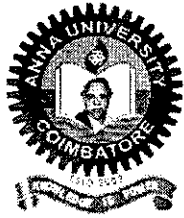


P-3623



**A CLASSIFICATION BASED PAGE RANKING
ALGORITHM FOR WEB SEARCH ENGINE**



PROJECT REPORT

Submitted by

MADHANGANESH .S

Reg.No: 0710108027

KAMALA KANNAN.C

Reg.No: 0710108303

In partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

**(An Autonomous Institution Affiliated to Anna University of
Technology, Coimbatore)**

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University, Coimbatore)

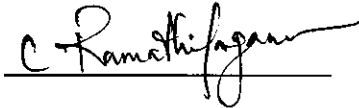
COIMBATORE - 641049

Department of Computer Science and Engineering

PROJECT WORK, April 2011

BONAFIDE CERTIFICATE

This is to certify that the project entitled “CLASSIFICATION BASED PAGE RANKING ALGORITHM FOR WEB SEARCH ENGINE”, the bonafide work of MADHANGANESH.S, KAMALAKANNAN.C who carried out the project work under my supervision.



[Mrs.C.RAMATHILAGAM, M.E.,]

Project Guide



[Mrs.S.Devaki,M.E.,]

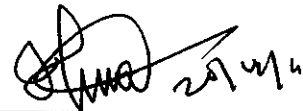
Head of the Department

The candidates with University Register Nos. 0710108027 & 0710108303 was examined by us in project viva-voce examination

held on 20-04-2011.



Internal Examiner



External Examiner

DECLARATION

We,

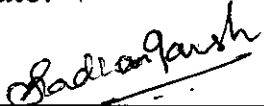
MADHANGANESH.S
KAMALA KANNAN.C

Reg.No:0710108027
Reg.No:0710108303

Hereby declare that the project entitled “**CLASSIFICATION BASED PAGE RANKING ALGORITHM FOR WEB SEARCH ENGINE**”, submitted in partial fulfillment to Anna University as the project work of Bachelor of Engineering (Computer Science and Engineering) degree, is record of original work done by us under the supervision and guidance of Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore.

Place: Coimbatore

Date: 19-04-2011



[MADHANGANESH.S]



[KAMALAKANNAN.C]

Project Guided by,



[Mrs.C.Ramathilagam, M.E.,]

Assistant Professor

Department of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore-641 049.

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Lord Almighty for enabling us to complete this project.

We express our profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.**, for giving this opportunity to pursue this course.

We would like to thank Dr.S.Ramachandran, **Ph.D.**, *Principal* for providing the necessary facilities to complete our thesis.

We take this opportunity to thank **Dr.S.Thangasamy Ph.D.**, *Dean, Research and Development*, for his precious suggestions. We also thank **Mrs.S.Devaki M.S.**, *HOD*, Department of Computer Science and Engineering, for her support and timely motivation.

We register our hearty appreciation to the Guide **Mrs.C.Ramathilagam, M.E.**, *Assistant Professor*, Department of Computer Science and Engineering, our Project advisor. We thank for her support, encouragement and ideas. We thank her for the countless hours she has spent with us, discussing everything from research to academic choices.

We would like to convey our honest thanks to all **Teaching** staff members and **Non Teaching** staffs of the department for their support. We would like to thank all our classmates who gave us a proper light moments and study breaks apart from extending some technical support whenever we needed them most.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	
	LIST OF FIGURES	
1	INTRODUCTION	3
	1.1 OVERVIEW	3
	1.2 OBJECTIVE	5
	1.3 BACKGROUND STUDY	6
	1.3.1 STUDY ON EXISTING SYSTEM	
	1.3.2.1 DRAWBACKS	
	1.3.2 PROJECT DESCRIPTION	
	1.3.3 FEASIBILITY STUDY	
2	SYSTEM ANALYSIS	11
	2.1 STUDY ON PROPOSED SYSTEM	11
	2.1.1 CLASSIFICATION SEARCH ARCHITECTURE	
	2.1.2 DEVELOPING SOLUTION STRATEGIES	
	2.2 SYSTEM SPECIFICATION	15
	2.2.1 HARDWARE SPECIFICATION	
	2.2.2 SOFTWARE SPECIFICATION	
	2.2.2.1 SOFTWARE DESCRIPTION	

3	SYSTEM DESIGN	24
	3.1 FUNDAMENTAL DESIGN CONCEPTS	24
	3.2 DESIGN CONCEPTS	26
4	MODULE DESCRIPTION	42
	4.1 CRAWLING	42
	4.2 DYNAMIC CATEGORY PARSING	43
	4.3 CLASSIFIED KNOWLEDGE BASE	44
	4.4 PAGE RANKING	45
5	SYSTEM IMPLEMENTATION AND TESTING	46
	5.1 SYSTEM IMPLEMENTATION	48
	5.2 SYSTEM TESTING	49
	5.2.1 TYPES OF TESTING	
	5.2.2 TEST CONSIDERATION FOLLOWED IN THIS PROJECT	
6	CONCLUSION	55
7	SCOPE FOR FURTHER DEVELOPMENT	56
8	APPENDIX	57
	8.1 SCREENS	57
9	REFERENCES	65

ABSTRACT

The existing search engines sometimes give unsatisfactory search result for lack of any categorization of search result. If there is some way to know the preference of user about the search result and rank pages according to that preference, the result will be more useful and accurate to the user. In this paper a web page ranking algorithm is being proposed based on classification of web pages. Classification does not bother about the meaning of the content of a web page.

Our Approach is to make a classification of web pages based on only syntax of the page and change the search engine interface to take the proper class of a page along with the query topic. The web page classification will be like Web page, Blog Page, Wiki pages, and RSS Feeds Pages etc.

For example classifiers are like Education, Art & Humanity, and Entertainment etc according to the webpage. The classification is done based on several properties of a web page which are not dependent on the meaning of its content.

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
D 1.1	Classification Search Diagrams	23
D 1.2	Classification Sequence Diagrams	24
D 1.3	System Flow Diagram	24
D 1.4	Web Crawling Model	40

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the last years, with the massive growth of the Web, we assisted to an explosion of information accessible to Internet users. Nevertheless, at the same time, it has become ever more critical for end users to explore this huge repository and find needed resources by simply following the hyperlink network as foreseen by Berners-Lee and Fischetti in 1999. Today, search engines constitute the most helpful tools for organizing information and extracting Knowledge from the Web.

However, it is not uncommon that even the most renowned search engines return result sets including many pages that are definitely useless for the user. This is mainly due to the fact that the very basic relevance criterions underlying their information retrieval strategies rely on the presence of query keywords within the returned pages. It is worth observing that statistical algorithms are applied to “tune” the result and, more importantly, approaches based on the concept of relevance feedback are used in order to maximize the satisfaction of user’s needs.

Imagine that while researching cats, you wish to know the average weight of a jaguar. If you decide to search AltaVista with the keywords *jaguar* and *weight*, you will find roughly 900 matching documents, but unfortunately you will not immediately see any that answer your query. The results are clouded with many pages concerning Jaguar automobiles, the Atari Jaguar home game system, and possibly even the Jacksonville Jaguars football team. Of those 900, we have found that the highest document on the list that

contains the information we want is document 41. (males average between 125-250 pounds.) Can we somehow tell a search engine (such as AltaVista) that our search using these keywords should be restricted to documents concerning zoology, or even just to science? One approach to restricting our search is to use a directory such as Yahoo!; unfortunately these (being manually generated) tend to cover only a small portion of the Web.

There is no doubt that the user would be able to easily decide which Results are really of interest by looking, for example, at the two-line excerpt of the Web page presented in the displayed list or by quickly examining each page. Anyway, the presence of unwanted pages in the result set would force him or her to perform a post processing on retrieved information to discard unneeded ones. Even though several automatic techniques have been recently proposed, result refinement remains a time-waste and click-expensive process, which is even more critical when the result set has to be processed by automatic software agents.

However, when the query was sent to the search Engine logic, these hidden details were lost. The search logic usually tries to recover this information by exploiting many text-matching techniques (such as the number of occurrences and distance among terms). Nevertheless, traditional search engines do not have the necessary infrastructure for exploiting relation-based information that belongs to the semantic annotations for a Web page.

1.2 OBJECTIVE

With the tremendous growth of information available to end users through the Web, search engines come to play ever a more critical role. Nevertheless, because of their general-purpose approach, it is always less uncommon that obtained result sets provide a burden of useless pages.

The next-generation Web architecture, represented by the Semantic Web, provides the layered architecture possibly allowing overcoming this limitation. Several search engines have been proposed, which allow increasing information retrieval accuracy by exploiting a key content of Semantic Web resources, that is, relations.

However, in order to rank results, most of the existing solutions need to work on the whole annotated knowledge base. In this paper, we propose a relation-based page rank algorithm to be used in conjunction with Semantic Web search engines that simply relies on information that could be extracted from user queries and on annotated resources. Relevance is measured as the probability that a retrieved resource actually contains those relations whose existence was assumed by the user at the time of query definition.

1.3 BACKGROUND STUDY

1.3.1 Study ON Existing System

Swoogle is a search engine for Semantic Web ontologies, documents, terms and data published on the Web. Swoogle employs a system of crawlers to discover RDF documents and HTML documents with embedded RDF content. Swoogle reasons about these documents and their constituent parts (e.g., terms and triples) and records and indexes meaningful metadata about them in its database.

Swoogle provides services to human users through a browser interface and to software agents via RESTful web services. Several techniques are used to rank query results inspired by the PageRank algorithm developed at Google but adapted to the semantics and use patterns found in semantic web documents.

Web query topic classification/categorization is a problem in information science. The task is to assign a Web search query to one or more predefined categories, based on its topics. The importance of query classification is underscored by many services provided by Web search. A direct application is to provide better search result pages for users with interests of different categories. For example, the users issuing a Web query “apple” might expect to see Web pages related to the fruit apple, or they may prefer to see products or news related to the computer company. Online advertisement services can rely on the query classification results to promote different products more accurately. Search result pages can be grouped according to the categories predicted by a query classification algorithm. However, the computation of query classification is non-trivial. Different from the document classification tasks, queries submitted by Web search users are usually short and ambiguous; also the meanings of the queries are evolving over time. Therefore, query topic

classification is much more difficult than traditional document classification tasks

1.3.1.1 DRAWBACKS

Commonly used searching methodologies Mäkelä describes five mainly used methodologies:

- RDF Path Traversal - traversing the net formed by the RDF data format.
- Keyword to Concept Mapping
- Graph Patterns - used to formulate patterns for locating interesting connecting paths between resources. Also commonly used in data visualization.
- Logics - by using inference based on OWL
- Fuzzy Concepts, Fuzzy Relations, Fuzzy Logics

1.3.2 PROJECT DESCRIPTION

1.3.2.1 CLASSIFICATION OF WEBPAGES

Generally an Internet surfer does not bother to go through more than 10 to 20 pages shown by a search engine. So the web page ranking should concentrate very much for giving higher rank to the relevant pages. In spite of all sophistication of the existing search engine, sometimes they do not give satisfactory result. The reason is that most of the time a surfer wants a particular type of page like an index page to get the links to good web pages or an article to know details about a topic, What is lacking in the existing search engines is a proper classification of the search pages and ranking according to that.

The advanced search options of search engines take very raw input like link to or from a particular website, or mandatory portion of a query etc, which are easier to use for a search engine software but difficult to interpret and use for a layman or non computer professional person. A lot of work has been done on web classification but that are based on semantics of the content of pages. In the classification is based on co-citations among web pages. But the existing classifiers are like Education, Art & Humanity, and Entertainment etc.

This type of classification is of not much help to web page ranking for most of the search strings. For example if a search topic like "Human Computer Interaction" is given, it is easy to guess that education related pages are wanted; there is no need of using any extra knowledge to derive the user's demand for the proper class of pages. In the user's intention is felt by identifying and separately analyzing the fuzzy portion of a query. The changed interface of a search engine in this case will not be much user friendly. The approaches taken

in or do not take the user preference explicitly. So the ranking has no control over a particular type of page.

1.3.3 FEASIBILITY STUDY

Feasibility study is a test of the system proposal according to the workability impact of the organization, ability to meet user's needs and effective use of resources. The feasibility study is to serve as a decision document: it must satisfy the following factors:

- 1) User demonstrable needs
- 2) Problem worth solving
- 3) Method of solving problem.

❖ Economical Feasibility:

Economic feasibility is the most frequently used method of evaluating the effectiveness of a candidate system. The procedure is to determine the savings and benefits from the candidate system and compare the costs. If the benefits outweigh the costs then it is decided to go ahead with the project. Otherwise, further justification or alternations in the proposed system will have to be made if it to have a chance of being approved. It is an on-going effort that improves the accuracy at each phase of the system life cycle.

In the economic feasibility study, the following points were found out:

- The automated system will cost bit as the initial expenditure.
- Maintenance also involves some investment in terms of money.

Once the computerized system is installed, it can cater to the needs of the customer and the business without much of manual work, which is more cost-effective for the management. In the sense, the automated system is economically feasible.

❖ **Technical Feasibility:**

Technical feasibility centers on the existing system and to what extent it can support the proposed system. It involves financial considerations to accommodate technical enhancements. If the budget is a serious constraint, then the project will be judged not feasible. Now considering the proposed system, and company planning to implement

So the owner having realized the advantages, benefits and economic feasibility of the new system is ready to afford the extra expense that may arise for the satisfaction of all the hardware and software requirements.

Operational Feasibility:

People are inherently resistant to change and computers have been known to facilitate change. An estimate should be made how strong a reaction the general public is likely to have towards a new computer system. It is common knowledge that computer installations have a lot to do with the turnover transfer retaining and changes to employee job status. Therefore it is understandable that the introduction of the candidate system requires special effort to educate and train the staff on a new way of conducting business. But since ultimately the introduction of a new system will only reduce the staff's workload, staff's may have no objection to install a computerized system and of course will be eager to extend their co-operation.



P-3623

❖ **Behavioral Feasibility:**

Behavioral feasibility deals with how the developed software behaves in different scenarios when deployed. It is also a very important part in the different stages of software development.

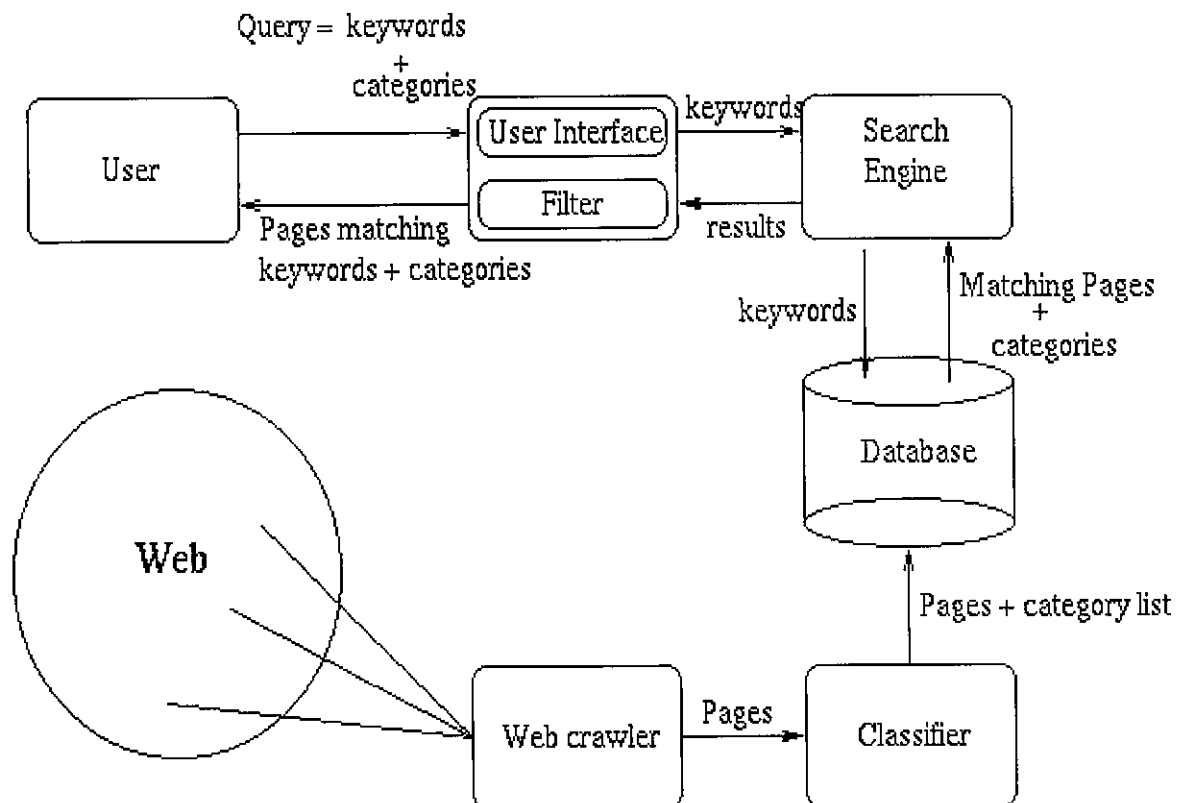
CHAPTER 2

SYSTEM ANALYSIS

2.1 STUDY ON PROPOSED SYSTEM

We now assemble the various steps illustrated in the previous sections to present the overall ranking methodology. The user starts defining query keywords and concepts. The search engine logic accesses the Web page database, constructs the initial result set including all those pages that contain queried keywords and concepts, and computes the query subgraph.

2.1.1 CLASSIFICATION SEARCH ARCHITECTURE:



We propose the above architecture for better search using automatic classification. The user specifies not only the keywords for a syntactic search, but also one or more categories about which he is interested. The search engine first retrieves the documents matching the keywords, then filters them by comparing their pre-computed categories against those chosen by the user.

The filtering is a cutoff based process where a page matches the user's categories if any of them occurs within the first k categories of the ordered list assigned by the classifier to the page. The default value of k will be a small number which is a good compromise between precision and recall. Figure 2 gives an overview of the search architecture. One could use classification to aid the user's search in another way. For a given set of keywords, the search engine could return a list of categories under which the pages fall. This would let the user prune away unwanted pages without browsing through a long list.

It could also make him aware of the kinds of ambiguities present in his keywords and prompt him to refine his search. To help the user in this, one could supplement the categories with keywords that identify the pages with their categories.

2.1.2 Developing Solution Strategies

To address the problem of privacy-preserving management of digital identity attributes in domains with heterogeneous name spaces, we propose a privacy-preserving multi-factor identity attribute verification protocol supporting a matching technique based on look-up tables, dictionaries, and ontology mapping techniques to match cloud service providers and client's vocabularies.

The protocol uses an aggregate zero knowledge proofs of knowledge cryptographic protocol to allow clients to prove with a single interactive proof the knowledge of multiple identity attributes without the need to provide them in clear.

2.2. SYSTEM SPECIFICATION

2.2.1 HARDWARE SPECIFICATION

Processor	:	Intel Pentium P4 1.7 GHZ
RAM	:	512 MB
Hard Disk Drive	:	80 GB

2.2.2 SOFTWARE SPECIFICATION

Operating System	:	Windows 7
IDE used	:	Visual Studio .Net Framework 2010
Web Technologies	:	ASP.NET
Language Used	:	Visual C#

2.2.2.1 SOFTWARE DESCRIPTION

VISUAL STUDIO .NET FRAME WORK 4.0

The Microsoft .NET Framework is a software framework that can be installed on computers running Microsoft Windows operating systems. It includes a large library of coded solutions to common programming problems and a virtual machine that manages the execution of programs written specifically for the framework. The .NET Framework is a Microsoft offering and is intended to be used by most new applications created for the Windows platform. The framework's Base Class Library provides a large range of features including user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers, who combine it with their own code to produce applications. Programs written for the .NET Framework execute in a software environment that manages the program's runtime requirements. Also part of the .NET Framework, this runtime environment is known as the Common Language Runtime (CLR). The CLR provides the appearance of an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program.

WINDOWS PRESENTATION FOUNDATION:

This subsystem is a part of .NET Framework 3.0. The Windows Presentation Foundation (or WPF) is a graphical subsystem for rendering user interfaces in Windows-based applications. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0. Designed to remove dependencies on the aging GDI subsystem, WPF is built on DirectX, which provides hardware acceleration and enables modern UI features like transparency, gradients, and transforms. WPF provides a consistent programming model for building applications and provides a clear separation between the user interface and the business logic.

WPF also offers a new markup language, known as XAML, which is an alternative means for defining UI elements and relationships with other UI elements. A WPF application can be deployed on the desktop or hosted in a web browser. It also enables rich control, design, and development of the visual aspects of Windows programs.

It aims to unify a number of application services: user interface, 2D and 3D drawing, fixed and adaptive documents, advanced typography, vector graphics, raster graphics, animation, data binding, audio and video. Microsoft Silverlight is a web-based subset of WPF that enables Flash-like web and mobile applications with the same programming model as .NET applications.

ASP.NET

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code

your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic and C#. These languages enable you to develop ASP.NET applications that benefit from the common language runtime, type safety, inheritance, and so on.

Visual Web Developer is a full-featured development environment for creating ASP.NET Web applications. Visual Web Developer offers you the following features:

- **Web page designs** A powerful Web page editor that includes WYSIWYG editing and an HTML editing mode with IntelliSense and validation.
- **Page design features** Consistent site layout with master pages and consistent page appearance with themes and skins.
- **Code editing** A code editor that enables you to write code for your dynamic Web pages in Visual Basic or C#. The code editor includes syntax coloration and IntelliSense.
- **Testing and debugging** A local Web server for testing and a debugger that helps you find errors in your programs.
- **Deployment** Tools to automate typical tasks for deploying a Web application to a hosting server or a hosting provider.

VISUAL C#:

C# (pronounced "see sharp") is a multi-paradigm programming language encompassing imperative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma (ECMA-334) and ISO (ISO/IEC 23270). C# is one of the programming languages designed for the Common Language Infrastructure. C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Its development team is led by Anders Hejlsberg. The most recent version is C# 3.0, which was released in conjunction with the .NET Framework 3.5 in 2007. The next proposed version, 4.0, is in development.

FEATURES

By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN.

Some notable distinguishing features of C# are:

- There are no global variables or functions. All methods and members must be declared within classes. Static members of public classes can substitute for global variables and functions.

- Local variables cannot shadow variables of the enclosing block, unlike C and C++. Variable shadowing is often considered confusing by C++ texts.
- C# supports a strict Boolean datatype, `bool`. Statements that take conditions, such as `while` and `if`, require an expression of a type that implements the `true` operator, such as the `boolean` type.
- While C++ also has a `boolean` type, it can be freely converted to and from integers, and expressions such as `if(a)` require only that `a` is convertible to `bool`, allowing `a` to be an `int`, or a pointer. C# disallows this "integer meaning true or false" approach on the grounds that forcing programmers to use expressions that return exactly `bool` can prevent certain types of common programming mistakes in C or C++ such as `if (a = b)` (use of assignment `=` instead of equality `==`).
- In C#, memory address pointers can only be used within blocks specifically marked as `unsafe`, and programs with `unsafe` code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined null value; it is impossible to obtain a reference to a "dead" object (one which has been garbage collected), or to a random block of memory.
- An `unsafe` pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack. Code that is not marked as `unsafe` can still store and manipulate pointers through the `System.IntPtr` type, but it cannot dereference them.
- Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem of memory leaks by freeing the programmer of responsibility for releasing memory which is no longer needed.

- In addition to the try...catch construct to handle exceptions, C# has a try...finally construct to guarantee execution of the code in the finally block.
- Multiple inheritance is not supported, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication and simplify architectural requirements throughout CLI.
- C# is more typesafe than C++. The only implicit conversions by default are those which are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime. There are no implicit conversions between booleans and integers, nor between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type). Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default.
- Enumeration members are placed in their own scope.
- C# provides properties as syntactic sugar for a common pattern in which a pair of methods, accessor (getter) and mutator (setter) encapsulate operations on a single attribute of a class.
- C# currently (as of 3 June 2008) has 77 reserved words.
- C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime,

or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN. With Visual developers can build solutions for the broadest range of clients, including Windows, the Web, and mobile or embedded devices. Using this elegant programming language and tool, developers can leverage their existing C++ and Java-language skills and knowledge to be successful in the .NET environment.

- C# is more type safe than C++. The only implicit conversions by default are those which are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime.
- There are no implicit conversions between booleans and integers, or between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type).
- Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors and conversion operators, which are both implicit by default.
- In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references. Managed memory cannot be explicitly freed; instead, it is automatically garbage collected.

DYNAMIC SUPPORT

Visual C# 2010 provides support for late binding to dynamic types by introducing a new type, `dynamic`. This addition enables many new scenarios, including simplified access to COM APIs such as the Office Automation APIs, to dynamic APIs such as Iron Python libraries, and to the HTML Document Object Model (DOM).

CHAPTER 3

SYSTEM DESIGN

3.1 FUNDAMENTAL DESIGN CONCEPTS

System design sits in the technical kernel of software engineering and applied science regardless of the software process model that is used. Beginning once the software requirements have been analyzed and specified, tests that are required in the building and verifying the software is done. Each activity transforms information in a number that ultimately results in validated computer software.

There are mainly three characteristics that serve as guide for evaluation of good design,

- The design must be implement all of explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.
- The design must be readable, understandable guide for those who generate code and for those who test and subsequently support the software.
- The design should provide a complete picture of software, addressing the data, its functional and behavioral domains from the implementation perspective.

System design is thus a process of planning a new system or to replace or the complement of the existing system. The design based on the limitations of the existing system and the requirements specification gathered in the phase of system analysis.

Input design is the process of converting the user-oriented description of the computer based business information into program-oriented specification. The goal of designing input data is to make the automation as easy and free from errors as possible.

Logical Design of the system is performed where its features are described, procedures that meet the system requirements are formed and a detailed specification of the new system is provided

Architectural Design of the system includes identification of software components, decoupling and decomposing them into processing modules, conceptual data structures and specifying relationship among the components.

Detailed Design is concerned with the methods involved in packaging of processing modules and implementation of processing algorithms, data structure and interconnection among modules and data structure.

External Design of software involves conceiving, planning and specifying the externally observable characteristics of the software product. The external design begins in the analysis phase and continues till the design phase.

As per the design phase the following designs had to be implemented, each of these design were processed separately keeping in mind all the requirements, constraints and conditions. A step-by-step process was required to perform the design.

Process Design is the design of the process to be done; it is the designing that leads to the coding. Here the conditions and the constraints given in the system are to be considered. Accordingly the designing is to be done and processed.

3.2 DESIGN CONCEPTS

CLASS DIAGRAM

The class diagram is the main building block in object oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. The classes in a class diagram represent both the main objects and or interactions in the application and the objects to be programmed. In the class diagram these classes are represented with boxes which contain three parts:

A class with three sections.

- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

In the system design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

In order to further describe the behavior of systems, these class diagrams can be complemented by state diagram or UML state machine. Also instead of class diagrams Object role modeling can be used if you just want to model the classes and their relationships.

Instance Level Relationships

External links

A Link is the basic relationship among objects. It is represented as a line connecting two or more object boxes. It can be shown on an object diagram or class diagram. A link is an instance of an association. In other words, it creates a relationship between two classes.

Association

An Association represents a family of links. Binary associations (with two ends) are normally represented as a line, with each end connected to a class box. Higher order associations can be drawn with more than two ends. In such cases, the ends are connected to a central diamond.

An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties. There are five different types of association. Bi-directional and uni-directional associations are the most common ones. For instance, a flight class is associated with a plane class bi-directionally. Associations can only be shown on class diagrams. Association represents the static relationship shared among the objects of two classes. Example: "department offers courses", is an association relation.

Aggregation

Aggregation is a variant of the "has a" or association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As a type of association, an aggregation can

be named and have the same adornments that an association can. However, an aggregation may not involve more than two classes.

Aggregation can occur when a class is a collection or container of other classes, but where the contained classes do not have a strong life cycle dependency on the container—essentially, if the container is destroyed, its contents are not.

In UML, it is graphically represented as a hollow diamond shape on the containing class end of the tree of lines that connect contained class(es) to the containing class.

Composition

Class diagram showing Composition between two classes at top and Aggregation between two classes at bottom

Composition is a stronger variant of the "owns a" or association relationship; composition is more specific than aggregation.

Composition usually has a strong life cycle dependency between instances of the container class and instances of the contained class(es): If the container is destroyed, normally every instance that it contains is destroyed as well. Note that a part can (where allowed) be removed from a composite before the composite is deleted, and thus not be deleted as part of the composite.

The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained class(es) to the containing class.

Differences between Composition and Aggregation

When attempting to represent real-world whole-part relationships, e.g., an engine is part of a car, the composition relationship is most appropriate. However, when representing a software or database relationship, e.g., car model engine ENG01 is part of a car model CM01, an aggregation relationship is best, as the engine, ENG01 may be also part of a different car model. Thus the aggregation relationship is often called "catalog" containment to distinguish it from composition's "physical" containment.

The whole of a composition must have a multiplicity of 0..1 or 1, indicating that a part must belong to only one whole; the part may have any multiplicity. For example, consider University and Department classes. A department belongs to only one university, so University has multiplicity 1 in the relationship. A university can (and will likely) have multiple departments, so Department has multiplicity 1..*.

Class Level Relationships

Generalization

The Generalization relationship indicates that one of the two related classes (the subtype) is considered to be a specialized form of the other (the super type) and supertype is considered as 'Generalization' of subtype. In practice, this means that any instance of the subtype is also an instance of the supertype. An exemplary tree of generalizations of this form is found in binomial nomenclature: human beings are a subtype of simian, which are a subtype of mammal, and so on. The relationship is most easily understood by the phrase 'an A is a B' (a human is a mammal, a mammal is an animal).

The UML graphical representation of a Generalization is a hollow triangle shape on the super type end of the line (or tree of lines) that connects it to one or more subtypes.

The generalization relationship is also known as the inheritance or "is a" relationship.

The supertype in the generalization relationship is also known as the "parent", superclass, base class, or base type.

The subtype in the specialization relationship is also known as the "child", subclass, derived class, derived type, inheriting class, or inheriting type.

Note that this relationship bears no resemblance to the biological parent/child relationship: the use of these terms is extremely common, but can be misleading.

Generalization-Specialization relationship

A is a type of B

Generalization can only be shown on class diagrams and on Use case diagrams.

Realization

In UML modeling, a realization relationship is a relationship between two model elements, in which one model element (the client) realizes (implements or executes) the behavior that the other model element (the supplier) specifies. A realization is indicated by a dashed line with an unfilled arrowhead towards the supplier.

A realization is a relationship between classes, interfaces, components, and packages that connects a client element with a supplier element. A realization relationship between classes and interfaces and between components and interfaces shows that the class realizes the operations offered by the interface.

Dependency

Dependency is a weaker form of relationship which indicates that one class depends on another because it uses it at some point of time. Dependency exists if a class is a parameter variable or local variable of a method of another class.

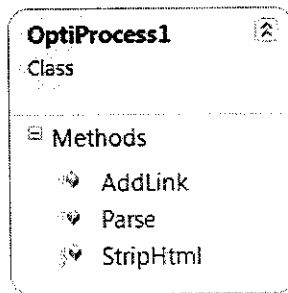
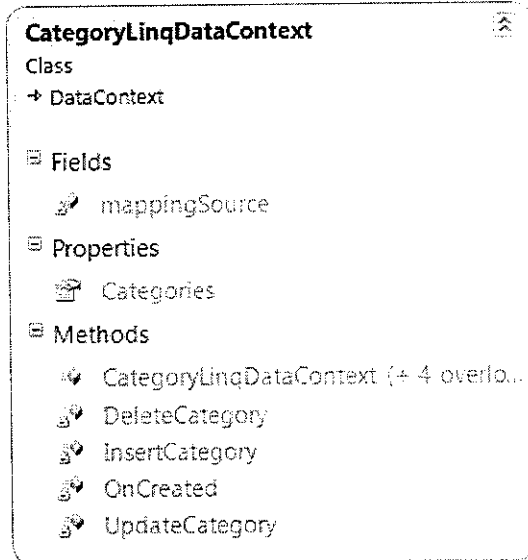
Multiplicity


The association relationship indicates that (at least) one of the two related classes makes reference to the other. In contrast with the generalization relationship, this is most easily understood through the phrase 'A has a B' (a mother cat has kittens, kittens have a mother cat).

The UML representation of an association is a line with an optional arrowhead indicating the role of the object(s) in the relationship, and an optional notation at each end indicating the multiplicity of instances of that entity (the number of objects that participate in the association).








FIGURES


D 1.1 CLASSIFICATION SEARCH DIAGRAMS




























WordsLinqDataContext 

Class
 → DataContext

- [-] Fields
 -  mappingSource
- [-] Properties
 -  NewsWords
- [-] Methods
 -  DeleteNewsWord
 -  InsertNewsWord
 -  OnCreated
 -  UpdateNewsWord
 -  WordsLinqDataContext (...)

NewsUI 

Class
 → Form


- [-] Fields :
- [-] Methods
 -  BlogOptimize
 -  button1_Click
 -  button2_Click
 -  button3_Click
 -  button4_Click
 -  button5_Click
 -  button6_Click
 -  button7_Click
 -  button8_Click
 -  button9_Click
 -  Dispose
 -  DragDropstopurl
 -  DragEnterColor
 -  DragEnterStop
 -  DragLeaveColor
 -  DragUrlCopy
 -  InitializeComponent
 -  NewsUI
 -  NewsUI_Load
 -  SecondOptimize
 -  SetText
 -  StartupOptimize
 -  timer1_Tick
 -  timer2_Tick
 -  toolStripContainer1...
- * Nested Types


NewsWord
Class


- Fields
 - _CatID
 - _Word
 - emptyChangin...
- Properties
 - CatID
 - Word
- Methods
 - NewsWord
 - OnCatIDChang...
 - OnCatIDChangi...
 - OnCreated
 - OnLoaded
 - OnValidate
 - OnWordChang...
 - OnWordChangi...
 - SendPropertyC...
 - SendPropertyC...
- Events
 - PropertyChang...
 - PropertyChangi...


Category
Class


- Fields
- Properties
 - CatID
 - CatLevel
 - CatMain
 - ContType
 - StopUrl
 - StopWord
- Methods
 - Category
 - OnCatIDChang...
 - OnCatIDChangi...
 - OnCatLevelCha...
 - OnCatLevelCha...
 - OnCatMainCha...
 - OnCatMainCha...
 - OnContTypeCh...
 - OnContTypeCh...
 - OnCreated
 - OnLoaded
 - OnStopUrlChan...
 - OnStopUrlChan...
 - OnStopWordC...
 - OnStopWordC...
 - OnValidate
 - SendPropertyC...
 - SendPropertyC...
- Events
 - PropertyChang...
 - PropertyChangi...




AllLinks 
Class

 Methods

-  RouteLinks

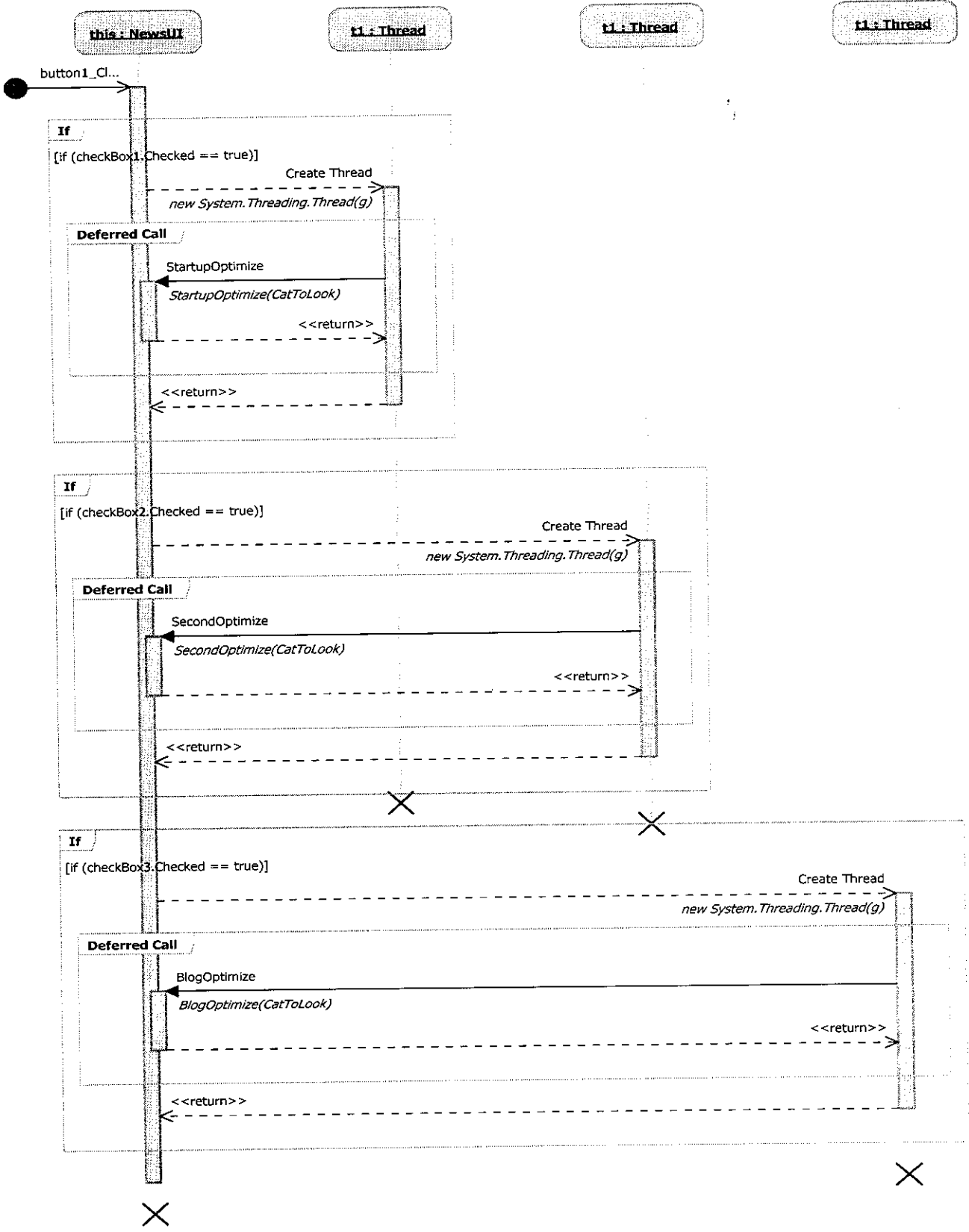
NewsProcess 
Class

 Methods

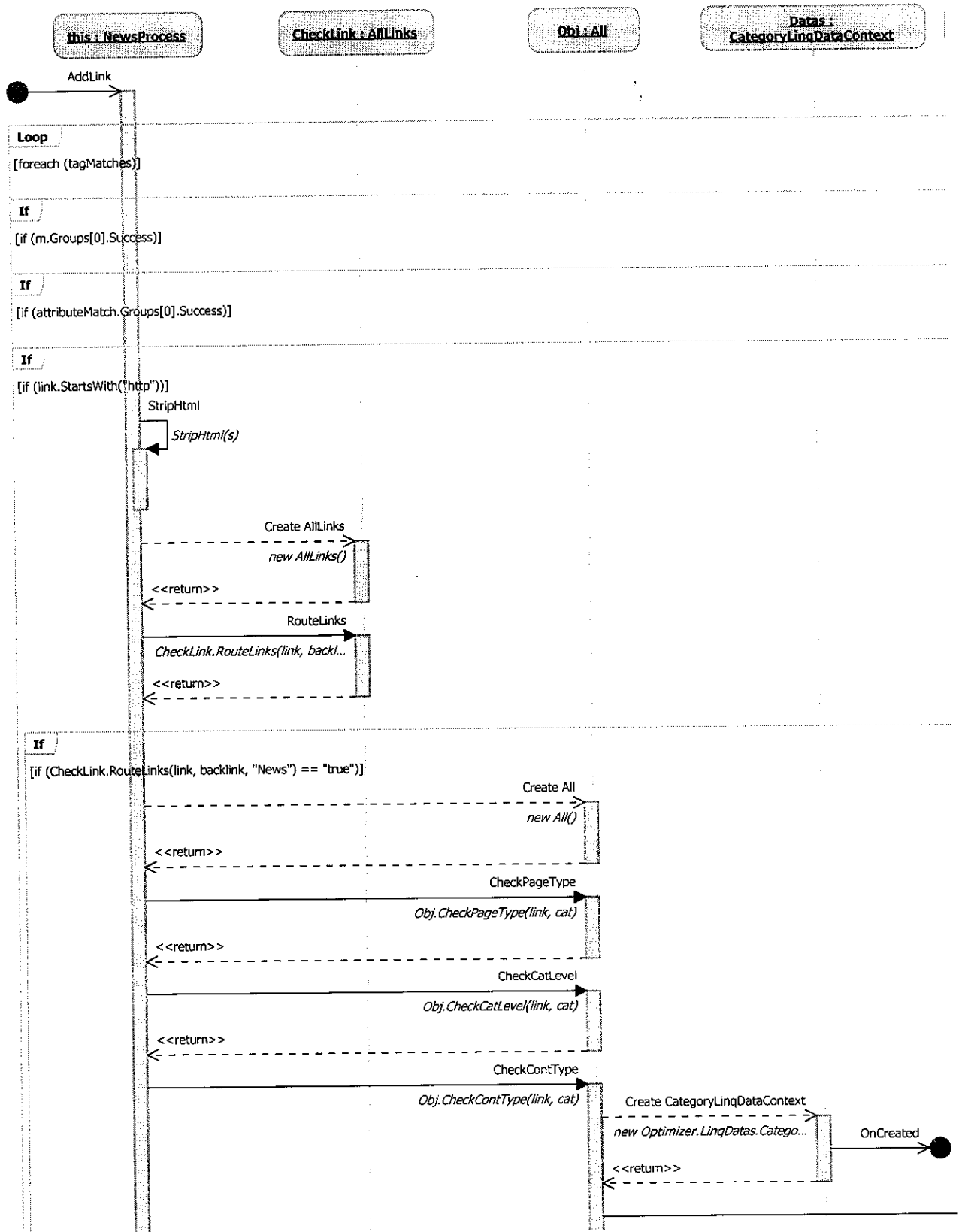
-  AddLink
-  Parse
-  StripHtml

D 1.2 SEQUENCE DIAGRAMS

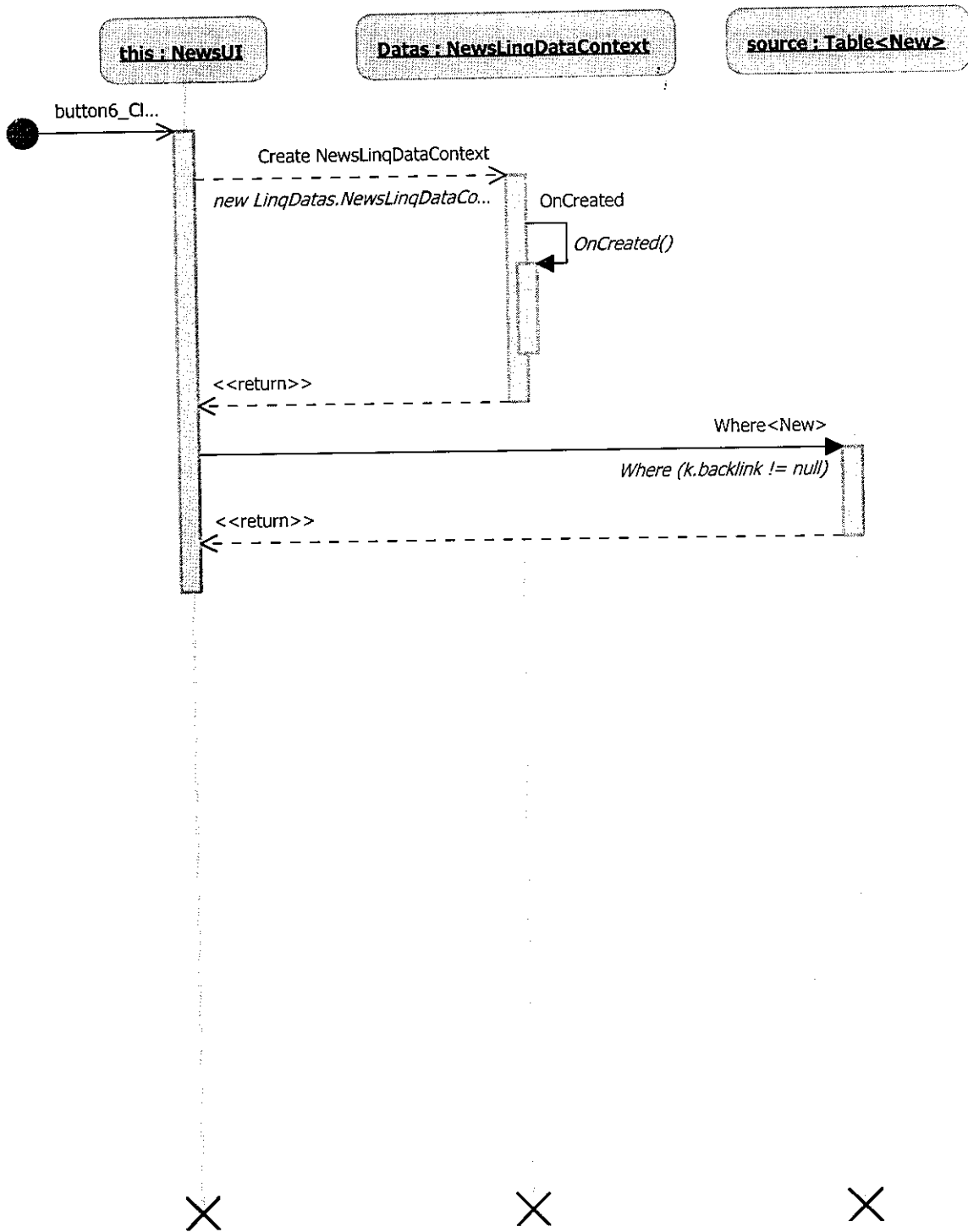
WEB CRAWLING



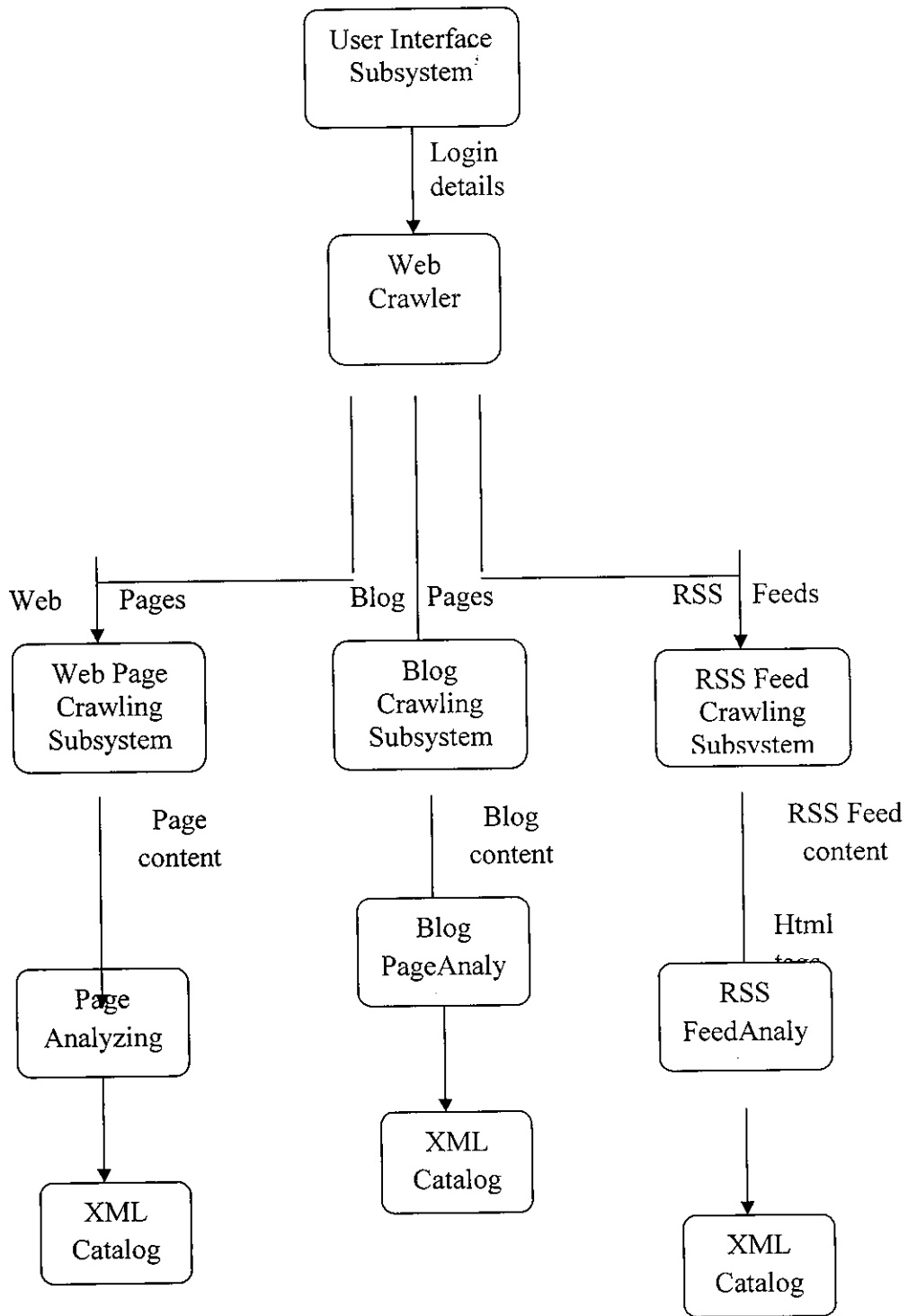
PARSING HTML PAGES



PACKING DATA TO XML

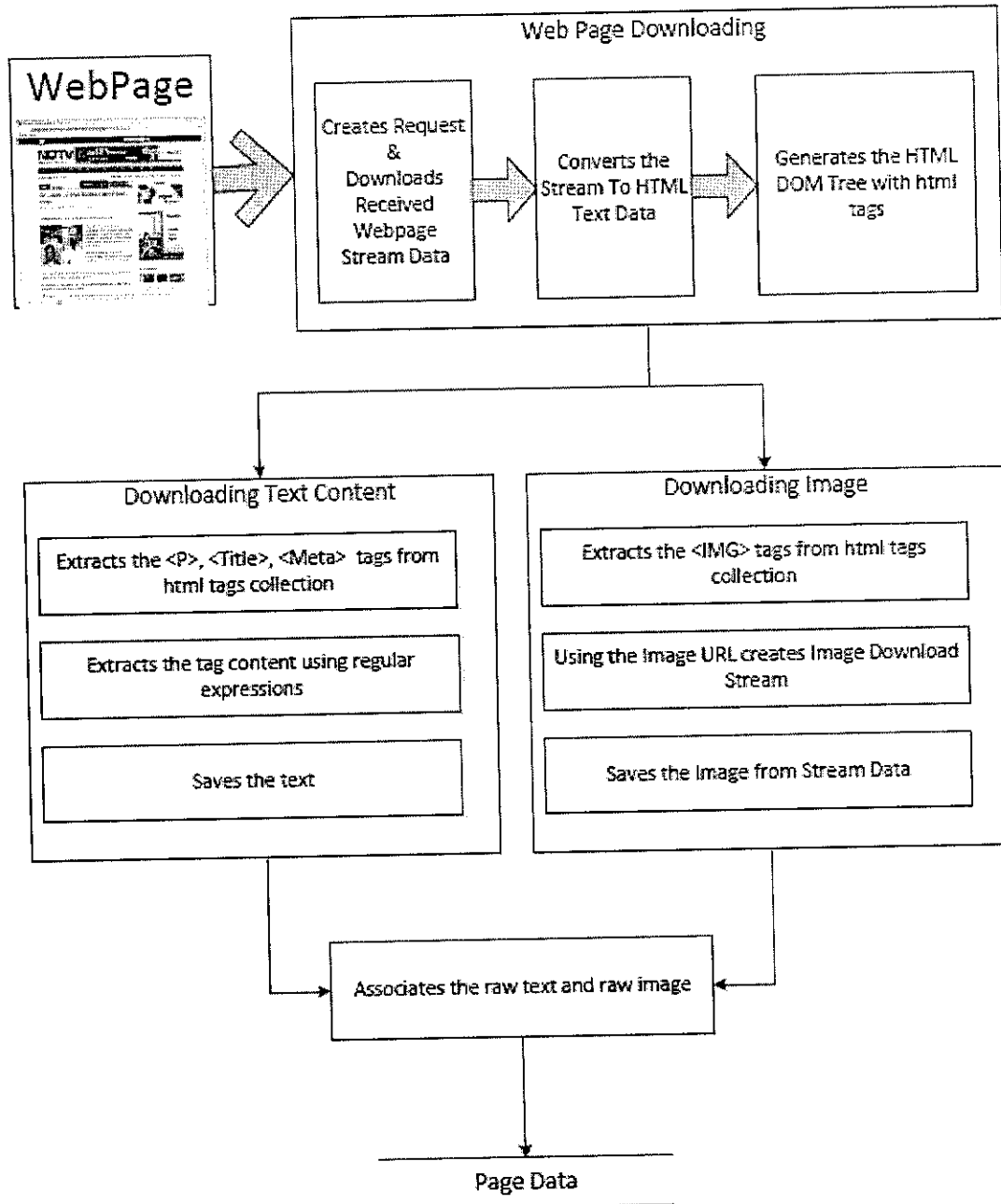


D 1.3 SYSTEM FLOW DIAGRAM



D 1.4 WEB CRAWLING MODEL

Web Crawling Block Model



CHAPTER 4

MODULE DESCRIPTION

MODULES:

1. CRAWLING
2. DYNAMIC CATEGORY PARSING
3. CLASSIFIED KNOWLEDGE BASE
4. PAGE RANKING

4.1 CRAWLING

A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner or in an orderly fashion. Other terms for Web crawlers are ants, automatic indexers, bots, Web spiders, Web robots, or—especially in the FOAF community—Web scutters.

This process is called Web crawling or spidering. Many sites, in particular search engines, use spidering as a means of providing up-to-date data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine that will index the downloaded pages to provide fast searches. Crawlers can also be used for automating maintenance tasks on a Web site, such as checking links or validating HTML code. Also, crawlers can be used to gather specific types of information from Web pages, such as harvesting e-mail addresses (usually for sending spam).

A Web crawler is one type of bot, or software agent. In general, it starts with a list of URLs to visit, called the seeds. As the crawler visits these URLs, it identifies all the hyperlinks in the page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited according to a set of policies.

The large volume implies that the crawler can only download a fraction of the Web pages within a given time, so it needs to prioritize its downloads.

The high rate of change implies that the pages might have already been updated or even deleted.

4.2 DYNAMIC CATEGORY PARSING

parser which is taking the document and splitting it into indexable text fragments. It usually works with different file formats, human languages and is preprocessing the text in maybe some fixed records and flow text. Linguistic algorithms (like stemmers - search for Porter Stemmer to get simple one) are also applied here.

Categorization is the process of associating a document with one or more subject categories. So the entry for a page on cross trainer shoes could go into Running, Manufacturing, Sports Medicine, or Rushkoff, Douglas! All of these are legitimate, depending on the context.

Search Engines Parser is enormously fast, 100% automatic search engine results extractor you were dreaming about for many times. Search Engines Parser can extract results from all search engines at the same time, parse titles, descriptions and links automatically. You can specify which search engine(s) to use and what kind of data to parse. Search Engines Parser can output results to screen, export to MySQL database and write to CSV file.

4.3 CLASSIFIED KNOWLEDGE BASE

After categorizing the links of the webpage, they are stored in the database, in turn it is classified into several fields. The classified fields are used to define the search logic of the search engine.

Cataloging and Classification come from libraries, where specialists enter the metadata (such as author, date, title and edition) for a document, apply subject categories to it, and place it into a class (such as a call number) for later retrieval. These tend to be used interchangeably with Categorization.

THE CLASSIFIED KNOWLEDGE BASE:

It has the following fields in the database. That is called Knowledge Base

- url
- Backlink
- Priority of target page
- Visited
- Pagetype
- Category main
- Category level
- Base url
- Date of crawl

4.4 PAGE RANKING (CLASSI – RANKING ALGORITHM)

In this algorithm, the classified categories are fetched. Categories are ranked first and then the links in the each category is ranked . The categories are ranked using the count of the links in each category, the maximum the count of the links , appears first. Then the links in each category is ranked using their corresponding priority number which is in the database.

In this section different parameters, selected for web page ranking, are discussed. The page ranking will be done by taking a weighted average of all or some of the parameters. The weight given to a particular parameter will depend upon the category of the page. In the proposed algorithm a single query may give different ranking to a page depending on the category of the page-which is not possible in any existing search engines. The algorithm is flexible in the sense that just by changing the weights the same algorithm provides ranking for different types of pages.

For a given user input query, a commercial search engine returns many pages of URLs, in an order determined by the underlying proprietary ranking algorithm. The quality of the returned results are largely evaluated on the URLs displayed in the very first page. The type of ranking problem in this study is sometimes referred to as dynamic ranking (or simply, just *ranking*), because the URLs are dynamically ranked (in real-time) according to the specific user input query. This is different from the query-independent static ranking.

CHAPTER 5

SYSTEM TESTING AND IMPLEMENTATION

After the successful study of requirement analysis the next step involved is the Design and Development phase that practically helps to build the project.

The methods that are applied during the development phase

- ❖ Software Design
- ❖ Code Generation
- ❖ Software Testing

The **Linear Sequential Model** or **Classic Life Cycle** or the **Waterfall Model** develops project. This is a sequential approach to software development that begins at the system level and progresses through analysis, design, coding and testing.

❖ **System / Information Engineering and Modeling**

Because software is always part of a larger system, work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software. System view is essential when software must interact with other elements such as hardware people and database.

◇ **Software requirements analysis**

Requirements is intensified and focused specially on software. To understand the nature of the program to be built, the software engineer must

understand the information domain for the software, as well as required function, behavior, performance and interface.

◇ **Design of the project**

The design process translates requirements into a representation of the software that can be accessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.

◇ **Code Generation**

The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.

After completing the design phase, code was generated using Visual Basic environment and the SQL Server 2000 was used to create the database. The server and the application were connected through ADO.Net concepts.

The purpose of code is to facilitate the identification and retrieval of items of information. Codes are built with the mutually exclusive features. They are used to give operational distractions and other information. Codes also show interrelationship among different items. Codes are used for identifying, accessing, sorting and matching records. The code ensures that only one value of code with single meaning is correctly applied to give entity or attribute as described in various ways. Codes can also be designed in a manner easily understood and applied by the user.

The coding standards used in the project are as follows:

1. All variable names are kept in such a way that it represents the flow/function it is serving.

2. All functions are named such that it represents the function it is performing.

5.1 SYSTEM IMPLEMENTATION

To evaluate the feasibility of the proposed approach, we first constructed a controlled web crawling environment.

Focused Crawling:

The proposed WebCrawler create a web request for webpage's one by one, once the WebPages are found, it retrieves all the html elements from the webpage. It then examines the all the HTML Elements on the webpage. It extracts the page links, page content, back link content's, page URL, page type, page last updated dateandtime.

Page Category:

Category of the webpage is determined from the extracted page links. Usually if we examine the webpage URL closely it contains the category of the page which it belongs to as sub domain name or as a sub folder name.

For e.g. 1. <http://www.microsoft.com/windowsvista/default.aspx>

2. <http://www.microsoft.com/windowsxp/default.aspx>

Above links one is for vista homepage, another is for XP homepage. If we extract the subfolder names from the urls we able to find the category of the pages.

Page catalog:

Pages content its extracted category are stored in a separate XML file which will be used by a search engine to provide a category based search system.

5.2 SYSTEM TESTING

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. Once the source code has been generated, software must be tested to uncover as many errors as possible before delivery to the customer. In order to find the highest possible number of errors, tests must be conducted systematically and test cases must be designed using disciplined techniques.

5.2.1 TYPES OF TESTING

◇ White box Testing

White box testing some times called as glass box testing is a test case design method that uses the control structures of the procedural design to derive test cases.

Using White Box testing methods, the software engineer can derive test case, that guarantee that all independent paths with in a module have been exercised at least once, exercise all logical decisions on their true and false sides, execute all loops at their boundaries and within their operational bounds, exercise internal data structures to ensure their validity. “Logic errors and incorrect assumptions are inversely proportional to the probability that a program path will be executed“.

The logical flow of a program is some times counterintuitive, meaning that unconscious assumptions about flow of control and data may lead to make design errors that are uncovered only once path testing commences.

“Typographical errors are random“

When a program is translated into programming language source code, it is likely that some typing errors will occur. Many will be uncovered by syntax and typing checking mechanisms, but others may go undetected until testing begins. It is as likely that a type will exist on an obscure logical path as on a mainstream path.

◇ Black box Testing

Black box testing, also called as behavioral testing, focuses on the functional requirements of the software. That is, black box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black box testing attempts to find errors in the following categories:

1. Incorrect or missing functions
2. Interface errors
3. Errors in data structures or external data base access
4. Behavior or performance errors
5. Initialization and termination errors

By applying black box techniques, a set of test cases that satisfy the following criteria were been created: Test cases that reduce, by a count that is greater than one, the number of additional test cases that must be designed to achieve reasonable testing and test cases that tell something about the presence or absence of classes of errors, rather than an error associated only with the specific test at hand.

Black - box testing is not an alternative to white - box testing techniques. Rather it is complementary approach that is likely to uncover a different class of errors than white - box methods.



◇ **Validation Testing**

Validation testing provides the final assurance that software meets all functional, behavioral and performance requirements. Validation testing can be defined in many ways, but a simple definition is that validations succeed when the software functions in a manner that is expected by the user. The software once validated must be combined with other system element. System testing verifies that all elements combine properly and that overall system function and performance is achieved. After the integration of the modules, the validation test was carried out over by the system. It was found that all the modules work well together and meet the overall system function and performance.

◇ **Integration Testing**

Integration testing is a systematic technique for constructing the program structure while at the same time conducting test to uncover errors associated with interfacing. The objective is to take unit - tested modules and build a program structure that has been dictated by design. Careful test planning is required to determine the extent and nature of system testing to be performed and to establish criteria by which the result will be evaluated.

All the modules were integrated after the completion of unit test. While Top - Down Integration was followed, the modules are integrated by moving downward through the control hierarchy, beginning with the main module. Since the modules were unit - tested for no errors, the integration of those modules was found perfect and working fine. As a next step to integration, other modules were integrated with the former modules.

After the successful integration of the modules, the system was found to be running with no uncovered errors, and also all the modules were working as per the design of the system, without any deviation from the features of the proposed system design.

◇ **Acceptance Testing**

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate that the implemented system satisfies its requirements. When custom software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all requirements.

In fact acceptance cumulative errors that might degrade the system over time will incorporate test cases developed during integration testing. Additional testing cases are added to achieve the desired level functional, performance and stress testing of the entire system.

◇ **Unit testing**

Static analysis is used to investigate the structural properties of source code. Dynamic test cases are used to investigate the behavior of source code by executing the program on the test data. This testing was carried out during programming stage itself.

After testing each every field in the modules, the modulus of the project is tested separately. Unit testing focuses verification efforts on the smallest unit of software design and field. This is known as field - testing.

5.2.2 TEST CONSIDERATIONS FOLLOWED IN THIS PROJECT

The test that occurs as part of unit testing is given below.

◇ Interface

Tested to ensure the information properly flows in and out of the program unit under test.

◇ Local Data Structures

The temporarily stored data in this module have been checked for integrity. It was seen that no lose of data or misinterpretation of data was taking place in this module.

◇ Boundary Conditions

The data to this module have fixed length and are known to have a particular range of values. The input data with corresponding lower bound and upper bound values and also the values in between the range, and was found that the module operates well with the boundary conditions.

◇ Independent Paths

The module was tested for independent paths to bound values and also the values in between the range, and was found that the module operates well with the boundary conditions.

Error Handling Paths

The module was tested for error handling conditions. The module was given wrong input and was checked for error paths. It was found that the module was able to produce appropriate error messages for all the wrong inputs given to the module. For example, when alphabets were entered for amount, the module generates error specifying that.

CHAPTER 6

CONCLUSION

The next-generation Web architecture represented by the Classified Web will provide adequate instruments for improving search strategies and enhance the probability of seeing the user query satisfied without requiring tiresome manual refinement. However, actual methods for ranking the returned result set will have to be adjusted to fully exploit additional contents characterized by the classification of web page . Several ranking algorithms for the Web exploiting classification-based metadata have been proposed. Nevertheless, they mainly use page relevance criteria based on information that has to be derived from the whole knowledge base, making their application often unfeasible in huge web environments.

The present paper discusses a web page ranking algorithm, which consolidates web page classification with web page ranking to offer flexibility to the user as well as to produce more accurate search result. The classification is done based on several properties of a web page which are not dependent on the meaning of its content. The existence of this type of classification is supported by applying fuzzy c-means algorithm and neural network and the knowledge will be created by characterized billions of pages, and possibly altered through next generation spam elimination techniques.

CHAPTER 7

SCOPE FOR FURTHER DEVELOPMENT

- Adding Support for more Categories
- Improving Categorization of Pages
- The system has much scope in the future and it can be developed to add more features to satisfy the user's request and company's request.
- Modification and enhancement can be made affecting any other part of the program because of the user friendliness and understandability of the project.
- The data screens can be upgraded and menus can be easily added when required. Items can be added to the forms when there comes necessity of new data.

CHAPTER 8

APPENDIX

8.1 SCREENS

LOGIN SCREEN



SARVA SEARCH

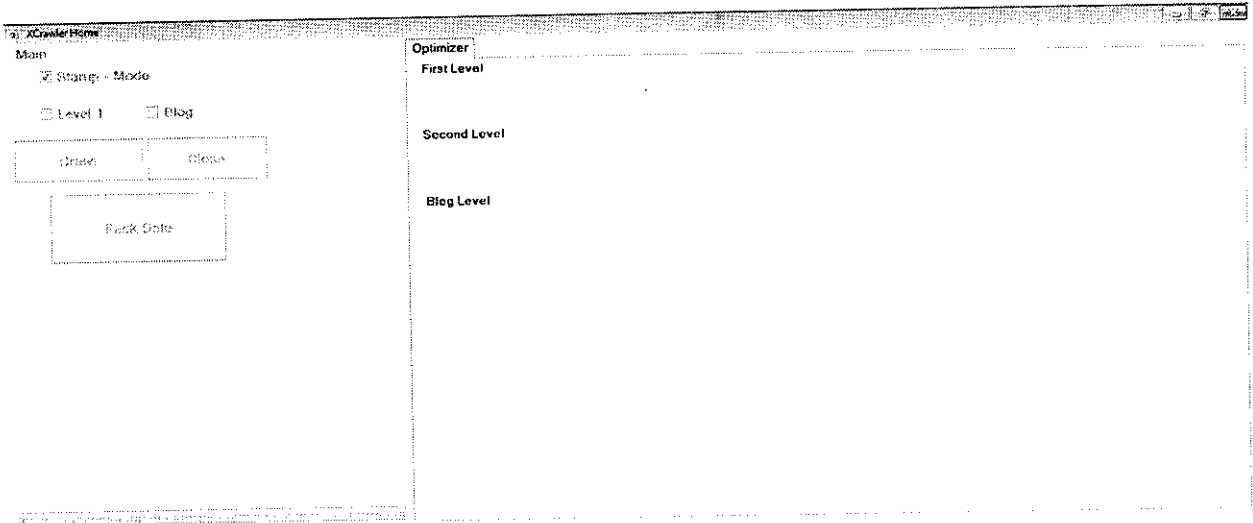
User ID

Admin

Password

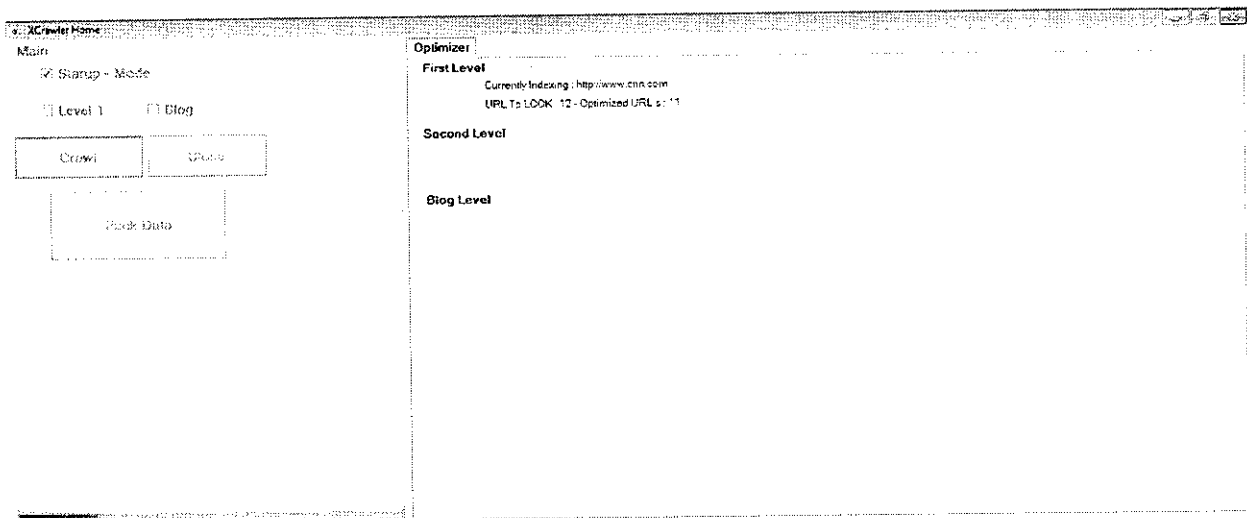
Login

HOME SCREEN



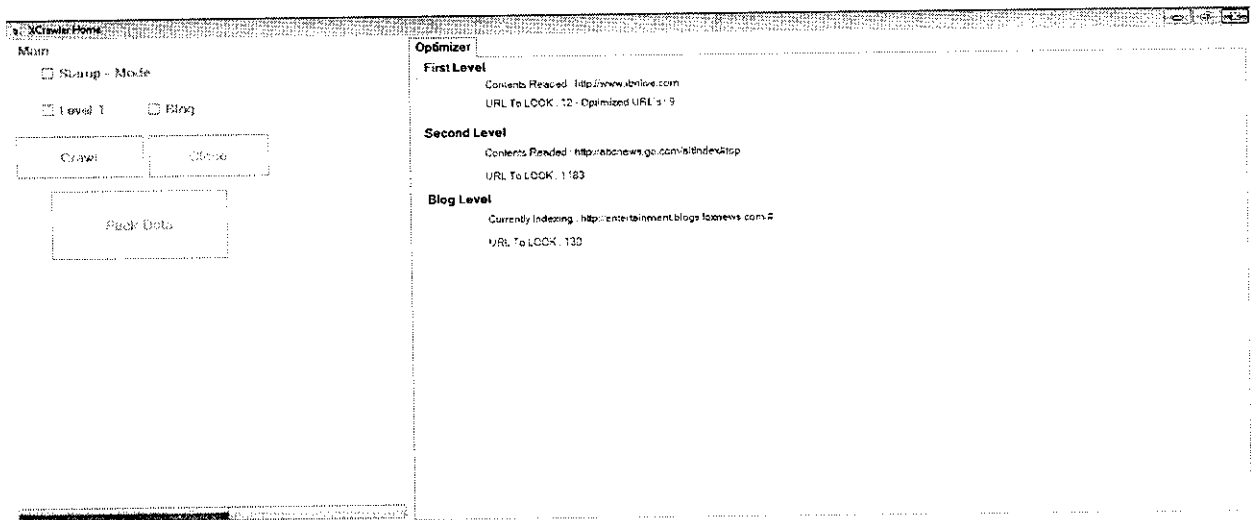
SARVA SEARCH

CRAWLING SCREEN



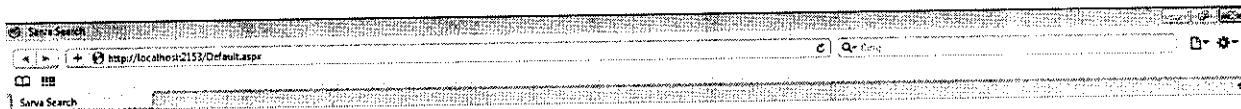
SARVA SEARCH

CRAWLING SCREEN



SARVA SEARCH

SEARCH HOME



SARVA SEARCH

Enter SEARCH

RESULT SCREEN



9. REFERENCES:

1. Steven Holzner, “**Visual Basic.NET Black Book**”, 2003 Edition, Dreamtech Publications
2. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. in, 1998
3. L. Brieman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*, Wadsworth, Belmont, CA, 1983.
4. B. Aleman-Meza, C. Halaschek, I. Arpinar, and A. Sheth, “A Context-Aware Semantic Association Ranking,” Proc. First Int’l Workshop Semantic Web and Databases (SWDB ’03),
5. R. Baeza-Yates, L. Caldero’n-Benavides, and C. Gonza’lez-Caro, “The Intention behind Web Queries,” Proc. 13th Int’l Conf. String Processing and Information Retrieval (SPIRE ’06), pp. 98-109, 2006.
6. N. Stojanovic, R. Studer, and L. Stojanovic, “An Approach for the Ranking of Query Results in the Semantic Web,” Proc. Second Int’l Semantic Web Conf. (ISWC ’03), pp. 500-516, 2003.

Websites:

1. www.msdn.microsoft.com
2. www.vbcity.com
3. www.vbdotnetheaven.com
4. www.swoogle.com
5. www.godaddy.com