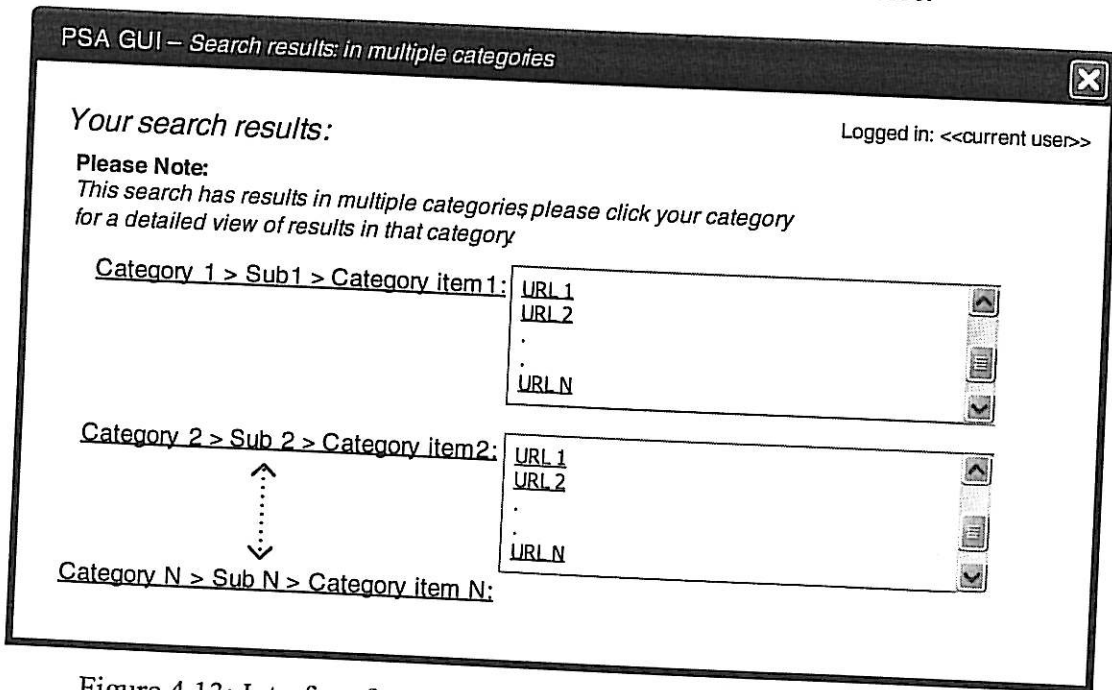


Figure 4.13: Interface for results in multiple categories or category items.

On selecting any of the categories the user is led to the interface shown in Figure 4.13. If the user clicks a specific URL for a search result, then the URL opens in a new, framed window where the upper frame is the PSA and the URL is loaded in the lower frame, as illustrated in

Figure 4.14. Also, the user will be requested to give feedback on the relevancy of the content, which determines whether to add this URL to current search database entries (in the case of a new search) or to update the ranking for it (in case if the URL is already in the database). The time is also counted while the user is viewing the URL through this window.

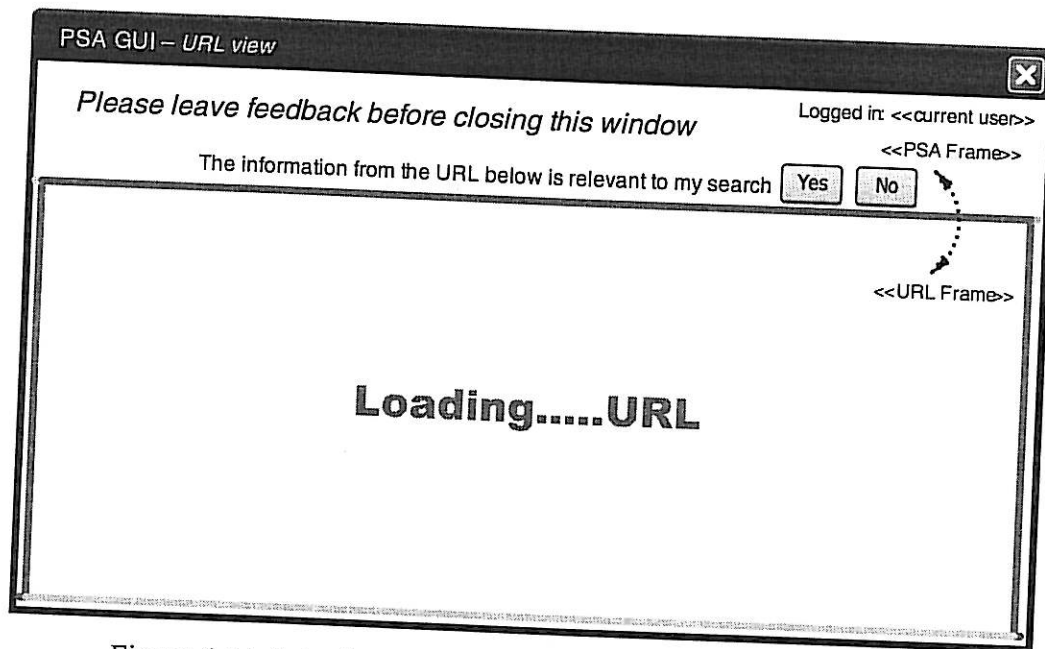


Figure 4.14: Interface for the display of details of a single URL.

If the user requested keyword is not in the local or global databases, then the PSA asks the user to select the relevant category details before making a search via the interface depicted in Figure 4.15.

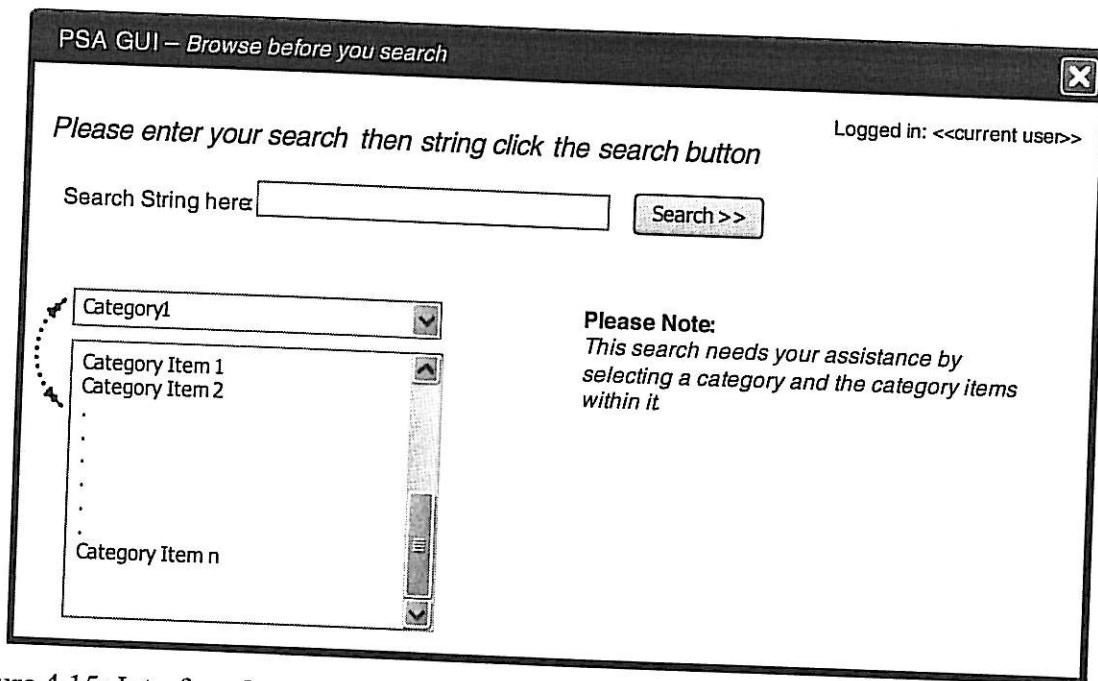


Figure 4.15: Interface for a search, with a keyword that is not in either the local or the global search databases.

If the user wishes to make changes to the category values then that is done using the interfaces shown in 4.16 and 4.17, respectively. The user first selects the category, using the interface shown in 4.16 and then makes the specific changes using the interface shown in Figure 4.17.

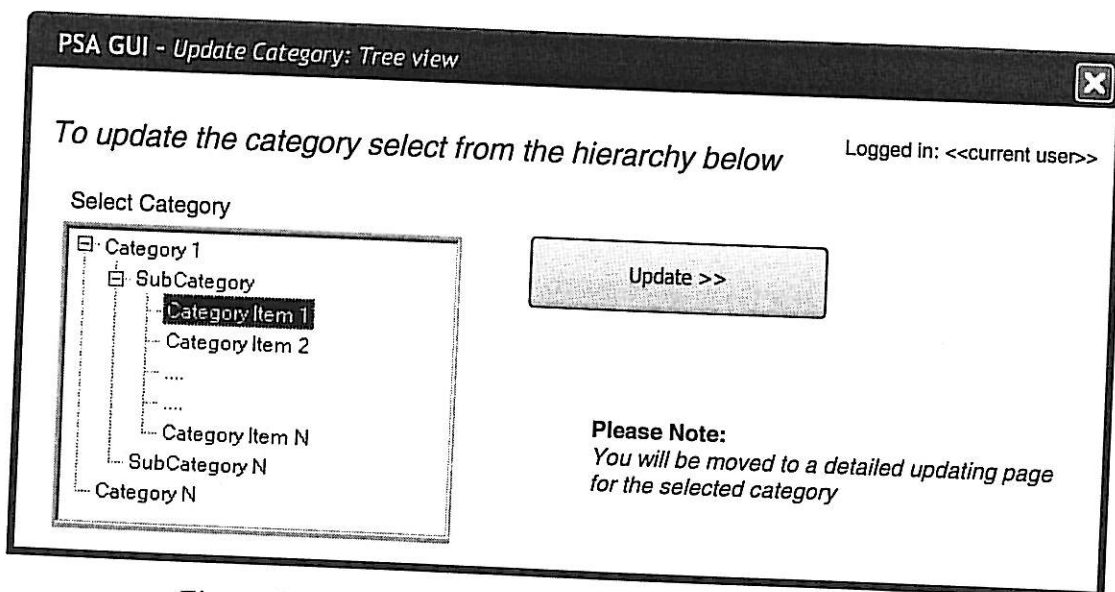


Figure 4.16: Tree hierarchy view of the category structure.

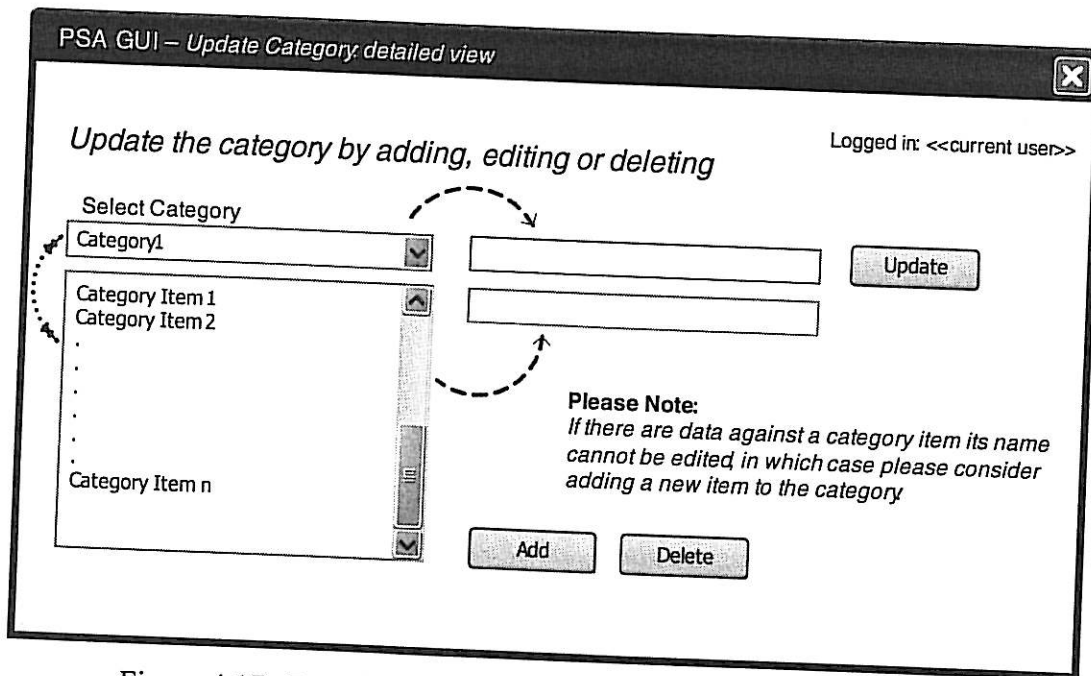


Figure 4.17: Detailed view interface for updating category details.

4.4 Prototype implementation of the PSA

As stated earlier, because of time constraints only the client component of the PSA prototype design above has been implemented. Detailed code of the implementation is attached in the appendix. However the main classes and methods are explained below.

GUI Class:- The PSA GUI is the interface class for the user on the client side of the system. This is a highly dynamic page as components are enabled or disabled or change behaviour on runtime as a result of creating fewer interfaces. The main methods used in this class are for capturing user data (like requests or feedback) and to show outputs (like search results, category updates etc.). Apart from the port operations, the GUI class also has responsibility for monitoring the active time the user spends on a particular link, as it is crucial in determining the link ranking.

SearchFilter Class: - The SearchFilter class has two implementations, one on the client side and the other on the server side. Basically this is the business class where all the decision-making and core functionalities of the PSA take place. The methods used in this class for generating search keyword sets, preparing search queries, changing information seeking paths based on the query details and user requirements etc.

PreferenceFilter Class: - This class is the learning part of the PSA both on the server and the client side. It has methods for managing various rankings and calculations. It also has the logical control over the categories. The PreferenceFilter class has potentially high expandability as, new methods can be added to learn more about the search preferences of the user, based on the current user search profile captured.

NetworkInterface Class: - This class, again both on the server and the client side, manages all the network interface needs of all the other classes in the PSA. It has methods for HTTP requests, Proxy handling and all sorts of functionalities to deal with the Internet. But, in the implemented prototype this class is just a gateway to the Internet for other classes.

ClientInterface Class: - This class exists only in the PSA server and this application interface class is specially designed for interaction between the PSA client and the server.

DatabaseInterface Class: - This class exists in both the clients and the server components of the PSA and is used for interacting with the database. This has been developed as part of the 3 tier architecture to give the application less coupling and more cohesion.

XMLInterface Class:- This class is used on both the client and the server side. The reason for the existence of this class is that the categories have been designed for implementation using XML and XML Schema. Several intermediate technologies used to communicate with and manipulate XML (like DOM), and therefore, the class is necessary to take these responsibilities, relieving the business logic classes from interaction burdens.

4.5 Assumptions made

The first page results from the search engines are taken for post processing, assuming that the search engine lists the highest priority listing first. The keyword-matching algorithm needs to be validated by means of a user survey. However, a general method was taken for implementation in this project. XML was the proposed data structure for implementing the category concept. However, a database was used, considering the time required to implement XML parses which can programmatically manage XML files and schema. It is assumed that for an experiential prototype, the impact of this change is negligible.

4.6 Summary

This chapter has described the implementation of the PSA prototype. Before proceeding to the actual software design stages, some supporting technologies and standards for software development were discussed. The chapter has also presented an analysis of the software requirements of the PSA conceptual model, using UML, as well as by the design and development of software artefacts for implementation of the prototype. The functionalities have also been analysed in detail using use cases and system sequence diagrams and a full class diagram for the PSA developed. ER Diagrams illustrates data structures and the interface designs are also developed for the PSA. The chapter also includes brief outlines of the classes in the PSA class diagram and the assumptions made during design and development. In the next chapter the prototype developed will be tested and bench-marked against current information search practices.

CHAPTER 5 TESTING, BENCHMARKING AND FINDINGS

This chapter includes the test criteria, results and benchmarking for the PSA prototype and candidate search engines. The search domain for this testing is limited to the University of Canberra website. The test criteria are framed to capture the relevancy of the results with respect to a particular group of users.

5.1 Software testing standards for the PSA

There are various standards for testing softwares. Initially, the acceptance testing was considered as the primary candidate for testing the PSA because, it allows testing of the functionalities of the PSA in an end user level. However, acceptance testing was later discarded as a test standard for the PSA because this test standard needs to have a Boolean output: pass or fail (Acceptance test in software engineering, 2005). There is generally no degree of success or failure can be measured with this software testing standard, which is not suitable for testing the PSA. On further analysis on the testing standards, it was found the *test cases*, a variation of the acceptance testing standard, is a more suitable testing standard to evaluate the performance and functionalities of the PSA. In software engineering, a *test case* is a set of conditions or variables under which a tester will determine if a requirement upon an application is partially or fully satisfied. For the test case standard, there is a *known input* and an *expected output*, which is worked out before the test. The known input should test a precondition and the expected output should test a postcondition. The written test cases include a description of the functionality to be tested taken from either the requirements or use cases, and the preparation required to ensure that the test can be conducted (Test case in software engineering, 2005).

5.2 Defining the test criteria: pre and post conditions

The test criteria were developed in accordance with the testing standard selected to assess the quality of the new system and the candidate search engines by measuring the efficiency, the responses and, most importantly, the relevancy of the result. Following are definitions of these three terms in the context of the testing.

Relevancy: - relevancy is highly personal and it is therefore difficult to define standards for its measurement. The best method is a user survey. However, because of the time limits, relevancy was measured based on a set of assumptions as follows:

- The search domain is set as the University Of Canberra website, www.canberra.edu.au.
- Relevancy is measured considering the user as a research student and by pre defining some of the preferences of such a user, which are listed in table 5.1 below.

Keyword (pre condition)	Preference assumed for a Research Student (post condition)
Job	Jobs in relation to research field
Fee	Fees for research students
Thesis	Guidelines for writing a thesis.
Scholarships	Scholarships for international research students.

Table 5.1: Assumed preferences against test keywords.

- The candidate systems selected for comparison are www.google.com, www.yahoo.com and www.altavista.com.
- How far the results are relevant to the assumption is judged from a research student's perspective. However, the results can be verified with a wider assumption and with greater variety in groups of users. This is just a matter of time.
- A good database entry has been made for the selected search keywords, as real use of the system comes with time and how long it has been used, hence how much data is available for that user.

Efficiency: - efficiency unveils how effectively the system can remember user preferences rather than asking users to recall their preferences each time. Apparently, it tests how the system learns by capturing user input and feedback and uses that information to increase the efficiency of current search practice.

Responsiveness: - responsiveness is not how fast the results are displayed to the user, but how fast the user receives relevant results.

5.3 Test results of the PSA

The PSA can be accessed using a temporary domain www.psa.dl.am. The test results for the PSA following the search criteria in Table 5.1 are tabulated in Tables 5.2 to 5.5 below.

No	Test No.1	Keyword: JOB	Service: www.psa.dl.am	Time: 1 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/careers/jobs/notices-overseas/voluntary-research-assistant131004.html		Job vacancy listing for <i>Voluntary Research Assistant</i> in South West Madagascar	*****	
2	http://www.canberra.edu.au/hr/jobs/		Official job posting page of the University of Canberra	***	
3	http://www.canberra.edu.au/recruitment/jobs/job-05-2076.html		Job vacancy listing for <i>Associate Professor</i> in the University of Canberra	**	
4	http://www.canberra.edu.au/recruitment/jobs/job-05-2076.html		Issues regarding job prospects of research students by Professor Don Aitkin, Vice Chancellor and President, University of Canberra, on 31 March 1999	*	
5	http://www.canberra.edu.au/recruitment/jobs/appinfo.html		Information for applicants about the University of Canberra as an employer and how to apply for a job	***	
Average			2.8	Total	14

Table 5.2: Test results of the PSA for keyword 'Job'

No	Test No.2	Keyword: FEE	Service: www.psa.dl.am	Time: 1.5 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/study/studying/fees/international-fees-pg.html		A list of international student fees for 2006 Postgraduate Programs	*****	
2	http://www.canberra.edu.au/student-services/fees/fee-help		Student administration help and assistance regarding fee payment	***	
3	http://www.canberra.edu.au/student-services/fees/fee_policy		University policy on domestic and international student fees	**	
4	http://www.canberra.edu.au/student-services/fees/domestic-postgraduate-tuition-fee-rates		Domestic Postgraduate Tuition Fee Rates, Calculation of Domestic Tuition Fees etc.	**	
Average			3	Total	12

Table 5.3: Test results of the PSA for keyword 'Fee'

No	Test No.3	Keyword: THESIS	Service: www.psa.dl.am	Time: 1 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/secretariat/goldbook/append.html		Higher Degrees by Research: Policy and Procedures, Appointment of Examiners for thesis evaluation, Certificate of Completion of Thesis for Higher Degree by Research etc.	*****	
2	http://www.canberra.edu.au/studyskills/writing/		Page from the academic skills program, with links to thesis proposals, thesis writing, essays etc.	***	
Average			4	Total	8

Table 5.4: Test results of the PSA for keyword 'Thesis'

No	Test No.4	Keyword: SCHOLARSHIPS	Service: www.psa.dl.am	Time: 0.5 Sec	
		URL	DESCRIPTION	Relevancy	
1		http://www.canberra.edu.au/student-services/scholarships/international-research-scholarships	List of all International Research Scholarships available in the University of Canberra.	*****	
2		http://www.canberra.edu.au/student-services/scholarships/research-scholarships	A list of all available doctoral and masters degrees by research scholarships available in the University of Canberra for both domestic and international students.	****	
3		http://www.canberra.edu.au/news_events/media_releases/media_03_08_04_3.html	News regarding the University of Canberra proposals for research funding	**	
4		http://www.canberra.edu.au/news_events/media_releases/2003_backup/media_12_9_03.html	News regarding new research scholarships from the University of Canberra	***	
5		http://www.canberra.edu.au/handbook2003/university/university-Bursarie.html	A list of all available scholarships in the University of Canberra.	*	
Average			3	Total	15

Table 5.5: Test results of the PSA for keyword ‘Scholarships’

5.4 Test results for the search engines

The test results for candidate search engines following the search criteria in Table 5.1 are described below.

www.google.com

The test results for Google.com following the search criteria in Table 5.1 are tabulated in Tables 5.6 to 5.9 below.

No	Test No.1	Keyword: JOB	Service: www.Google.com	Time: 0.55 Sec	
		URL	DESCRIPTION	Relevancy	
1		http://www.canberra.edu.au/tertiary-to-work/	Job fair website, detailing the job seek assistance and important dates of job fair etc.	**	
2		http://www.canberra.edu.au/hr/jobs/	Official job posting page of the University of Canberra	***	
3		http://www.canberra.edu.au/careers/development/applications.html	University career development website detailing how to apply for a job, how to prepare a CV, a covering letter etc.	**	
4		http://www.canberra.edu.au/special-ed/papers/shaddock2001c.html	Link in favour of disabled people, support for disabled job-seekers.	--	
5		http://www.canberra.edu.au/recruitment/jobs/appinfo.html	Information for applicants about the University of Canberra as an employer and how to apply for a job	***	
Average			2	Total	10

Table 5.6: Test results of google.com for keyword ‘Job’

No	Test No.2	Keyword: FEE	Service: www.Google.com	Time: 0.25 Sec	
		URL	DESCRIPTION	Relevancy	
1		http://www.canberra.edu.au/student-services/fees/fee-help	Student administration help and assistance regarding fee payment	***	
2		http://www.canberra.edu.au/student-services/fees/domestic-postgraduate-tuition-fee-rates	Domestic Postgraduate Tuition Fee Rates, Calculation of Domestic Tuition Fees etc.	**	
3		http://www.canberra.edu.au/student-services/fees/fee_policy	University policy on domestic and international student fees	**	
4		http://www.canberra.edu.au/secretariat/feepay.html	Administrative Procedural Guidelines for Fee Paying Short Courses	--	
		Average	1.75	Total	7

Table 5.7: Test results of google.com for keyword 'Fee'

No	Test No.3	Keyword: THESIS	Service: www.Google.com	Time: 0.18 Sec	
		URL	DESCRIPTION	Relevancy	
1		http://www.canberra.edu.au/courses/index.cfm?action=detail&subjectid=6019&year=2005	Page explaining honours thesis subject details.	--	
2		http://www.canberra.edu.au/courses/index.cfm?action=detail&subjectid=6609&year=2005	Page explaining honours Research thesis subject details	--	
		Average	0	Total	0

Table 5.8: Test results of google.com for keyword 'Thesis'

No	Test No.4	Keyword: SCHOLARSHIPS	Service: www.Google.com	Time: 0.25 Sec	
		URL	DESCRIPTION	Relevancy	
1		http://www.canberra.edu.au/student-services/scholarships	General information regarding scholarships and prizes in the University of Canberra	**	
2		http://www.canberra.edu.au/student-services/scholarships/scholarships-for-indigenous-students	Scholarships available to indigenous students.	--	
3		http://www.canberra.edu.au/student-services/scholarships/scholarships-awarded-automatically	Special scholarships and undergraduate scholarships are described	--	
4		http://www.canberra.edu.au/student-services/scholarships/international-research-scholarships	List of all International Research Scholarships available in the University of Canberra.	*****	
5		http://www.canberra.edu.au/student-services/scholarships/scholarships-policy-and-guidelines	The University of Canberra Scholarships Policy & Guidelines	**	
		Average	1.8	Total	9

Table 5.9: Test results of google.com for keyword 'Scholarships'

The test results for AltaVista.com following the search criteria in Table 5.1 are tabulated in Tables 5.10 to 5.13 below.

No	Test No.1	Keyword: JOB	Service: www.Altavista.com	Time: 1 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/uc/lectures/mantech/manpol/sem981/unit3432/HRM_1_Week_9_Job_Design,_Recruitment,_Selection.txt		A text document detailing job design, recruitment and selection criteria	--	
2	http://www.canberra.edu.au/tertiary-to-work/		Job fair website, detailing the job seek assistance and important dates of job fair etc.	**	
3	http://www.canberra.edu.au/hr/services/classifications/handbook.html		Handbook on the Evaluation Process for General Staff Positions of the University of Canberra	*	
4	http://www.canberra.edu.au/special-ed/papers/shaddock2001c.html		Link in favour of disabled people, support for to disabled job-seekers.	--	
5	http://www.canberra.edu.au/uc/lectures/mantech/manpol/sem972/unit3488/Unit3488+HRM2_lecture_7.2_Job_Selection_and_Competency-Based_Assessment.txt		Text file describing How HRD Can Improve the Job Selection Process	--	
Average			0.6	Total	3

Table 5.10: Test results of AltaVista.com for keyword 'Job'

No	Test No.2	Keyword: FEE	Service: www.Altavista.com	Time: 0.28 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/student-services/fees/fee-help		Student administration help and assistance regarding fee payment	***	
2	http://www.canberra.edu.au/student-services/fees/fee-help		Student administration help and assistance regarding fee payment	***	
3	http://www.canberra.edu.au/secretariat/feepay.html		Administrative Procedural Guidelines for Fee Paying Short Courses	--	
4	http://www.canberra.edu.au/secretariat/legislation/rules/32.html		FEES RULES 1995 University of Canberra	--	
Average			1.5	Total	6

Table 5.11: Test results of AltaVista.com for keyword 'Fee'

No	Test No.3	Keyword: THESIS	Service: www.Altavista.com	Time 0.74 Sec
	URL		DESCRIPTION	Relevancy
1	http://www.canberra.edu.au/QPR/1998/White1998.pdf		Thesis Writing For Supervisors: Article from Victoria University of Technology, Australia	**
2	http://www.canberra.edu.au/secretariat/goldbook/forms/thesisrqmt.pdf		Administrative requirements for a thesis: Appendix to Higher degrees by research: policy and procedures, part of the gold book	*****
Average			3.5	Total
				7

Table 5.12: Test results of AltaVista.com for keyword 'Thesis'

No	Test No.4	Keyword: SCHOLARSHIPS	Service: www.Altavista.com	Time 0.85 Sec
	URL		DESCRIPTION	Relevancy
1	http://www.canberra.edu.au/student-services/scholarships		General information regarding scholarships and prizes in the University of Canberra	**
2	http://www.canberra.edu.au/handbook2003/university/university-Bursarie.html		A list of all available scholarships in the University of Canberra.	*
3	http://www.canberra.edu.au/student-services/scholarships/undergraduate-scholarships		Undergraduate Scholarship details	**
4	http://www.canberra.edu.au/student-services/scholarships/research-scholarships		A list of all available doctoral and masters degrees by research scholarship available in the University of Canberra for both domestic and international students.	****
5	http://www.canberra.edu.au/student-services/scholarships/scholarships-awarded-automatically		Special scholarships and undergraduate scholarships are described	--
Average			1.8	Total
				9

Table 5.13: Test results of AltaVista.com for keyword 'Scholarships'

The test results for Yahoo.com following the search criteria in Table 5.1 are tabulated in Tables 5.14 to 5.17 below.

No	Test No.1	Keyword: JOB	Service: www.yahoo.com	Time: 0.31 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/tertiary-to-work/		Job fair website, detailing job seeker assistance and important dates of job fairs	**	
2	http://www.canberra.edu.au/careers/development/careers-guide.html		Careers Information Access Guide	**	
3	http://www.canberra.edu.au/uc/lectures/mantech/manpol/sem981/unit3432/HRM_1_Week_9_Job_Design,_Recruitment,_Selection.txt		A text document detailing job design, recruitment and selection criteria	--	
4	http://www.canberra.edu.au/hr/services/classifications/handbook.html		Handbook on the Evaluation Process for General Staff Positions of the University of Canberra	*	
5	http://www.canberra.edu.au/careers/links.html		Page with helpful links to employment websites.	**	
Average			1.4	Total	7

Table 5.14: Test results of Yahoo.com for keyword 'Job'

No	Test No.2	Keyword: FEE	Service: www.yahoo.com	Time: 0.24 Sec	
	URL		DESCRIPTION	Relevancy	
1	http://www.canberra.edu.au/student-services/fees/fee-help		Student administration help and assistance regarding fee payment	***	
2	http://www.canberra.edu.au/student-services/fees/domestic-postgraduate-tuition-fee-rates		Domestic Postgraduate Tuition Fee Rates, Calculation of Domestic Tuition Fees etc.	**	
3	http://www.canberra.edu.au/secretariat/feepay.html		Administrative Procedural Guidelines for Fee Paying Short Courses	--	
4	http://www.canberra.edu.au/secretariat/legislation/rules/32.html		FEES RULES 1995 University of Canberra	--	
Average			1.25	Total	5

Table 5.15: Test results of Yahoo.com for keyword 'Fee'

No	Test No.3	Keyword: THESIS	Service: www.yahoo.com	Time: 0.19 Sec
	URL		DESCRIPTION	Relevancy
1	http://www.canberra.edu.au/studyskills/writing/litreview.html		ABOUT WRITING A LITERATURE REVIEW	****
2	http://www.canberra.edu.au/QPR/1998/White1998.pdf		Thesis Writing For Supervisors: Article from Victoria University of Technology, Australia	**
Average			3	Total
				6

Table 5.16: Test results of Yahoo.com for keyword 'Thesis'

No	Test No.4	Keyword: SCHOLARSHIPS	Service: www.yahoo.com	Time: 0.07 Sec
	URL		DESCRIPTION	Relevancy
1	http://www.canberra.edu.au/student-services/scholarships		General information regarding scholarships and prizes in the University of Canberra	**
2	http://www.canberra.edu.au/handbook2003/university/university-Bursarie.html		A list of all available scholarships in the University of Canberra	*
3	http://www.canberra.edu.au/news_events/media_releases/media_03_08_04_3.html		News regarding University of Canberra proposals for research funding	**
4	http://www.canberra.edu.au/student-services/scholarships/undergraduate-scholarships		Undergraduate Scholarship details	**
5	http://www.canberra.edu.au/student-services/scholarships/research-scholarships		A list of all available doctoral and masters degrees by research scholarship available in the University of Canberra for both domestic and international students.	****
Average			2.2	Total
				11

Table 5.17: Test results of Yahoo.com for keyword 'Scholarships'

5.5 Bench marking

Bench marking for the test results for the PSA and candidate search engines was done based on the result tables above in section 5.3. Firstly, the average relevancy, termed the Relevancy Index, was found for each table for each candidate system. The candidate average was then calculated by taking the average of the tables. This is depicted in Table 5.18 below.

Keyword		PSA	Google	AltaVista	Yahoo
Relevancy Averages	Job	2.80	2.00	0.6	1.40
	Fee	3.00	1.75	1.5	1.25
	Thesis	4.00	0.00	3.5	3.00
	Scholarship	3.00	1.80	1.5	2.20
Total		12.8	5.5	7.1	7.85
Average (Relevancy index)		3.2	1.4	1.8	2

Table 5.18: Relevancy index of the PSA and the candidate search engines from a research student perspective.

The results data tables under section two of this chapter also contain the time taken to display the search results. Table 5.19 below shows the compilation of the time averages for the PSA and the candidate search engines.

Keyword		PSA	Google	AltaVista	Yahoo
Time Averages	Job	1.00	0.55	1.00	0.31
	Fee	1.50	0.25	0.28	0.24
	Thesis	1.00	0.18	0.74	0.19
	Scholarship	0.50	0.25	0.85	0.07
Total (in Seconds)		4.00	1.23	2.87	0.81
Average		1	0.31	0.72	0.20

Table 5.19: Averages of the time taken by the PSA and the candidate search engines under various search criteria

A relevancy response factor (time taken for delivering the relevant results not just the results) is just an indication of relevant results and can be taken by processing the data in compilation tables 5.18 and 5.19 as follows,

$$\text{Relevancy Response Factor} = \text{Relevancy Index} - \text{Average Response Time}$$

It is important to note that the above line is not a mathematically valid equation as it contains an abstract variable of the relevancy index. Instead this can be considered as an effort to test the relevancy in a more accountable way. Applying the above relationship, the relevancy response factor of the PSA and the candidate search engines are calculated in table 5.20 below.

CANDIDATE SYSTEM	RELEVANCY INDEX	AVERAGE RESPONSE TIME	RELEVANCY RESPONSE FACTOR
PSA	3.20	1.00	2.20
Google	1.40	0.31	1.09
AltaVista	1.80	0.72	1.08
Yahoo	2.00	0.20	1.80

Table 5.20: An effort to find new indicators for relevancy based on the relevancy factor and the time

Figure 5.1 illustrates a graphical representation of the above-discussed relationships.

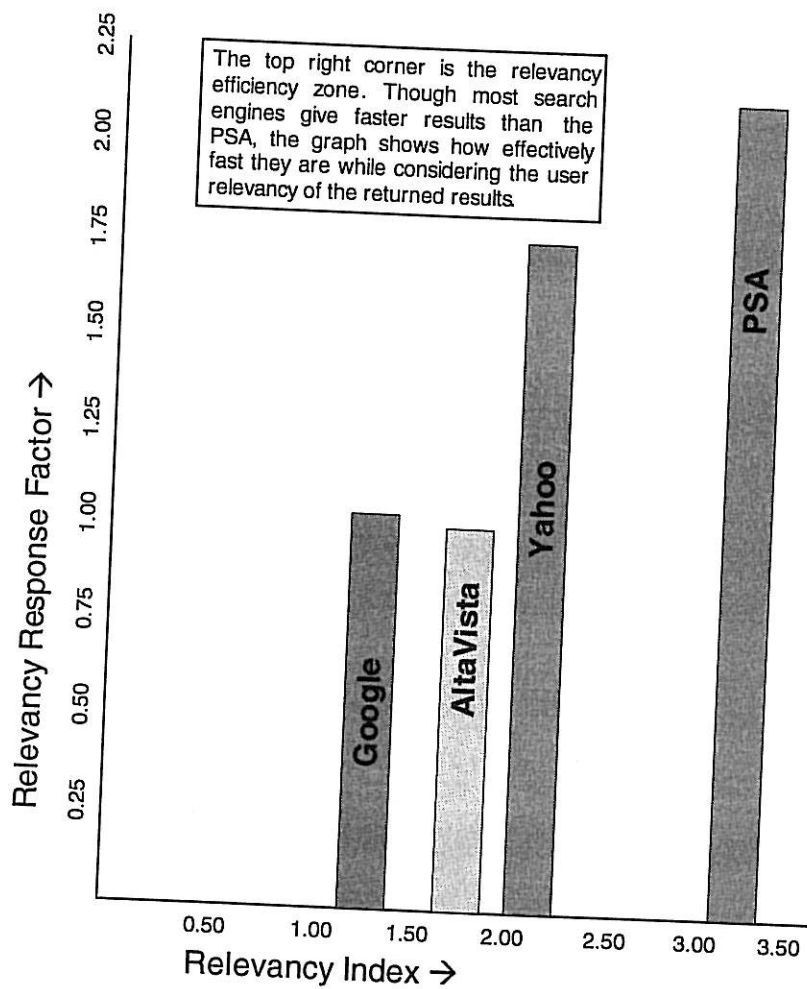


Figure 5.1: Graph depicting the *relevancy of the results* and the *response time* for delivering relevant results comparing the PSA with other candidate search engines.

5.6 Summary

This chapter has described the development of the test criteria, the testing of the PSA and also the bench marking of the test results with the candidate systems. The testing and benchmarking show a clear advantage for the PSA over the candidate search engines. Certain facts need to be considered in summarising the findings of the test. The test was carried out under strict assumptions that the search domain was limited to University of Canberra website, and all the search criteria were oriented in a single direction, that is from the perspective of a research student. Also, the PSA prototype was not implemented incorporating the client-server model. Instead, only the client side was implemented as it carries the core

concepts behind the PSA. Hence, there could be variations if a detailed test were conducted in a fully developed prototype, with more users and in the WWW domain. However the aim of this experiment lay more in establishing the trends and indicating the effectiveness of the PSA is in getting relevant, and hence personalised results for its user.

The three quality factors against which the test was carried out were relevancy, responses and efficiency. The graph shown in Figure 5.1 clearly demonstrates PSA's indicative advantage over the candidate system in regard to these qualities. Because the PSA takes the input from other search engines, obviously it cannot match the response time of the other search engines, as the PSA consumes time for pre and post processing between user and search engine. Apart from this, the efficiency of the server in which the service is hosted also affects the responsiveness. However, the effective time, that is, the time taken by the user to receive relevant results, are better with the PSA. Moreover, a repeated search can actually avoid search engine interaction hence saving the pre and post processing time. One other factor that determines the response time is the web server in which the programme (or service) is running. Overall, testing of the PSA prototype has resulted in positive outcomes, confirming the validity and practicability of the conceptual model of the PSA developed during this research. In the concluding chapter, future modifications and potential advancements for the PSA will be discussed.

CHAPTER 6 CONCLUSIONS AND FUTURE WORK

This chapter concludes the research personalisation web information searching through an agent based architecture with a summary of research findings. The chapter also considers potential areas of refinement and possible goals for the PSA in the future.

6.1 Summary of research findings

The objective of this research, restating from Chapter 1, is:

- To develop an effective technique to find user relevant information from search engine results
- To develop a technique to capture user feedbacks
- Learn from captured user feedback and user search practice
- Assisting the user to frame effective keyword combinations

The PSA system has been successful in meeting these objectives. The major research findings during this effort are:

- Developed four different ranking algorithms, namely, the keyword generating algorithm, the link ranking, the search engine ranking and the life saving ranking towards meeting the objective of developing an effective technique to find user relevant information.
- Developed a novice method to find the user relevant URLs through a combination of calculating the active time the user spend on the URL and user feedback on relevancy of the URLs.
- Developed a client-server model enabling generalisation-specialisation on information search. This model, helps to build a generalised, however, personalised information portfolio in the server side, which is a source of more organised information, than search engines.
- Developed an automated mechanism for generating effective keyword combinations via developing the keyword generating algorithm.

6.2 Conclusions

The goal of this research work was to find an effective technique for personalising current web search practice by using existing information search resources. The Personalised Search Agent was therefore constructed through the development of a conceptual model and testing

by means of a PSA prototype followed by testing in sample scenarios and comparing the results with candidate systems to reach this goal.

Because of the time constraints, strict limits were applied, especially to the development of the prototype. The prototype developed therefore is only a working model with just the core functionalities of the conceptual model of the Personalised Search Agent. However, much effort was put into developing an elaborate conceptual model for the Personalised Search Agent, addressing most of the issues raised through the research questions.

The PSA exploits information search resources such as popular search engines, using more than one search engine effectively. Indeed, the PSA actually senses which search engine has delivered more successful links to a particular user and prioritises the search results accordingly using the Search Engine Ranking mechanism. This means, the PSA not only uses the meta-search engine feature, but also uses it effectively.

The PSA is also successful in capturing feedback from the user, either by explicitly requesting feedback from the user or by learning the user's habits. In the explicit method, the PSA requests feedback on the relevancy of a link presented to the user. However, the PSA also receives many implicit inputs, for example, active time, when the user gives feedback on an explicit request from the user. These inputs are used to make various calculations regarding the relevancy of the link. Care has been taken to request only minimal non-task related inputs from the user, in consideration of user comfort and time. Moreover, the PSA effectively reduces the user inputs through features such as keyword generating algorithms, which generate possible combinations of search string words combined with the category values and automatically switch to search mode with different combinations if the user is unhappy with the presented results. In conversational method, that is using a search engine, the user has to provide different combinations of keywords explicitly, to achieve related results from the search engines.

The results for the test carried out in Chapter 5, clearly show the PSA is successful in delivering relevant and therefore personalised results to the user. Though the PSA shares many techniques (such as the category concept) with similar search personalising systems, explained in Chapter 2, it is still relevant in the current theoretical framework, and makes original contributions to the field of knowledge because of:

- its unique features such as the client-server based multi-layer architecture, determining the importance of a page by calculating the active time spent by the user on a page
- Prioritising the search results based on the success rate of the search engines for a particular user.

With the client-server architecture, the PSA server, is potentially a good resource for an organised, personalised information database. Each and every link in the server database is reviewed and accepted by at least one user, which confirms that it is relevant to that particular category. This is particularly important, considering that, information search services like the MSN Search are more likely to present human-powered listings (How search engines work, 2002), whereas the PSA has all its cached URLs reviewed and ranked during the search process itself.

Though PSA fulfils its goal of achieving more personalised search results compared to current search practices, it still has potential areas of refinement, which throw light on future work in regard to the PSA, and are explained in the following section of this chapter.

6.3 Future Work and Extensions for the PSA

Though a prototype for the PSA was developed as part of the research, only the client side was implemented because of the strict time frame. Also, the current prototype does not identify multiple users. This could easily be solved with a general login but, because, it is merely a technical issue, it is excluded from the current prototype. An enhanced prototype which implements the full features of the PSA as developed and designed in the conceptual framework, will enable more sophisticated tests which could reveal more run-time issues in personalisation of web-searching using the PSA (this iteration is part of the selected research methodology: Systems Development).

The conceptual framework has well developed methods to tap into user action and to change it to useful information in establishing the relevancy of the pages. However, the global and local categories and preference databases, have more potential uses than are actually served in the current model. Finding and mapping inter-category relations would enable the PSA to predict a keyword with its associated category. This apparently needs the application of data

mining and artificial intelligence techniques to the already gathered data. Because proper data is needed to apply the above techniques effectively, the PSA has excellent potential for further research, by creating unique user profiles, which, as explained in Chapter 3, comprises organised information about the personal search history of a user.

Another issue with the PSA is the technical barrier to tapping information from various search engines. Most search engines develop highly complicated HTML pages as a result and hence it is a difficult task to filter out necessary information from such services. An effective alternative for this has to be found. Application Programmes Interfaces (API) are promising in this regard though only a few search engines, like google.com, are offering such facilities.

Currently the PSA does not provide a mechanism to check the current availability of a cached URL, that is, whether the information resource is relocated or the service is unavailable. An effective mechanism has to be developed to overcome this limitation leaving more scope for further research on the PSA. Also there is a noted limitation in the currency of the results cached by the PSA. At present the PSA updates its cached results only with user interaction. That means, if the user does not prompt for a new search, the cached results will not be updated. This leaves scope for a new technique which can update the cached results automatically, without the interaction of the user.

Finally, though XML is the data structure for the PSA conceptual model, during development of the prototype database (MS Access) was used for data storage. Again this is merely technical, and can be incorporated into future refinements of the current prototype of the PSA.

BIBLIOGRAPHY

Acceptance test in software engineering: From Wikipedia, the free encyclopaedia. (2005). Retrieved 3 August 2005, from http://en.wikipedia.org/wiki/Acceptance_test

Ah-Hwee,T., Hwee-Leng, O., Hong Pan, J. & Qiu-Xiang, L. (2001). *FOCI: A Personalized Web Intelligence System*. Singapore: 21 Heng Mui Keng Terrace.

Analia, A. & Daniela, G. (2000). *PersonalSearcher: An Intelligent Agent for Searching Web Pages*. Argentina: La Pampa.

Baudoin, C. & Hollowell, G. (1996). *Realizing the Object-Oriented Lifecycle*. New Jersey: Prentice Hall, Upper Saddle River.

Björkman, E. & Ohlsson, R. (2000). *Research på Internet*. Stockholm: Liber.

Björn, H. (1997). *Intelligent Software Agents on the internet: an inventory of currently offered functionality in the information society and prediction of future developments*. USA: University of California.

Boris, R., Parisch, B., Paul M. & Mícheál, O. (2004). *User Profiling for Content Personalisation in Information Retrieval*. Ireland: Waterford Institute of Technology.

Brooks, R.A. (1991). *Intelligence without Representation: Artificial Intelligence*. USA: Cambridge, p.9-12.

Chignell, M. H., Gwizdka, J. & Bodner, R.C. (1999). *Discriminating meta-search: a framework for evaluation, Information Processing & Management*, vol 35, p. 337, Elsevier Science Ltd.

Convey, J. (1992). *Online Information Retrieval*. Library Association Publishing: London.

Cooke, A. (1999). *A guide to finding quality information on the Internet*. London: Library Association Publishing.

Cyrus, S. & Yi-Shin, C. (2003). *Web Information Personalization: Challenges and Approaches*. USA: University of Southern California, Los Angeles, CA.

Edelstein, H. (1994). *Unraveling Client/Server Architecture*. DBMS, Vol.34, p.7.

Extensible Markup Language: XML activity statement. (2002). Retrieved 2 January, 2005, from <http://www.w3.org/XML>

General Subject Directories: Table of Features. (2005). Retrieved 2 January 2005, from <http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/SubjDirectories.html#Guides>

Getting Started with ASP.NET: What is ASP.NET. (2005). Retrieved 10 January, 2005, from <http://msdn.microsoft.com/asp.net/gettingstarted/>

- Giuseppe, A. & Umberto, S. (1999). *User profile modelling and applications to digital libraries*. Italy: C.N.R.
- Harter, S. (1986). *Online Information Retrieval: Concepts, Principles and Techniques*. Orlando: Academic Press.
- Haveliwala, T.H. (2002). *Topic-sensitive PageRank: Proceedings of the 11th International World Wide Web Conference*. USA: Hawaii, p. 517 – 526.
- How search engines work: hybrid search engines. (2002). Retrieved 12 April, 2005, from <http://searchenginewatch.com/webmasters/article.php/2168031>
- Hyacinth, S.N. (1996) *Software Agents: An Overview, Knowledge Engineering Review*, Vol.11, No 3, pp.1-40.
- John, P.M. (2003). *A Multiple Model Approach to Personalised Information Access*. Ireland: Dublin.
- Khawar, Z.A. & Cary, E.U. (2002). *Developing Enterprise Java Applications with J2EE™ and UML*. p. 93 – 121.
- Kristy, W. (2000). *Research methods for students and professionals*. NSW: Waga Waga.
- Large, A., Tedd, L. A. & Hartley, R. J. (1999). *Information Seeking in the Online Age*. London: Bowker-Saur.
- Lawrence, S. & Giles, C. L. (1999, July 8). Accessibility of Information on the Web. *Nature*, Vol 400, p.107–109. USA
- Linell & Rose, M. (2001). *Electronic Information Search-tools – A study of desired features*. Sweden: Jönköping.
- Marchionini, G. (1997). *Information Seeking in Electronic Environments*. UK: Cambridge University Press
- Maring, B. (1996 May). Object-Oriented Development of Large Applications, *IEEE Software* 13 (3). 33-40.
- Marshall, C. & Rossman, G.B. (1995). *Designing qualitative research, 2nd edn*. Sage. CA: Thousand Oaks.
- Meta search engines: What Are "Meta-Search" Engines? How Do They Work?. (2004). Retrieved 29 January, 2005, from <http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html>
- Microsoft Introduces Highly Productive .NET Programming Language: C#. (2005). Retrieved 20 March 2005, from <http://www.microsoft.com/presspass/press/2000/jun00/CsharpPR.msp>

Newell, D. & Machura, M. (1995). "Interoperable Object Models for Large Scale Distributed Systems," 30-31. *Proceedings. International Seminar on Client/Server Computing. La Hulpe, Belgium, October 30-31 1995.* London.

Nunamaker, j., Chen , M. & Purdin, T. (1991). Systems development in information systems research. *Journal of Management Information Systems*, 7(3), 89-106.

Recommended subject directories: UC Berkeley - Teaching Library Internet Workshops. (2005). Retrieved 11 May, 2005, from <http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/SubjDirectories.html>

Rapoport, R.N. (1970). *Three Dilemmas in Action Research, Human Relations*, (23:4), pp. 499-513.

Review of AltaVista. (2003). Retrieved 11 November 2004 from <http://www.notess.com/search/dir/altavista/index.html>

Review of Google.(2004). Retrieved 27 February, 2005, from <http://searchengineshowdown.com/features/google/review.html>

Review of Yahoo search. (1999). Retrieved 14 November, 2004, from <http://searchengine showdown.com/features/yahoo/review.html>

Search engine architecture. (n.a). Retrieved 01 December, 2004, from <http://www.searchenginewatch.com/Webmasters/work.htm> (Authorised access).

Stan, F. & Art, G. (1996). *Is it an Agent, or Just a program?: A Taxonomy for Autonomous Agents.* Germany: University of Memphis, Springer-Verlag.

Test case in software engineering: From Wikipedia, the free encyclopaedia. (2005). Retrieved 3 August 2005, from http://en.wikipedia.org/wiki/Test_case

The WDG's glossary of terms: a definition for search engine. (1997). Retrieved 11 November, 2004, from <http://www.htmlhelp.com/reference/glossary.html>

Three Tier Software Architectures: Technical Detail. (2004). Retrieved 03 December, 2005, from http://www.sei.cmu.edu/str/descriptions/threetier_body.html

Trends & Statistics: The Web's Richest Source. (2005). Retrieved 1 February, 2005, from http://www.clickz.com/stats/web_worldwide

Trist, E. (1976). *Experimenting with Organizational Life: The Action Research Approach.* New York: Plenum.

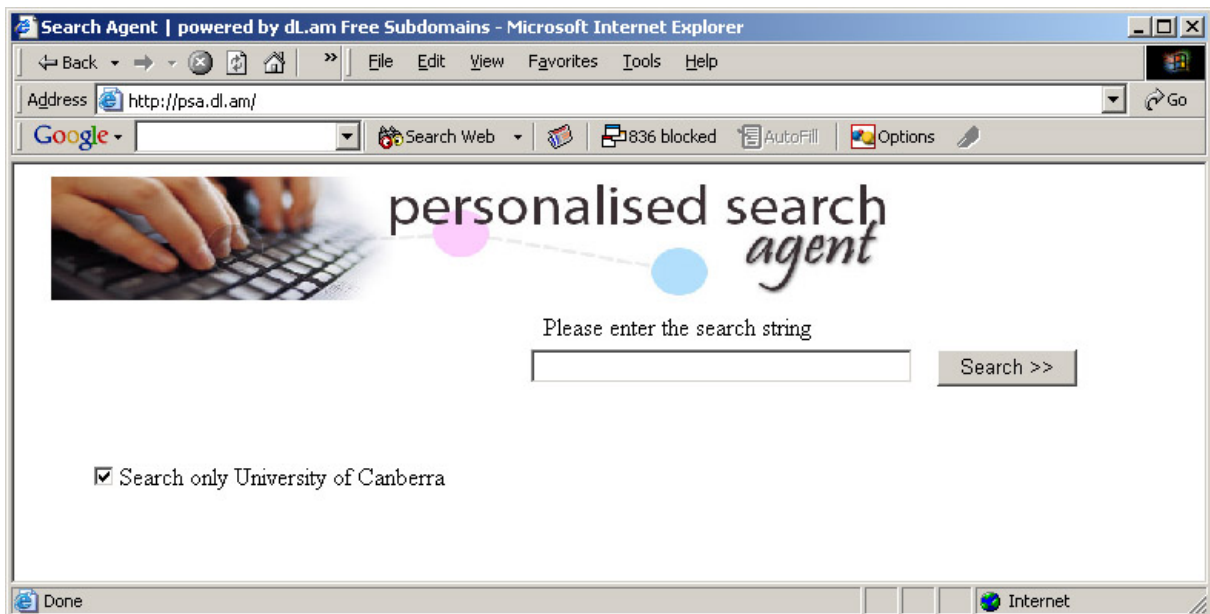
UCC Glossary: XML Standardisation. (2003). Retrieved 2 January, 2005, from <http://usnet03.uc-council.org/glossary>

UML Resource Page: Introduction to OMG's Unified Modelling Language (2005). Retrieved 11 February, 2005, from http://www.omg.org/gettingstarted/what_is_uml.htm

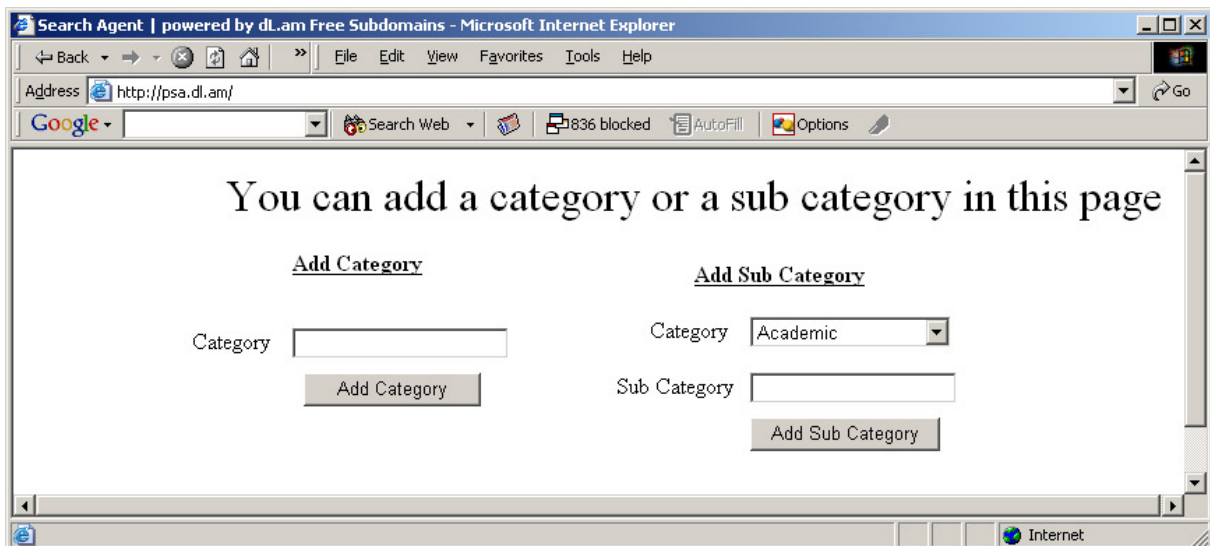
What is .NET: Driving business values with the Microsoft .NET platform. (2005). Retrieved 11 January, 2005, from <http://www.microsoft.com/Net/Basics.aspx>

What is the "Invisible Web"?. (2004). Retrieved 12 May, 2005, from <http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/InvisibleWeb.html>

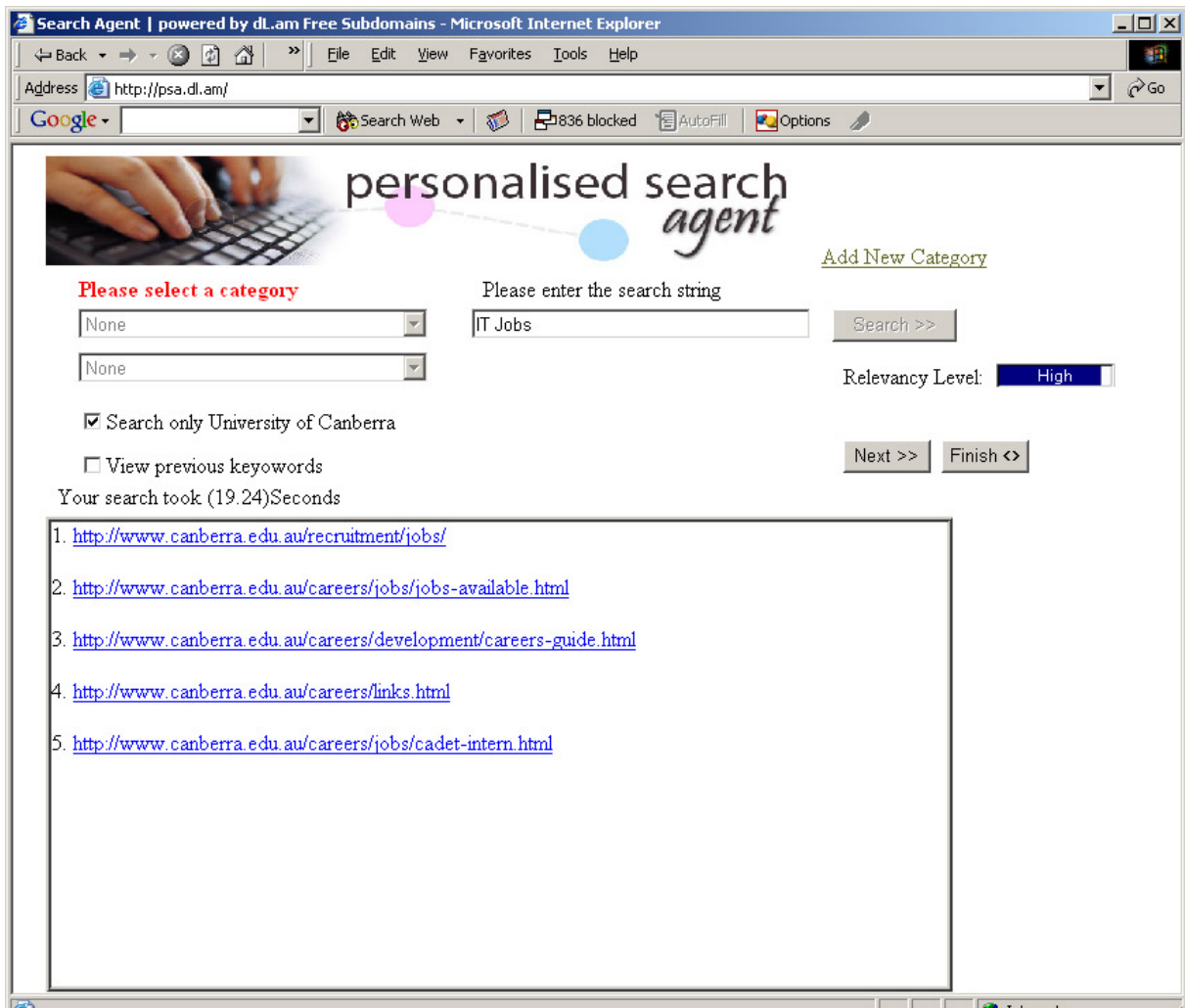
APPENDIX A SAMPLE INTERACTION SCREEN SHOTS OF PSA



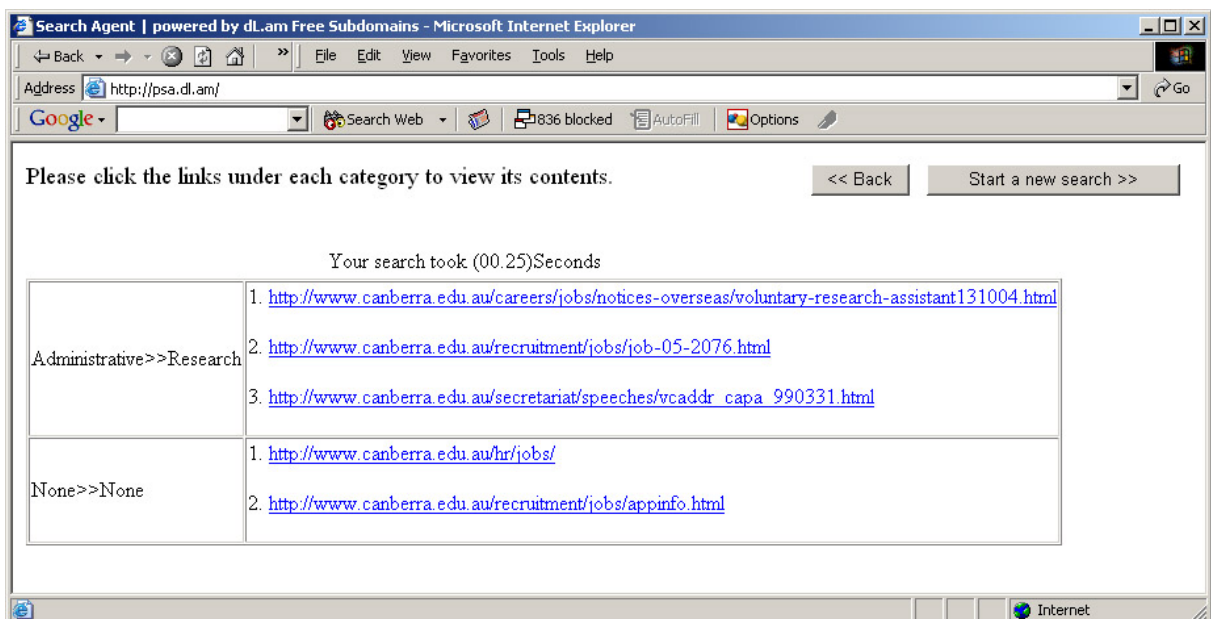
Home page for PSA prototype



Page for adding category or sub-category



Primary search result page of PSA prototype



Cached search result page of PSA prototype

Result - Microsoft Internet Explorer

Address http://vijaysagar.com/lgon/PSA/Result.aspx?got=yes&page=http://www.canberra.edu.au/recruitment/jobs/&found=true&cat=None&sub=None&key=IT%20Jobs&Time=4:20:15%

personalised search agent

You are visiting :- <http://www.canberra.edu.au/recruitment/jobs/>

UC HOME | SEARCH | KEY CONTACTS | SITE INDEX

UNIVERSITY OF CANBERRA, AUSTRALIA

Job Vacancies

HUMAN RESOURCES HOME | CORPORATE SERVICES DIVISION

ABOUT HR	STAFF SERVICES	EMPLOYMENT CONDITIONS	JOB VACANCIES	HEALTH & SAFETY	EQUITY	DEVELOPMENT & TRAINING	HR FORMS
----------	----------------	-----------------------	---------------	-----------------	--------	------------------------	----------

IN THIS SECTION:

- [Internal Job Vacancies](#)
- [External Job Vacancies](#)
- [Information for Applicants](#)

Positions Vacant

The following positions are currently vacant at the University:

- [ASSOCIATE PROFESSOR - MANAGEMENT](#) (Reference No. 05/2076)
Applications close 26 July 2005
- [LECTURER/SENIOR LECTURER - CULTURAL HERITAGE STUDIES](#) (Reference No. 05/2095)
Applications close 29 July 2005
- [LECTURER - COMMUNICATION STUDIES](#) (Reference No. 05/2096)
Applications close 29 July 2005
- [LECTURER/SENIOR LECTURER - HUMAN RESOURCE MANAGEMENT](#) (Reference No. 05/2104)
Applications close 12 August 2005
- [MARKETING MANAGER](#) (Reference No. 05/2105)

Done Unknown Zone (Mixed)

Page that opens individual URLs from the result.

APPENDIX B SOURCE CODE OF THE PSA PROTOTYPE

GUI.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace PSA
{
    /// <summary>
    /// Summary description for WebForm1.
    /// </summary>
    public class WebForm1 : System.Web.UI.Page
    {
        protected System.Web.UI.HtmlControls.HtmlGenericControl DIV1;
        protected System.Web.UI.HtmlControls.HtmlGenericControl DIV2;
        private SearchFilter searchFilter;
        protected System.Web.UI.WebControls.Button search_cmd;
        protected System.Web.UI.WebControls.Image Image1;
        protected System.Web.UI.WebControls.Label webcrawler_lbl;
        protected System.Web.UI.WebControls.Label lbl_CategoryName;
        protected System.Web.UI.WebControls.Label lblKeyword;
        protected System.Web.UI.WebControls.DropDownList cmb_Category;
        protected System.Web.UI.WebControls.TextBox keyword_txt;
        private string keyword;
        protected System.Web.UI.WebControls.Label lblSearchString;
        protected System.Web.UI.WebControls.Label lblNum;
        protected System.Web.UI.WebControls.Button cmd_Next;
        protected System.Web.UI.WebControls.Button cmd_Finish;
        protected System.Web.UI.WebControls.ListBox lst_Keyword;
        protected System.Web.UI.WebControls.DropDownList
cmb_subCategory;
        protected System.Web.UI.WebControls.CheckBox chk_UC;
        protected System.Web.UI.WebControls.CheckBox chk_History;
        protected System.Web.UI.WebControls.Image img_relevancyLevel;
        protected System.Web.UI.WebControls.Label lbl_relevancyLevel;
        protected System.Web.UI.WebControls.HyperLink HyperLink1;
        protected System.Web.UI.WebControls.Label lblTime;
        private DBConnection db;

        private void Page_Load(object sender, System.EventArgs e)
        {
            if (!(Page.IsPostBack))
            {
                db = new DBConnection();
                string sql;
                sql = "SELECT DISTINCT [Category] FROM PSA";
                DataSet dsCategory = db.selectQuery(sql);
            }
        }
    }
}
```

```

        cmb_Category.DataSource = dsCategory;
        cmb_Category.DataValueField = "Category";
        cmb_Category.DataBind();

        cmb_Category.Visible = false;
        cmb_subCategory.Visible = false;
        lbl_CategoryName.Visible = false;
        DIV1.Visible = false;

        lbl_CategoryName.ForeColor = Color.Black;
        lbl_CategoryName.Font.Bold = false;
        cmb_Category.SelectedValue = "None";
        cmb_Category_SelectedIndexChanged(sender, e);
    }
}

#region Web Form Designer generated code
override protected void OnInit(EventArgs e)
{
    //
    // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
    //
    InitializeComponent();
    base.OnInit(e);
}

/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.lst_Keyword.SelectedIndexChanged += new
System.EventHandler(this.lst_Keyword_SelectedIndexChanged);
    this.search_cmd.Click += new
System.EventHandler(this.search_cmd_Click);
    this.cmb_Category.SelectedIndexChanged += new
System.EventHandler(this.cmb_Category_SelectedIndexChanged);
    this.cmb_subCategory.SelectedIndexChanged += new
System.EventHandler(this.cmb_subCategory_SelectedIndexChanged);
    this.cmd_Next.Click += new
System.EventHandler(this.cmd_Next_Click);
    this.cmd_Finish.Click += new
System.EventHandler(this.cmd_Finish_Click);
    this.chk_History.CheckedChanged += new
System.EventHandler(this.chk_History_CheckedChanged);
    this.Load += new System.EventHandler(this.Page_Load);
}
#endregion

private void search_cmd_Click(object sender, System.EventArgs
e)
{
    DateTime startTime;
    TimeSpan totalTime;
    startTime = DateTime.Now;
    keyword = keyword_txt.Text ;
    db = new DBConnection();
    string sql = "SELECT COUNT(*) FROM PSA WHERE keyword = '"

```

```

+ keyword + "'";

        if (db.selectScalar(sql) > 0)
            try
            {
                Request.Params.Get("isNew").Equals("yes");
            }
            catch { Response.Redirect("Search.aspx?key=" +
keyword); }
        else
        {
            lbl_CategoryName.Visible = true;
            lbl_CategoryName.ForeColor = Color.Red;
            lbl_CategoryName.Font.Bold = true;
            cmb_Category.Visible = true;
            cmb_subCategory.Visible = true;
        }

//

        int num = 1;

        switch (num)
        {
            case 1:
                img_relevancyLevel.Visible = true;
                lbl_relevancyLevel.Visible = true;
                img_relevancyLevel.ImageUrl = "high.jpg";
                break;
            case 2:
                img_relevancyLevel.ImageUrl = "average.jpg";
                lbl_relevancyLevel.Visible = true;
                img_relevancyLevel.Visible = true;
                break;
            case 3:
                img_relevancyLevel.ImageUrl = "low.jpg";
                lbl_relevancyLevel.Visible = true;
                img_relevancyLevel.Visible = true;
                break;
            case 0:
                img_relevancyLevel.Visible = false;
                lbl_relevancyLevel.Visible = false;
                break;
        }

        if (lblSearchString.Text.Equals(keyword_txt.Text))
            num = int.Parse(lblNum.Text) + 1;

        lblSearchString.Text = keyword_txt.Text;
        lblNum.Text = num.ToString();

        if (num > 3)
        {
            lblSearchString.Text = "Nomore";
            lblNum.Text = "1";
        }
        else
        {

```

```

        searchFilter = new SearchFilter();
        string temp =
searchFilter.isInHistory(keyword_txt.Text);
        searchFilter.setKeyword(cmb_Category.SelectedValue,
cmb_subCategory.SelectedValue, this.keyword, num,chk_UC.Checked);
        string result = searchFilter.getResult();
        if (result.Equals("Sorry no result found") && num <
3)
            cmd_Next_Click(sender, e);
        else
        {
            DIV1.Visible = true;
            DIV1.InnerHtml = result;
            cmd_Finish.Visible = true;
            cmd_Next.Visible = true;

            cmb_Category.Enabled = false;
            cmb_subCategory.Enabled = false;
            search_cmd.Enabled = false;
            totalTime = DateTime.Now.Subtract(startTime);
            lblTime.Text = "Your search took ( " +
totalTime.ToString().Substring(6,5) + ")Seconds";
        }
    }

    private void cmb_Category_SelectedIndexChanged(object sender,
System.EventArgs e)
    {
        if (cmb_Category.Visible == false)
            DIV1.Visible = false;

        db = new DBConnection();
        string category, sql;
        DataSet dsSubCategory;

        category = cmb_Category.SelectedValue;
        sql = "SELECT DISTINCT [SubCategory] FROM PSA WHERE
[Category] = '" + category + "'";
        dsSubCategory = db.selectQuery(sql);
        cmb_subCategory.DataSource = dsSubCategory;
        cmb_subCategory.DataValueField = "SubCategory";
        cmb_subCategory.DataBind();

        cmb_subCategory.SelectedIndex = 0;
        cmb_subCategory_SelectedIndexChanged(sender, e);
    }

    private void cmb_subCategory_SelectedIndexChanged(object
sender, System.EventArgs e)
    {
        lblNum.Text = "0";

        if (cmb_subCategory.Visible == false)
            DIV1.Visible = false;

        db = new DBConnection();
        string category, subCategory, sql;
        DataSet dsKeyword;

        category = cmb_Category.SelectedValue;

```



```

        subCategory = cmb_subCategory.SelectedValue;
        sql = "SELECT DISTINCT [Keyword] FROM PSA WHERE
[Category] = '" + category + "' AND [SubCategory] = '" + subCategory + "'";
        dsKeyword = db.selectQuery(sql);
        lst_Keyword.DataSource = dsKeyword;
        lst_Keyword.DataValueField = "Keyword";
        lst_Keyword.DataBind();
        lst_Keyword.SelectedIndex = 0;
        lst_Keyword_SelectedIndexChanged(sender, e);
    }

    private void lst_Keyword_SelectedIndexChanged(object sender,
System.EventArgs e)
    {
        db = new DBConnection();
        string category, subCategory, keyword, sql, result = "",
str;
        string start =
"http://vijaysagar.com/ligon/PSA/guiResult.aspx?got=no&page=";
        string end = "&found=false";

        int numRecord;
        DataSet dsURL;

        category = cmb_Category.SelectedValue;
        subCategory = cmb_subCategory.SelectedValue;
        keyword = lst_Keyword.SelectedValue;
        sql = "SELECT [URL] FROM PSA WHERE [Category] = '" +
category + "' AND [SubCategory] = '" + subCategory + "' AND [Keyword] = '"
+ keyword + "' ORDER BY [Time] DESC";

        dsURL = db.selectQuery(sql);
        numRecord = dsURL.Tables[0].Rows.Count;
        end = end + "&cat=" + category + "&sub=" + subCategory +
"&key=" + keyword;
        for (int i = 0; i < numRecord; i ++)
        {
            str = dsURL.Tables[0].Rows[i][0].ToString();
            result = result + "<p>" + " " + (i + 1) + ". " +
"<a href=" + start + str + end + " target=\"_blank\">" + str + "</a></p>" +
"<p>";
        }

        if (lst_Keyword.Items.Count > 1)
            chk_History.Visible = true;
        else
        {
            chk_History.Visible = false;
            //DIV1.Visible = true;
            DIV1.InnerHtml = result;
        }
        //*****
        string key = "";
        try
        {
            key = Request.Params.Get("keyword");
            if (key.Equals("") == false)
            {
                keyword_txt.Text = key;
                cmb_Category.Visible = true;
                cmb_subCategory.Visible = true;
            }
        }
    }

```

```

        lbl_CategoryName.Visible = true;
//        search_cmd_Click(sender, e);
    }
}
catch{}
}

private void cmd_Next_Click(object sender, System.EventArgs e)
{
    keyword = keyword_txt.Text ;
    int num = 2;

    if (lblSearchString.Text.Equals(keyword_txt.Text))
        num = int.Parse(lblNum.Text) + 1;

    switch (num)
    {
        case 1:
            img_relevancyLevel.Visible = true;
            lbl_relevancyLevel.Visible = true;
            img_relevancyLevel.ImageUrl = "high.jpg";
            break;
        case 2:
            img_relevancyLevel.ImageUrl = "average.jpg";
            lbl_relevancyLevel.Visible = true;
            img_relevancyLevel.Visible = true;
            break;
        case 3:
            img_relevancyLevel.ImageUrl = "low.jpg";
            lbl_relevancyLevel.Visible = true;
            img_relevancyLevel.Visible = true;
            break;
        case 0:
            img_relevancyLevel.Visible = false;
            lbl_relevancyLevel.Visible = false;
            break;
    }

    lblSearchString.Text = keyword_txt.Text;
    lblNum.Text = num.ToString();

    if (num > 3)
    {
        lblSearchString.Text = "Nomore";
        lblNum.Text = "0";
        //*****Add text here*****
        DIV1.InnerHtml = "Please consider changing your
keyword";

        lblTime.Text = "";
        //*****
    }
    else
    {
        DateTime startTime;
        TimeSpan totalTime;
        startTime = DateTime.Now;
        searchFilter = new SearchFilter();
        string temp =
searchFilter.isInHistory(keyword_txt.Text);
        searchFilter.setKeyword(cmb_Category.SelectedValue,

```

```

cmb_subCategory.SelectedValue, this.keyword, num, chk_UC.Checked);
        string result = searchFilter.getResult();
        if (result.Equals("Sorry no result found") && num <
3)
            cmd_Next_Click(sender, e);
            DIV1.InnerHtml = result;
            totalTime = DateTime.Now.Subtract(startTime);
            lblTime.Text = "Your search took (" +
totalTime.ToString().Substring(6,5) + ")Seconds";
        }
    }

    private void cmd_Finish_Click(object sender, System.EventArgs
e)
    {
        DIV1.InnerHtml = " ";
        cmb_Category.Enabled = true;
        cmb_subCategory.Enabled = true;
        search_cmd.Enabled = true;
        cmd_Next.Visible = false;
        cmd_Finish.Visible = false;
        lblNum.Text = "0";
        lbl_relevancyLevel.Visible = false;
        img_relevancyLevel.Visible = false;
        Response.Redirect("./gui.aspx");
    }

    private void chk_History_CheckedChanged(object sender,
System.EventArgs e)
    {
        if (chk_History.Checked)
            lst_Keyword.Visible = true;
        else
            lst_Keyword.Visible = false;
    }
}
}
}

```

GUIResult.aspx.cs

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace PSA
{
    /// <summary>

```

```

/// Summary description for GUIResult.
/// </summary>
public class GUIResult : System.Web.UI.Page
{
    protected System.Web.UI.WebControls.Image Image1;
    protected System.Web.UI.WebControls.Label lblURL;
    private string page, found, got, key, cat, sub;
    protected System.Web.UI.WebControls.Label lblSession;
    protected System.Web.UI.WebControls.Button btnExit;
    protected System.Web.UI.WebControls.Label lblTotalTime;
    protected System.Web.UI.WebControls.Label lbl_TotalTime;
    private DBConnection db;

    private void Page_Load(object sender, System.EventArgs e)
    {
        // Put user code to initialize the page here
        got = Request.Params.Get("got");
        page = Request.Params.Get("page");
        if (got.Equals("no"))
        {
            found = Request.Params.Get("found");
            key = Request.Params.Get("key");
            cat = Request.Params.Get("cat");
            sub = Request.Params.Get("sub");
            if (found.Equals("false"))
                Response.Redirect("Result.aspx?got=yes&page="
+ page + "&found=true&cat=" + cat + "&sub=" + sub + "&key=" + key +
"&Time=" + DateTime.Now.ToLongTimeString());
        }
        else
        {
            lblURL.Text = "You are visiting :- " + page;
            lblSession.Text = "You came at :- " +
Request.Params.Get("Time");
        }
    }

    #region Web Form Designer generated code
    override protected void OnInit(EventArgs e)
    {
        //
        // CODEGEN: This call is required by the ASP.NET Web Form
Designer.
        //
        InitializeComponent();
        base.OnInit(e);
    }

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.btnExit.Click += new
System.EventHandler(this.btnExit_Click);
        this.Load += new System.EventHandler(this.Page_Load);
    }
}
#endregion

```

```

private void btnExit_Click(object sender, System.EventArgs e)
{
    lbl_TotalTime.Visible = true;
    DateTime startTime, time;
    TimeSpan endTime;
    string totalTime;

    startTime = DateTime.Parse(Request.Params.Get("Time"));
    key = Request.Params.Get("key");

    cat = Request.Params.Get("cat");
    sub = Request.Params.Get("sub");
    page = Request.Params.Get("page");

    endTime = DateTime.Now.Subtract(startTime);
    totalTime = endTime.ToString();
    totalTime = totalTime.Substring(0, 8);
    time = DateTime.Parse(totalTime);
    db = new DBConnection();
    string sql;
    sql = "SELECT COUNT(ID) FROM PSA WHERE [Category] = '" +
cat + "' AND [SubCategory] = '" + sub + "' AND [Keyword] = '" + key + "'
AND [URL] = '" + page + "'";

    if (db.selectScalar(sql) == 0)
        sql = "INSERT INTO [PSA] ([Category],
[SubCategory], [Keyword], [Time], [URL]) VALUES ('" + cat + "', '" + sub +
"', '" + key + "', #" + time.ToLongTimeString() + "#, '" + page + "')";
    else
    {
        sql = "SELECT [Time] FROM PSA WHERE [Category] = '"
+ cat + "' AND [SubCategory] = '" + sub + "' AND [Keyword] = '" + key + "'
AND [URL] = '" + page + "'";
        time =
DateTime.Parse(db.selectQuery(sql).Tables[0].Rows[0][0].ToString());
        time = time.Add(endTime);

        sql = "UPDATE [PSA] SET [Time] = #" +
time.ToLongTimeString() + "# WHERE [Category] = '" + cat + "' AND
[SubCategory] = '" + sub + "' AND [Keyword] = '" + key + "' AND [URL] = '"
+ page + "'";
    }
    db.insertORUpdate(sql);
    lblTotalTime.Text = time.ToLongTimeString() + " AND
endtime = " + endTime.ToString();
    btnExit.Visible = false;
}
}
}

```

Search.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace PSA
{
    /// <summary>
    /// Summary description for Search.
    /// </summary>
    public class Search : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.Button btnSearch;
        protected System.Web.UI.HtmlControls.HtmlGenericControl DIV1;
        protected System.Web.UI.WebControls.Label lbl_Heading;
        protected System.Web.UI.WebControls.Button cmd_back;
        protected System.Web.UI.WebControls.Button btnNewSearch;
        protected System.Web.UI.WebControls.Label lblTime;
        protected System.Web.UI.WebControls.TextBox txtKeyword;

        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
            if (!(Page.IsPostBack))
            {
                txtKeyword.Text = Request.Params.Get("Key");
                btnSearch_Click(sender, e);
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form
            Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.btnSearch.Click += new
System.EventHandler(this.btnSearch_Click);
            this.cmd_back.Click += new
System.EventHandler(this.cmd_back_Click);
            this.btnNewSearch.Click += new

```

```

System.EventHandler(this.btnNewSearch_Click);
        this.Load += new System.EventHandler(this.Page_Load);
    }
#endregion

private void btnSearch_Click(object sender, System.EventArgs e)
{
    DateTime startTime;
    TimeSpan totalTime;
    startTime = DateTime.Now;
    DBConnection db = new DBConnection();
    string category, subCategory, keyword, sql, result = "",
str;
    string start =
"http://vijaysagar.com/ligon/PSA/guiResult.aspx?got=no&page=";
    string end;

    int numRecord1;
    DataSet dsCategory;

    keyword = txtKeyword.Text;
    sql = "SELECT DISTINCT[Category] FROM PSA WHERE [Keyword]
= '" + keyword + "'"; // ORDER BY [Time] DESC";

    dsCategory = db.selectQuery(sql);

    numRecord1 = dsCategory.Tables[0].Rows.Count;
    //*****
    result = "<TABLE id=\"Table1\" cellSpacing=\"1\"
cellPadding=\"1\" border=\"1\">";
    for (int x = 0; x < numRecord1; x++)
    {
        int numRecord2;
        DataSet dsSubCategory;

        sql = "select DISTINCT[SubCategory] FROM PSA WHERE
[Category] = '" + dsCategory.Tables[0].Rows[x]["Category"].ToString() + "'
AND [Keyword] = '" + keyword + "'";
        dsSubCategory = db.selectQuery(sql);

        numRecord2 = dsSubCategory.Tables[0].Rows.Count;
        for (int y = 0; y < numRecord2; y++)
        {
            DataSet dsURL;
            int numRecord;
            sql = "select * FROM PSA WHERE [SubCategory]
= '" + dsSubCategory.Tables[0].Rows[y]["SubCategory"].ToString() + "' AND
[Category] = '" + dsCategory.Tables[0].Rows[x]["Category"].ToString() + "'
AND [Keyword] = '" + keyword + "'";
            dsURL = db.selectQuery(sql);
            numRecord = dsURL.Tables[0].Rows.Count;

            //*****
            end = "&found=false";
            category =
dsURL.Tables[0].Rows[0]["Category"].ToString();
            subCategory =
dsURL.Tables[0].Rows[0]["SubCategory"].ToString();

```

```

        end = end + "&cat=" + category +
"&sub=" + subCategory + "&key=" + keyword;
        result = result + "<TR><TD>" + category +
">>" + subCategory ;
//*****
        result = result + "</TD><TD>"; //<DIV
id=\"DIV1\" + y + "\"" + x + "\" dataFormatAs=\"html\" runat=\"server\"
ms_positioning=\"FlowLayout\">";
        for (int i = 0; i < numRecord; i ++)
        {

                str =
dsURL.Tables[0].Rows[i][ "URL"].ToString();
                result = result + "<p>" + " " + (i + 1)
+ ". " + "<a href=" + start + str + end + " target=\"_blank\">" + str +
"</a></p>" + "<p>";
                }
        result = result + "</TD></TR>";
    }
    }result = result + "</TABLE>";
    DIV1.InnerHtml = result;
    totalTime = DateTime.Now.Subtract(startTime);
    lblTime.Text = "Your search took (" +
totalTime.ToString().Substring(6,5) + ")Seconds";

    }

//     private void Button1_Click(object sender, System.EventArgs e)
//     {
//         Response.Redirect("Gui.aspx");
//     }

    private void btnNewSearch_Click(object sender, System.EventArgs
e)
    {
        Response.Redirect("Gui.aspx?isNew=yes&keyword=" +
txtKeyword.Text);
    }

    private void cmd_back_Click(object sender, System.EventArgs e)
    {
        Response.Redirect("Gui.aspx");
    }
}
}

```


AddCategory.aspx.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;

namespace PSA
{
    /// <summary>
    /// Summary description for AddCategory.
    /// </summary>
    public class AddCategory : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.TextBox txtSub;
        protected System.Web.UI.WebControls.DropDownList ddlCategory;
        protected System.Web.UI.WebControls.Button btnAddCategory;
        protected System.Web.UI.WebControls.Button btnAddSub;
        protected System.Web.UI.WebControls.Label Label1;
        protected System.Web.UI.WebControls.Label Label2;
        protected System.Web.UI.WebControls.Label Label3;
        protected System.Web.UI.WebControls.Label Label4;
        protected System.Web.UI.WebControls.Label Label5;
        protected System.Web.UI.WebControls.TextBox txtCategory;
        protected System.Web.UI.WebControls.HyperLink HyperLink1;
        protected System.Web.UI.WebControls.Label Label6;
        DBConnection db = new DBConnection();
        private void Page_Load(object sender, System.EventArgs e)
        {
            // Put user code to initialize the page here
            if (!(Page.IsPostBack))
            {
                string sql;
                sql = "SELECT DISTINCT [Category] FROM PSA";
                DataSet dsCategory = db.selectQuery(sql);
                ddlCategory.DataSource = dsCategory;
                ddlCategory.DataValueField = "Category";
                ddlCategory.DataBind();
            }
        }

        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: This call is required by the ASP.NET Web Form
            Designer.
            //
            InitializeComponent();
            base.OnInit(e);
        }

        /// <summary>
```

```

        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.btnAddCategory.Click += new
System.EventHandler(this.btnAddCategory_Click);
            this.btnAddSub.Click += new
System.EventHandler(this.btnAddSub_Click);
            this.Load += new System.EventHandler(this.Page_Load);

        }
        #endregion

        private void btnAddCategory_Click(object sender,
System.EventArgs e)
        {
            string sql;
            sql = "INSERT INTO PSA (Category) Values ('" +
txtCategory.Text + "')";
            db.insertORUpdate(sql);
        }

        private void btnAddSub_Click(object sender, System.EventArgs e)
        {
            string sql;
            sql = "INSERT INTO PSA (Category, subCategory) Values ('"
+ ddlCategory.SelectedValue + "', '" + txtSub.Text + "')";
            db.insertORUpdate(sql);
            //txtCategory.Text = sql;
        }
    }
}

```

DBConnection.cs

```

using System;
using System.Data;
using System.Data.OleDb;
using System.Web.UI.WebControls;

namespace PSA
{
    /// <summary>
    /// Summary description for DBConnection.
    /// </summary>
    public class DBConnection
    {
        public DBConnection()
        {
            myConnString = "Provider=Microsoft.Jet.OLEDB.4.0;DATA
Source= " + SOURCE;
            conn = new OleDbConnection(myConnString);
        }
    }
}

```

```

public void insertORUpdate(string sql)
{
    try
    {
        conn.Open();
        OleDbCommand cmd = new OleDbCommand(sql, conn);
        cmd.ExecuteNonQuery();
        conn.Close();
    }
    finally
    {
        if (conn != null)
            conn.Close();
    }
}

public DataSet selectQuery(string sql)
{
    try
    {
        conn.Open();
        dataAdapter = new OleDbDataAdapter(sql, conn);
        DataSet dataSet = new DataSet();
        dataAdapter.Fill(dataSet);
        dataAdapter.Dispose();
        conn.Close();
        return(dataSet);
    }
    finally
    {
        if (conn != null)
            conn.Close();
    }
}

public int selectScalar(string sql)
{
    try
    {
        conn.Open();
        OleDbCommand cmd = new OleDbCommand(sql, conn);
        int result =
Int32.Parse(cmd.ExecuteScalar().ToString());
        conn.Close();
        return result;
    }
    catch
    {
        return 1;
    }
    finally
    {
        if (conn != null)
            conn.Close();
    }
}

const string SOURCE =
"D:\\Clients\\vijaysagar.com\\vijaysagar.com\\Projects\\Service\\DBFolder\\"

```

```

PSA.mdb";
        private string myConnString;
        private OleDbConnection conn;
        private OleDbDataAdapter dataAdapter;
    }
}

```

SearchFilter.cs

```

using System;
using System.Text.RegularExpressions;

namespace PSA
{
    /// <summary>
    /// This class will do the core functionalities for PSA
    /// </summary>
    public class SearchFilter
    {
        //private Regex regex;
        //private Match match;
        private string keyword;
        private string googleSearchString, googleRawResult;
        private string aolSearchString, aolRawResult;
        private int count;
        private string result;
        private PreferenceFilter preferenceFilter;
        private NetworkInterface networkInterface;
        public string start =
"http://vijaysagar.com/ligon/PSA/guiResult.aspx?got=no&page=";
        public string end = "&found=false";

        public SearchFilter()
        {
            networkInterface = new NetworkInterface();
            preferenceFilter = new PreferenceFilter();
            keyword = "";
            result = "";
            count = 1;
        }

        public string isInHistory(string keyword)
        {
            return preferenceFilter.isInHistory(keyword);
        }

        public void setKeyword(string category, string sub, string key,
int index, bool domain)
        {
            if (category.Equals("None"))
                keyword = key;
            else if (key.Equals(""))
                keyword = sub;
            else

```

```

        {
            switch (index)
            {
                case 1:
                    keyword = "\" + sub + " " + key +
"\\"";
                    break;
                case 2:
                    keyword = "\" + key + "\"+\"" + sub + "\"";
                    break;
                case 3:
                    keyword = "\" + key + "\"";
                    break;
            }
        }
        end = end + "&cat=" + category + "&sub=" + sub + "&key="
+ key.Replace(" ", "+");
        if (domain)
            generateSearchStringForUC();
        else
            generateSearchString();
    }
    private void generateSearchString( )
    {
        googleSearchString =
"http://www.google.com.au/search?ie=UTF-8&oe=UTF-8&sourceid=deskbars&q=" +
keyword;
        networkInterface.setSearchString(googleSearchString);
        googleRawResult =
networkInterface.getResultFromSearchEngine();
        aolSearchString
="http://search.aol.com/aolcom/search?invocationType=topsearchbox.webhome&q
uery=" + keyword;
        networkInterface.setSearchString(aolSearchString);
        aolRawResult =
networkInterface.getResultFromSearchEngine();
    }

    private void generateSearchStringForUC( )
    {
        googleSearchString =
"http://www.google.com.au/search?sourceid=navclient&ie=UTF-
8&rls=GGLD,GGLD:2005-24,GGLD:en&q=site:www%2Ecanberra%2Eedu%2Eau+" +
keyword;
        networkInterface.setSearchString(googleSearchString);
        googleRawResult =
networkInterface.getResultFromSearchEngine();
        aolSearchString
="http://search.aol.com.au/search?hl=en&lr=lang_en&newwindow=1&c=aol-au&q="
+ keyword + "+site%3Ahttp%3A%2F%2Fwww.canberra.edu.au&btnG=SEARCH&meta=";
        networkInterface.setSearchString(aolSearchString);
        aolRawResult =
networkInterface.getResultFromSearchEngine();
    }

    //This method will be called from GUI for getting the results
    public string getResult()
    {
        googlePostSearchFilter();
        aolPostSearchFilter();
    }

```

```

        return result;
    }

    private void googlePostSearchFilter()
    {
        string temp="",str="";
        string str1;
        int index;
        try
        {
            for(int i =1; i<6; i++)
            {
                index = googleRawResult.IndexOf("<p
class=g><a href=");
                if (index < 0)
                    throw new Exception();

                temp = googleRawResult.Substring(index + 19);
                googleRawResult = temp;
                str = temp.Substring(0, temp.IndexOf(">"));
                str1 = str.Substring(1, str.Length-2);
                result = result + "<p>" + " " + i + ". " +
"<a href=" + start + str1 + end + " target=\"_blank\">" + str1 + "</a></p>"
+ "<p>";
                count = i + 1;
            }
        }
    }

    private void aolPostSearchFilter()
    {
        string str="",temp="";
        int index;
        try
        {
            for(int i = count; i< count + 5; i++)
            {
                index =
aolRawResult.IndexOf("target=\"_blank\" onmouseover=\"self.status=");
                if (index < 0)
                    throw new Exception();

                temp = aolRawResult.Substring(index + 42);
                str = temp;
                aolRawResult = str;
                str = str.Substring(0,str.IndexOf("; return
true;\" onmouseout=\"self.status="));
                //str1 = str;
                if(result.IndexOf(str) == -1)
                    result = result + "<p>" + " " + i + ".
" + "<a href=" + start + str + end + " target=\"_blank\">" + str +
"</a></p>" + "<p>";
                else
                    i--;
            }
        }
        // return result;
    }
    catch
    {

```

```

        if (result.Equals(""))
            result = "Sorry no result found";
        // return result;
    }
}
}

```

NetworkInterface.cs

```

using System;
using System.Net;
using System.IO;
namespace PSA
{
    /// <summary>
    /// This class interact with the internet and the search engines
    /// </summary>
    public class NetworkInterface
    {
        private Uri objURI;
        private WebRequest webRequest;
        private WebResponse webResponse;
        private Stream stream;
        private StreamReader streamReader;

        private string searchString;
        private string resultFromGoogle;
        public NetworkInterface()
        {
        }
        //This method will be called from SearchFilter class to pass
the search string
        public void setSearchString(String searchString)
        {
            this.searchString = searchString;
            connectWeb();
        }
        //This method opens a connection and pass the search string to
search engine
        private void connectWeb()
        {
            objURI = new Uri(searchString);
            webRequest = WebRequest.Create(objURI);
            webResponse = webRequest.GetResponse();
            stream = webResponse.GetResponseStream();
            streamReader = new StreamReader(stream);
            resultFromGoogle = streamReader.ReadToEnd();
        }
        //This method will be called from SearchFilter class to -
//get the output from Search Engine
        public string getResultFromSearchEngine()
        {
            return resultFromGoogle;
        }
    }
}

```

Appendix C

This chapter has been removed due to copyright restrictions.

This chapter is available as:

Ligon, G.L., Balachandran, B., and Sharma, Dharmendra. (2005) Personalised search on electronic information. 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, 14-16 September 2005, Melbourne. 35-41.

Links to this chapter:

Print	http://webpac.canberra.edu.au/record=b1255611~S4
Online subscribed content (UC community)	http://ezproxy.canberra.edu.au/login?url=http://link.springer.com/chapter/10.1007%2F11554028_6
Online general public	http://link.springer.com/chapter/10.1007%2F11554028_6
Research Repository	http://www.canberra.edu.au/researchrepository/items/9b6ce2e2-baa7-672d-70c5-9e83b8dc194f/1/
DOI	10.1007/11554028_6

Abstract

In the current world of Electronic Informatics, Info Search is more a relative term. General Search results have no relevance where people are looking for personalised, highly specific information. Generally, a user must revise a large number of uninteresting documents and consult several search engines before finding relevant information. Almost all search engines rely on text-based search algorithms and hence the probability of getting user desired information is heavily based on users ability to frame right keywords or search strings. This paper aims to address this problem by presenting a Personalized Search Agent(PSA) that generates effective keywords and also captures and organises different areas of interests of the user in a hierarchy that defines the user profile. The idea is to create a filter application, which applies a dual methodology of browsing and searching, that works between user and one or more commercial search engines specified by the user. This paper describes the design, architecture and the functional components of the PSA system and also describes how the system works thorough a scenario based approach.