# WEB SITE ANALYZER

# DEVELOPMENT METHODOLOGY

JIANG, FAN

A MAJOR REPORT

IN

THE DEPARTMENT

OF

COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE

CONCORDIA UNIVERSITY

MONTREAL, QUEBEC, CANADA

AUGUST 2003

*Your file Votre référence*
*ISBN: 0-612-83904-4*
*Our file Notre référence*
*ISBN: 0-612-83904-4*

# Abstract

Web Site Analyzer (WSA) is a tool to analyze a web site structure and generate a report to present the accessibility categories of the file in this web site. WSA uses Client/Server/Database Architecture. Interface design adopts web browser based user graphic interface. During the development of this tool, two of Extreme Programming practices are applied: Pair Programming and Test Driven Programming, which enhance the co-operation and quality of this project.

# Table of Contents

## List of Figures

# 1 Introduction

Maintaining a web site is a significant task for a web master / developer. A personal web site may contain thousands of files. What's the reference relationship of all these files? What's file accessibility structure of the web site? Are there any dead links in a web site that prevents the end users from accessing? There are tools available to resolve the above-mentioned concerns, but they are slow due to remote access, expensive, or cumbersome.

This major report is to develop an application named WSA (Web Site Analyzer), which is simple to use and provides useful results quickly. This system is designed to analyze a web site structure and generate a web browser based user graphic interface report to present the categories of the file in this web site. The category is based on the web accessibility. The main purpose of this application is to let the web masters or personal web site owners to easily know / maintain the file accessibility structure of their web sites.

During the implementation of this project, I tried to apply two extreme programming practices (Pair programming and Test driven programming) in every phase of development. This major report will discuss how pair programming and test driven programming enhance the co-operation and quality of this project.

In this document, section 3 identifies phase of requirement gathering and analysis. Section 4 describes related work done by previous developers. Section 5 discusses the design principles and the major areas that we want to improve WSA application. Section 6 describes the major components of this application and the relationship of classes. Section 7 presents extreme programming methodologies applied in this project. Section 8 defines the system test procedure.  Section 9 compares a commercial product to the application we developed. Section 10 summarizes the conclusion and suggests the future works to improve WSA application.

.

# 2 Scope and Project milestone

## 2.1 Scope

The scope of this Major Report is to develop a tool to analyze the file accessibility of a web site and give the user a graphic report of analysis. The user starts WSA by giving the URL address as input via the user interface. The program scans HTML files recursively and builds a tree menu graph.

## 2.2 Joint Effort

This project is the joint-effort from two developers (Lu, Yukui and Jiang, Fan) who are working as a pair through every phase of this application. We take the advantage of co-operation between two persons by pair analyzing, pair design, pair programming and pair testing. This pair working style is encouraged by the extreme programming methodology, which we believe in. Therefore, the major report documentation part of Lu, Yukui and Jiang, Fan will be similar in the following sections:

- Introduction Part
- Requirement Part
- Test Part
- Conclusion and related work

According to our architecture design, we have two components with nearly no dependency. Therefore, each of us leads the effort of one component, even though we still work as a pair from time to time. The following sections of this document are written separately:

- Design
- Implementation
- Technology Tools (Lu, Yukui)
- Extreme Programming Methodology (Jiang, Fan)

## 2.3 Milestone

Supervised by Professor Peter Grogono, this major report was started from January 2002. The milestone and procedure of this work are following:

1. Be familiar with the topic and related works in this field as well as be familiar with the old WSA tool.
2. Analysis the old WSA's architecture, design, implementation, and performance as well as its advantage which we could inebriate and disadvantage which we should improve.
3. New WSA architecture and design, background technology and tool learning.
4. Implementation—Coding and Unit Testing.
5. Test case design and System Testing
6. Documentation
7. Make a conclusion for this research work and provide recommendations for future works.

# 3 Requirements

## 3.1 Original Requirement

Requirements for this project covers the requirements given by Professor Peter Grogono [8].

The Analyzer should find all files in the starting directory and all files that are:

1. Reachable from those files by traversing links and
2. On the local system.

The second condition is intended to prevent the Analyzer returning the entire World Wide Web as its result.

The Analyzer reports:

1. The files that it found;
2. For each file, the links from it and the links to it;
3. “Orphans" ---- files with no incoming links;
4. “Leaves" ----files with no outgoing links;
5. “Foreigners"---- links of remote files.
6. “Dead link” – URLs which are not found.

In order to work properly, the Analyzer has to parse HTML, at least partially. As a side effect of its work, it could produce a list of warnings of HTML errors.

## 3.2 Requirement Analysis

### 3.2.1 Links

Based on 3.1, we categorize the links and analyze the following five types of links:

1. Links Reference To (Incoming links to the specific file)
2. Links Referenced By (Outgoing links from the specific file)
3. Foreign Links
4. Dead Links (All un-reachable links, ex. permission denied and inaccessible)
5. Leave Links

There is some relationship among these types. For example, leave link and foreign link may have incoming links.

In this project, we did not implement the orphan link since our traverse algorithm cannot reach this type of links. This will be mentioned on the future work section.

### 3.2.2 GUI

- Project requires that user has graphic interface to input the base link.
- Application should have the clear visual presentation of link structure.

## 3.3 Development Environment

The development environment and software tools that were used to develop WSA are described as following:

- Platform – WINDOWS 2000

- Developing languages – JAVA, JSP, JavaScript, HTML
- Database – Microsoft Access 2000
- Web Server – Apache Tomcat 3.2.4
- IDE – JBuilder Foundation 4.0.

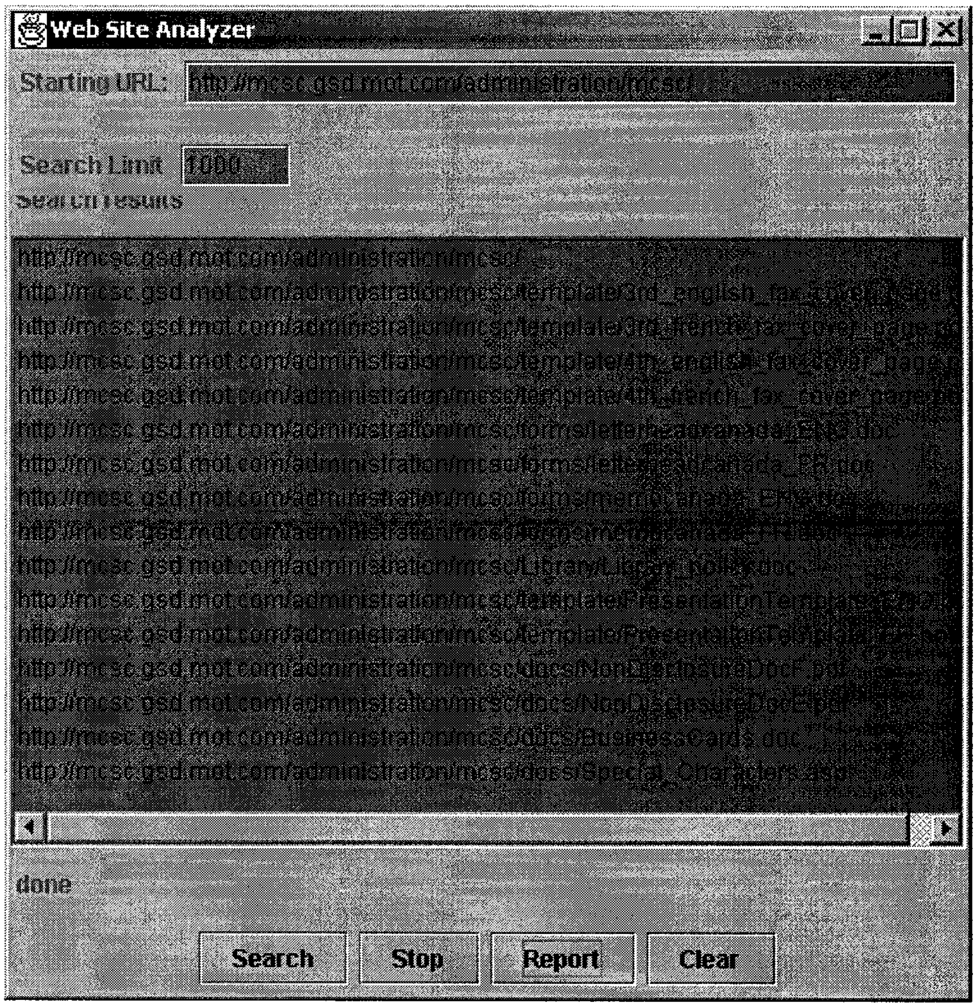
# 4 Related Work

This project did not start from the scratch. We analyzed WSA C++ version implemented by Professor Peter Grogono and Java version implemented by Graduate student Yuan Xu [1].

We inherit and reuse parse algorithm/code from these two versions of implementation. Yuan Xu's WSA version has two different graphic interfaces:

- User input interface implemented in Java Swing
- Report interface in the static HTML file

Showed as below pictures:



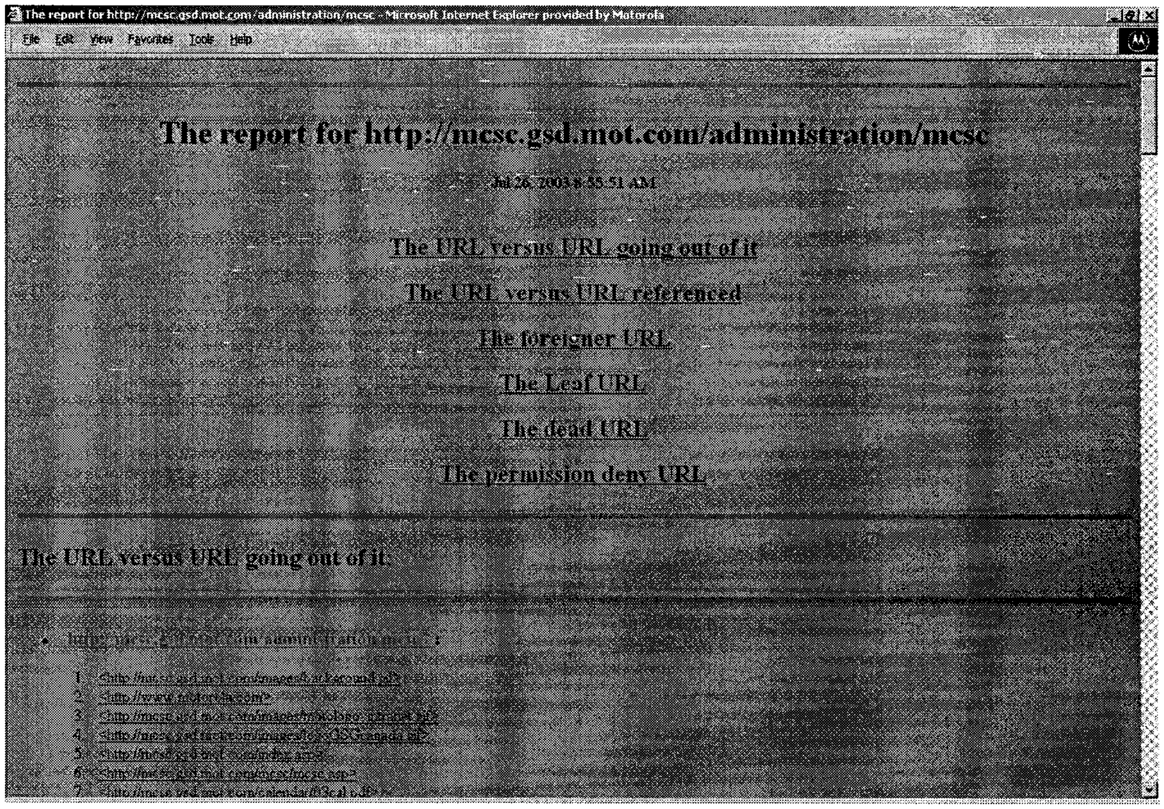
Starting URL: http://mcsc.gsd.mot.com/administration/mcsc/

Search Limit 1000

```
http://mcsc.gsd.mot.com/administration/mcsc/
http://mcsc.gsd.mot.com/administration/mcsc/template/3rd_english_fax_cover_page
http://mcsc.gsd.mot.com/administration/mcsc/template/3rd_french_fax_cover_page
http://mcsc.gsd.mot.com/administration/mcsc/template/4th_english_fax_cover_page
http://mcsc.gsd.mot.com/administration/mcsc/template/4th_french_fax_cover_page
http://mcsc.gsd.mot.com/administration/mcsc/forms/letterheadcanada_ENG.doc
http://mcsc.gsd.mot.com/administration/mcsc/forms/letterheadcanada_FR.doc
http://mcsc.gsd.mot.com/administration/mcsc/forms/memocanada_ENG.doc
http://mcsc.gsd.mot.com/administration/mcsc/forms/memocanada_FR
http://mcsc.gsd.mot.com/administration/mcsc/Library/Library_Policy.doc
http://mcsc.gsd.mot.com/administration/mcsc/template/PresentationTemplate
http://mcsc.gsd.mot.com/administration/mcsc/template/PresentationTemplate
http://mcsc.gsd.mot.com/administration/mcsc/docs/NonDisclosureDocF.pdf
http://mcsc.gsd.mot.com/administration/mcsc/docs/NonDisclosureDocE.pdf
http://mcsc.gsd.mot.com/administration/mcsc/docs/BusinessCards.doc
http://mcsc.gsd.mot.com/administration/mcsc/docs/Special_Characters.asp
```

done

**Figure 1 Based code Java GUI**


The report for http://mcsc.gsd.mot.com/administration/mcsc - Microsoft Internet Explorer provided by Motorola
File Edit View Favorites Tools Help

The report for http://mcsc.gsd.mot.com/administration/mcsc

Jul 26, 2003 8:55:51 AM

The URL versus URL going out of it

The URL versus URL referenced

The foreigner URL

The Leaf URL

The dead URL

The permission deny URL

The URL versus URL going out of it:

- [illegible] :
  1. <http://mcsc.gsd.mot.com/images/background.gif>
  2. <http://www.motorola.com>
  3. [illegible]
  4. [illegible]
  5. [illegible]
  6. [illegible]
  7. [illegible]

**Figure 2 Based code Report GUI**

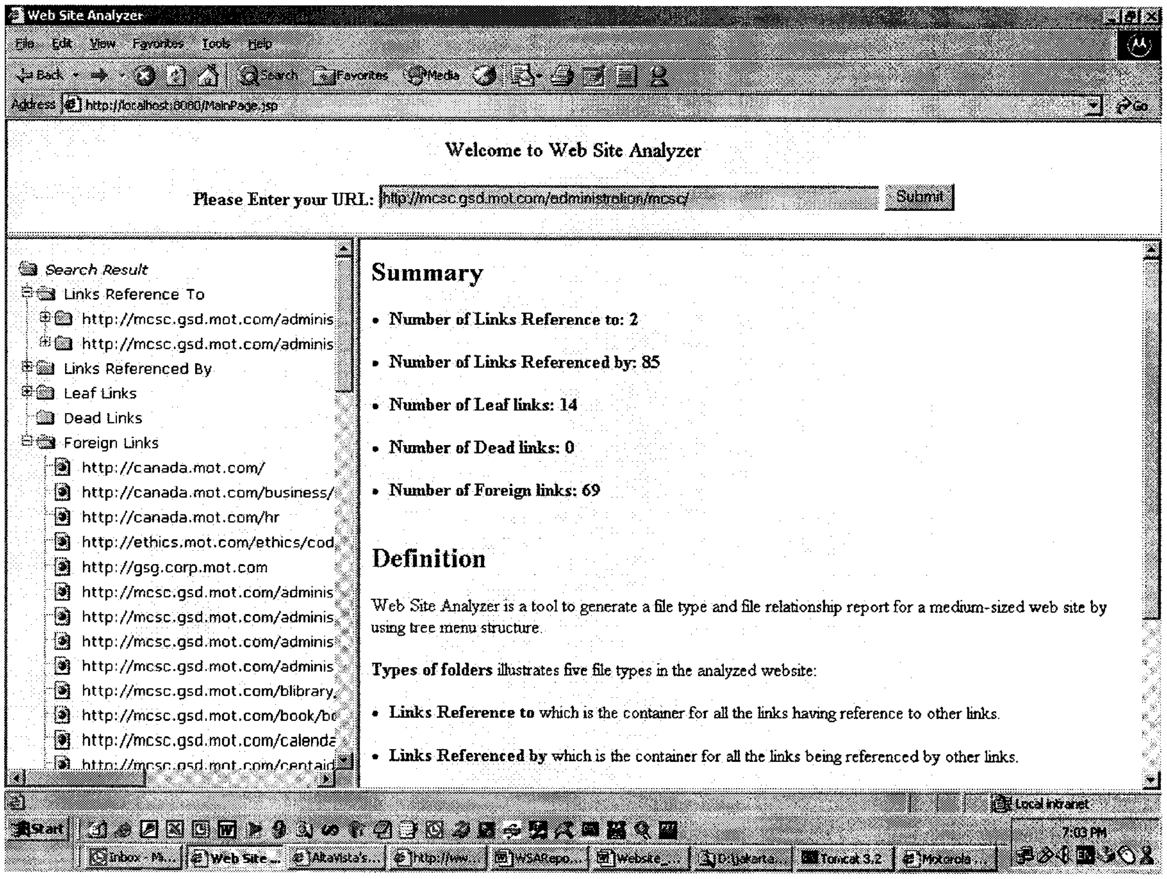
This interface provides user clear output and offers the user a friendly interface for input.

We analyze this GUI implementation:

- The report was a plain HTML page without frame, so the relationship between links and link types are not straightforward.
- When clicking on each category to see the detailed information of this category or view the contents of a link, the high level view will be lost since the page is updated after the click action.
- Report can be lengthy after analyzing the large web site and it will be hard to identify the place which the user want to go.

- User input was from a Java Applet application and the report will be available in a HTML pages. This needs two GUI windows.
- Report page is static HMTL. User cannot do any interaction in this page.

In order to resolve all these limitations by the high level design, we decide to apply tree-menu graphic implementation. The picture below shows our final presentation. More details will be discussed in the design part of this document.


Web Site Analyzer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/MainPage.jsp
Go
Local intranet
Start
7:03 PM

Welcome to Web Site Analyzer

Please Enter your URL: http://mcsc.gsd.mot.com/administration/mcsc/ Submit

Search Result
Links Reference To
http://mcsc.gsd.mot.com/adminis
http://mcsc.gsd.mot.com/adminis
Links Referenced By
Leaf Links
Dead Links
Foreign Links
http://canada.mot.com/
http://canada.mot.com/business/
http://canada.mot.com/hr
http://ethics.mot.com/ethics/cod
http://gsg.corp.mot.com
http://mcsc.gsd.mot.com/adminis
http://mcsc.gsd.mot.com/adminis
http://mcsc.gsd.mot.com/adminis
http://mcsc.gsd.mot.com/adminis
http://mcsc.gsd.mot.com/blibrary
http://mcsc.gsd.mot.com/book/bc
http://mcsc.gsd.mot.com/calenda
http://mcsc.gsd.mot.com/centai

**Summary**

- **Number of Links Reference to: 2**
- **Number of Links Referenced by: 85**
- **Number of Leaf links: 14**
- **Number of Dead links: 0**
- **Number of Foreign links: 69**

**Definition**

Web Site Analyzer is a tool to generate a file type and file relationship report for a medium-sized web site by using tree menu structure.

**Types of folders** illustrates five file types in the analyzed website:

- **Links Reference to** which is the container for all the links having reference to other links.
- **Links Referenced by** which is the container for all the links being referenced by other links.

**Figure 3 New Interface Design**

# 5 Design

## 5.1 Architecture design

The biggest design changes of this version of WSA are:

### 5.1.1 Data storage

Our based code is using vector as data structure to hold all the data, which will be used for final report display. Since vector is declared in memory, it temporarily exists and cannot be reusable. When the end user exits the program, all the data will be lost. The other shortage of this design is that the application depends on the memory capacity since vectors are run time variable. Therefore, our design adopts database as a medium to exchange the data between client and server. This way, the web site analyzer can have more capacity to handle the larger site if time is not issue. We have a test that if we analyze a web site containing 3000 - 4000 links, it takes about 2 minutes. Theoretically, this design can analyze any size of web site. And at the same time, DB can be saved and referenced later or can be input for another application.

### 5.1.2 Client/Server/Database Architecture

This architecture, which distributes the functionalities of the system between different components, possibly scattered in various locations, are suitable for Web applications that handle intensive users' interaction with a central database. Due to the fact that our software system has the general characteristics of a client/server/database system, we are building it in accordance with the client/server/database architecture.

### 5.1.3 Database Connectivity

Our development used MS Access DB system as DBMS since it is easy to get even though it is not very powerful. It needs ODBC connectivity protocol to access its data. But our application adopts all the technology from non-Microsoft family: JDK java language, JSP. These need JDBC as their connection representative. Therefore we use ODBC-JDBC driver to bridge these two connectivity protocols. The beauty of this connectivity protocol is that they are standard and not depending on DB type. Therefore we can extend the DB type easily. At most, we need load proper driver in Java application.

## 5.2 Loose coupling

Client server architecture can let this application have loose coupling. The application algorithm and application Database building up are in server site developed by Java JDK. And user interface is at client site, which is totally separated from server part. This way allows two developers code the whole application in parallel. Database and Tomcat web server connect these two parts together.

This loose coupling also is good for maintenance code and to isolate the errors.

## 5.3 Interface design

We decided to use standard web browser as our graphic interface container. Now more and more industry application move from window based GUI to Web browser based GUI. Web browser based GUI has several pros:

- **Standard client application**: Now almost every computer has its browser for Internet. Normally there is no cost for usage.
- **User friendly**: the most of end users are familiar with browser application. We do not need complicated user manual to teach user how to use this GUI.
- **Match WSA characteristics**: WSA itself is an application related to web. So it makes more sense to use browser based GUI.
- **GUI can be easily extended**: there are a lot of standard techniques to improve a web page's look and feel. So this interface can easily be upgraded by just modified HTML code without any language compiling.

According to the above, we get rid of all the JAVA GUI part in our based version and let browser contain all end user interfaces (Refer to figure 3).

# 6 Implementation

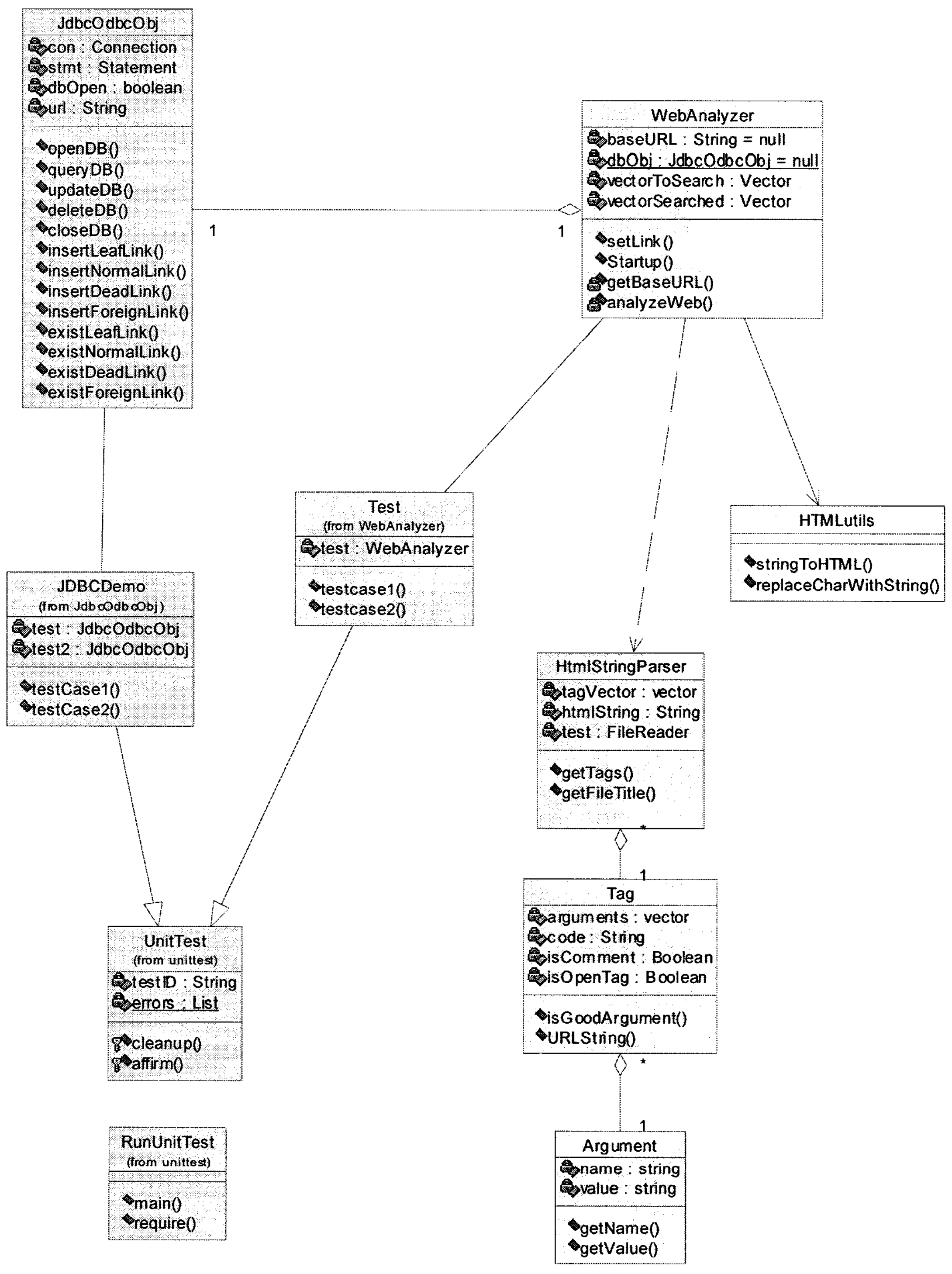


**Figure 4 Class Diagram**

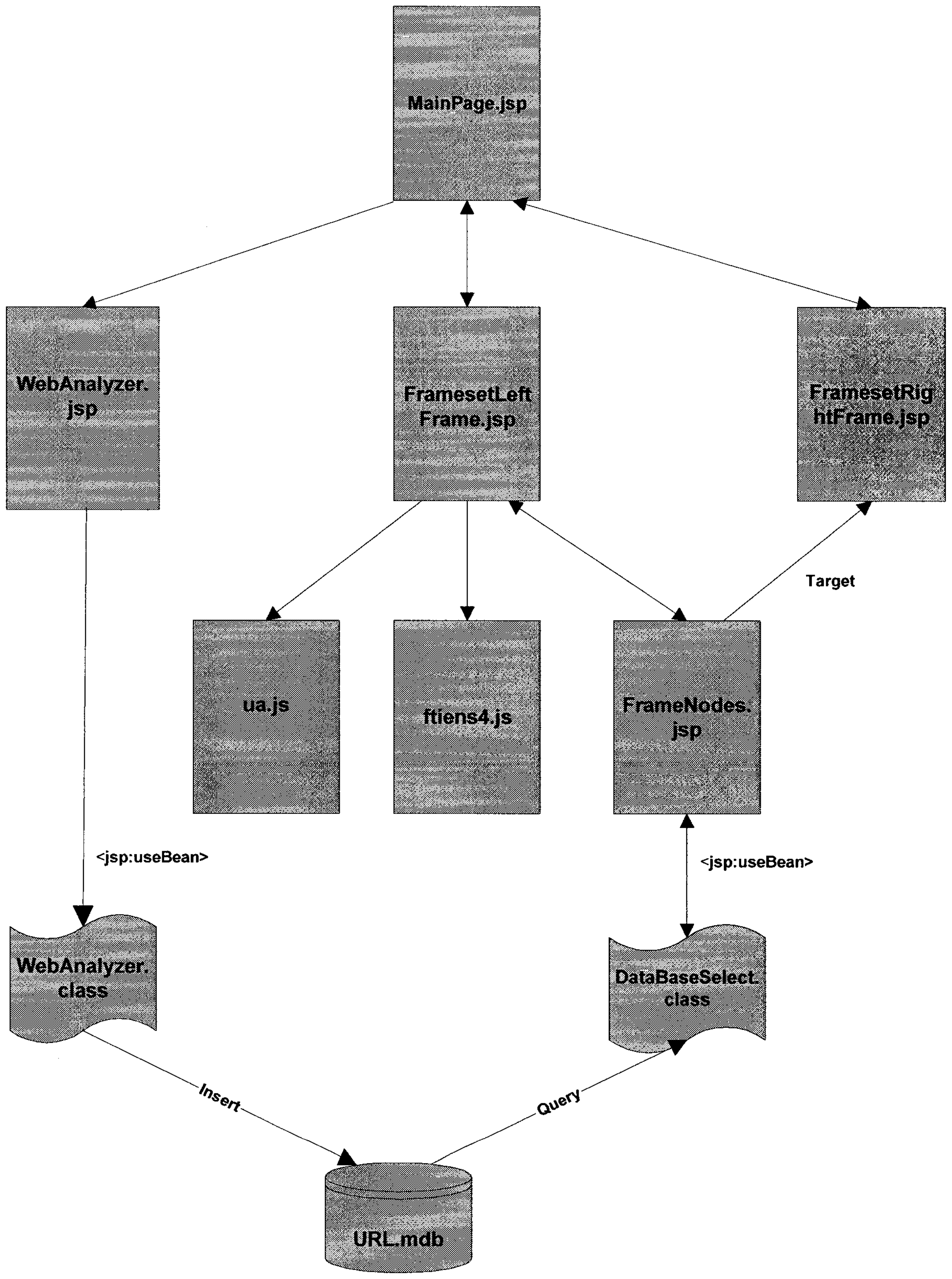


**Figure 5 JSP Diagram**

The class diagram shows how the classes involved in this major report.

This section will be focus on functionality of classes in class diagram. JSP pages and user interface part will be elaborated by the other developer's Major Report since this product is a joint effort. Also all the unit test classes will be discussed by the Unit Test section of this report.

## 6.1 Main

We have two main entries:

- Unit test entry is for unit test purpose. This can be run from the **main** function of RunUnitTest Class. More detail of this entry will be discussed later in Unit Test Section.

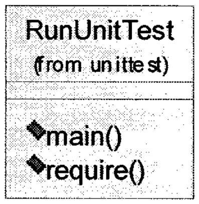


**Figure 6 RunUnitTest Class**

- Product entry is from WebAnalyzer.jsp JSP page. Javabean methods setLink( ) will be triggered from dynamic page.

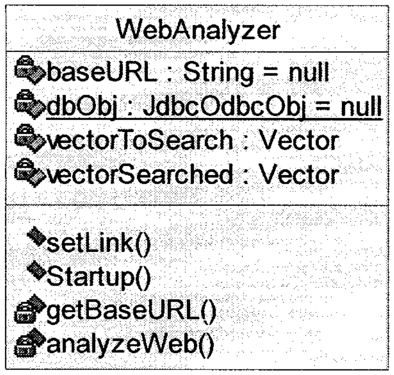


**Figure 7 WebAnalyzer Class**

WebAnalyzer Class is not designed to contain GUI code as before, since the major design change is that moving GUI part from java code to dynamic page. Major task of this class is to update the database. At the end of analysis, DB should contain all the links for display in the output.

## 6.2 DB access Component

This component only contains one class – JdbcOdbcObj. We tried to isolate all database related functionality into this class. This design is for code maintainability and reusability. This class has all the DB standard operations used in this project like open DB, close DB, queryDB …

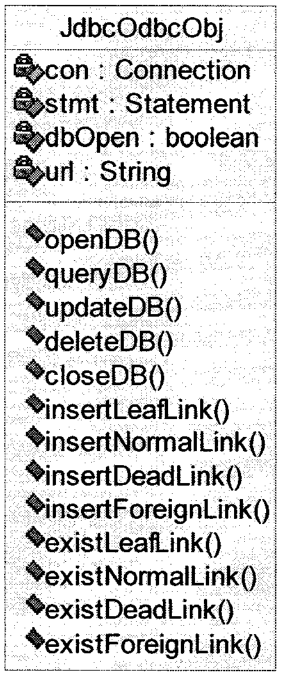


**Figure 8 JdbcOdbcObj Class**

## 6.3 Parser Component

This Component is responsible for parsing the html file, which is passed from main class.

Refer to [1] for the more detail of this component.

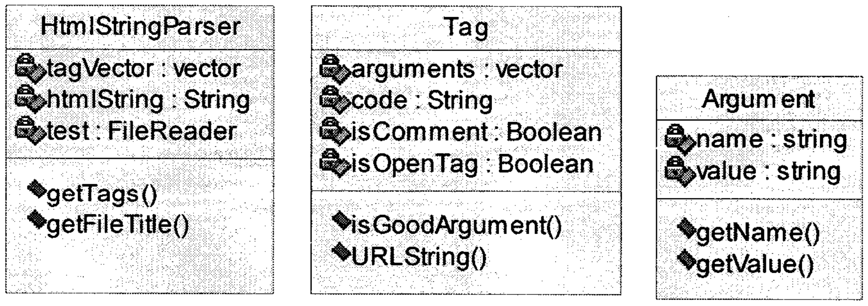


**Figure 9 Parser Classes**

# 7 Extreme Programming Methodology

We applied two methods of the extreme programming methodology in this project: Pair Programming and Test driven programming.

## 7.1 Pair Programming

### 7.1.1 Experience of Pair Programming

I applied and am applying pair programming in my professional career. In March 2003, I tasted pair programming accidentally in my company. The team I participated encountered a brand new project in complicated system and this project was just one portion of this system. The manager just group his stuff by two since the task was only configuration, which was new to everybody. He did not mention pair programming and just said this way (two persons working together) can easily warm up in a new project. Later on, when we had more experience in our task, he let us work alone. Therefore I do not think he used pair programming on purpose. But this chance of practice allowed me to discover that pair programming was so enjoyable.

Three months later, I switched to another project and was assigned to a task, which was technically complicated. At the same time, we did not have enough domain knowledge on it. In order to conquer it, my project leader let me and another developer work together to find solution. Then I was entering the second experience of pair programming. This time the task was very challenging. But we found two of us working together were kind of creative and almost we can resolve any problem we met. We were thinking loud, discussing actively, brainstorming, etc. From this experience, I became a pair-programming fan and was more and more interested on extreme programming. Furthermore, I presented the advantage of this methodology to project leader and hoped I could continue this practice throughout the whole project. He accepted since our productivity convinced him.

Since then I worked with two other project managers, one of them gave me great support on XP and the other one had already practiced it before. Thus I had almost one-year experience on pair programming and am continuously practicing it now. Up to now, I have paired with 4 different developers.

There are two developers involved into this WSA project, so we have a perfect chance to practice pair programming. Actually it's not just pair "programming", it's pair working. Even though we have loose coupling design and two components have little dependency, we did not work separately as solo programming. From the beginning of the project, we are working in same physical place, discussing and analyzing the requirement, coming out a design solution by pair. In the code and unit test phase, each of us leads effort for one component, but we still apply pair programming by sitting in front of the same desktop from time to time.

### 7.1.2 Advantages of Pair Programming

According to my experience of pair programming, and the paper "The Costs and Benefits of Pair Programming" [2], the following advantages can be highlighted:

#### 7.1.2.1 Teamwork

It's pair programming that let me understand the spirit of teamwork. From my experience, collaborating with the other person means that sharing idea, understanding the difference, and compromising each other's schedule. The more important thing is anytime when a progress is made, there is always someone sharing the happiness immediately, genuinely. When developers are faced to the high stress, there is always someone sharing the burden.

#### 7.1.2.2 Review code

Today, almost everyone in software engineering field knows that how important the code review and code inspection are. Code review can catch the error/defect in the early stage and avoid the big headache in the custom site. But in the reality, as I know, a quite few developer are reluctant to review the code written by the other persons. Firstly code reviewer and code developer may not in the same context and domain, so in order to review the code, the reviewer may spend some extra time to learn the background of code. Even reviewer is from the same background (ex. Same group member or key work partner), but digging into the other person's code might be difficult. Theoretically review meeting should be each reviewer review the code and post his or her comments electronically or in the paper and in the meeting, reviewer and author validate all the commends. But some time, people review code during meeting.

Pair programming is a perfect way for the code review. Every line of the code is out of two people. Navigator, which acts as code reviewer, consistently reviews the code written by the driver, which acts as code author, every minute. Since they are facing same problem and they are in exactly same context and sharing the same design by pair designing, so the review is effective.

#### 7.1.2.3 Be creative

Creativity is not only from the genius. Ordinary people can be creative by exchanging/ sharing the ideas together.

- **Thinking loud**

I experienced some issues resolved just by thinking loud. We find the solution just after our discussion with the pair, or sometimes one person just explains the problem and without the pair any input, he find the solution right way.

- **Brainstorm**

During the pair programming, some challenge issues may block the progress. there are a lot of discuss and idea exchange between peers. And one thought will sparkle the other thought and finally the solution is dig out. This is most beautiful moment in my pair programming experience.

#### 7.1.2.4 Stick to process

In the organization and a big project, there might be some process we need to stick to, for example, following a specific code convention, proper documentation, source code labeling, etc. In pair programming, since the other person is monitoring coding, the code author will be more willing to stick to process. This way, the code development has good tracebility.

#### 7.1.2.5 Enjoyable

Not everyone likes pair programming at the first try. People get used to be solo programmer. But after several practice, my partners like it. This makes the work environment more enjoyable. We found a lot of fun/satisfaction during software development.

There is a citation from paper [2]:

*"The adjustment period from solo programming to collaborative programming was like eating a hot pepper. The first time you try it, you might not like it because you are not used to it. However, the more you eat it, the more you like it."*

#### 7.1.2.6 Be productive

It seems like that pair programming waste the time by two persons working on the single task. But in the reality, it reduces the time of the development cycle.

**High quality means less post release bug fix**

According to [2], there are only 15% more time than solo programming during code and unit testing phase, but since pair programming produces high quality software by continuously reviewing and sticking to effective process, it reduce the number of defects and saves tremendous post release effort.

**Quickly find the solution**

Most of people have been stuck with some problems (it may be hard or may be very easy) and it takes long time to figure them out. By pair programming, it seems that nothing can stop us from advancing.

**Fewer interruptions**

According to [4], people are more reluctant to interrupt a pair than they are to interrupt someone working alone.

## 7.2 Unit testing

### 7.2.1 Principle

In this major report, I incorporated the two major theories into our code since I am truly addictive to them:

- Think in Patterns with Java [3]. In this design pattern book, the author Bruce Eckle highly valued the unit test, and put "Unit Test" as first design pattern in his book.
- Test driven Programming

Actually Bruce Eckle "Unit Test" design pattern is based on XP test-driven programming. Some developers are reluctant to do the some "extra" works, like:

- Documentation (structured document and on-line document in code – comment)
- Code Review
- Structured test

Traditional unit test is not formalized and in the proper process. The developers just verify something that he's not sure during coding. All these test cases are randomly popped out to the authors' mind. This way, it is hard to cover the entire scenario that will happen. Therefore we need a process to help us formalize this.

As Bruce Eckle indicated in his book:

> *One of the important recent realizations is the dramatic value of unit testing. This is the process of building integrated tests into all the code that you create, and running those tests every time you do a build. It's as if you are extending the compiler, telling it more about what your program is supposed to do. That way, the build process can check for more than just syntax errors, since you teach it how to check for semantic errors as well.*

If the unit test set is an extend compiler. We will be more confident about our code functionality.

But all these should build on top of **Test Driven Programming.** According to [4]:

Extreme Programming champions the use of tests as a development tool

The rule of this approach is: Write test code first. Since test first design will force developer to:

- Define precisely what a method does
- Begin writing a method
- Know when you are done by writing a method
- Know the minimal scaffolding needed to run a method

This major report cannot implement all these exactly but at least I have chance to practice this philosopher. My test code structure followed Bruce Eckle's one and reused his test main frame. According to him, this structure is simpler than Junit [5].

### 7.2.2 Unit test Structure

The following diagram explains this structure:

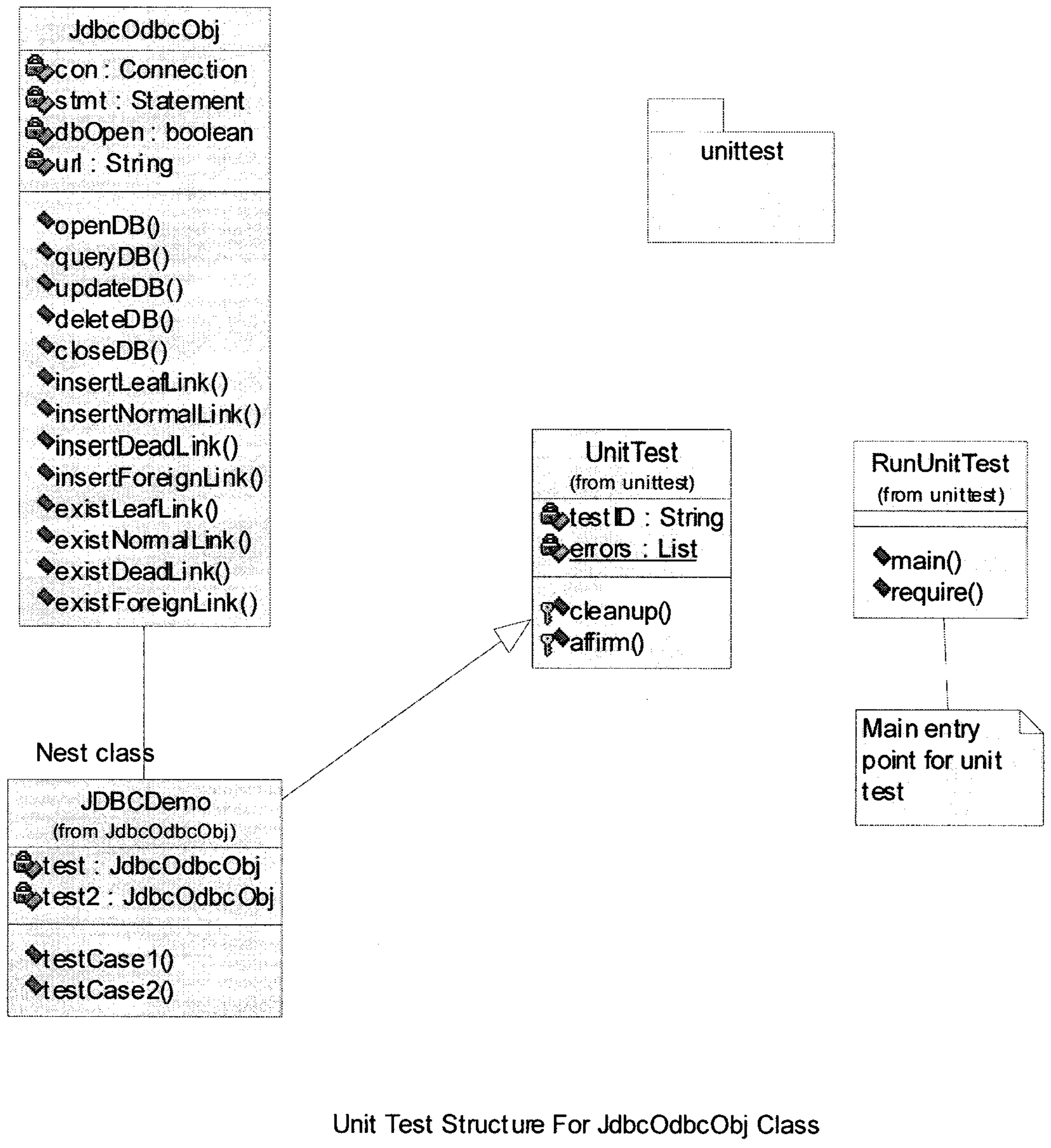


**Figure 10 Unit Test Structure For JdbcOdbcObj class**

The above diagram is based on how to unit test class JdbcOdbcObj. There are four classes involved:

- JdbcOdbcObj is the target class, which all the test cases will apply on it

- JDBCDemo is a class nested in JdbcOdbcObj which contains all the test cases and test objects (normally the type of these test objects are target class). This class is inherited from Unit test class.
- Unit test class is a generic class that can be shared all the other unit tests, so it is packed into unittest Package. This class contains all the generic features of unit test.

RunUnitTest is a generic class and part of unittest package. Under unit test environment, this is the main class to execute all the test cases of every target class.

## 7.2.3 Specific Scenario for Unit Test and how it works

I want to test the existing links features inside JDCBODBCObj class:

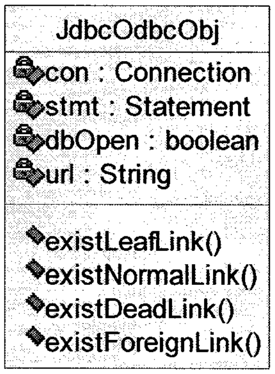


**Figure 11 JdbcOdbcObj Class**

I wrote a test case for this firstly:

```
public void testCase1() {
  test2.openDB();
```

```
    test2.deleteDB();

   System.out.println("Exist Dead Link? = " +
test2.existDeadLink("Dea"));

   System.out.println("Exist Foreign Link? = " +
test2.existForeignLink("http://mcsc.gsd.mot.com/process/"));

   System.out.println("Exist Normal Link? = " +
test2.existNormalLink("http://mcsc.gsd.mot.com/administration/mcsc/", "ht
tp://mcsc.gsd.mot.com/blibrary/default.asp"));

   System.out.println("Exist Leaf Link? = " +
test2.existLeafLink("http/mcsc.gsd.mot.com/administration/mcsc/template/
4th_french_fax_cover_page.pdf"));

   // test.updateDB(updateString);

    test2.closeDB();

   }
```

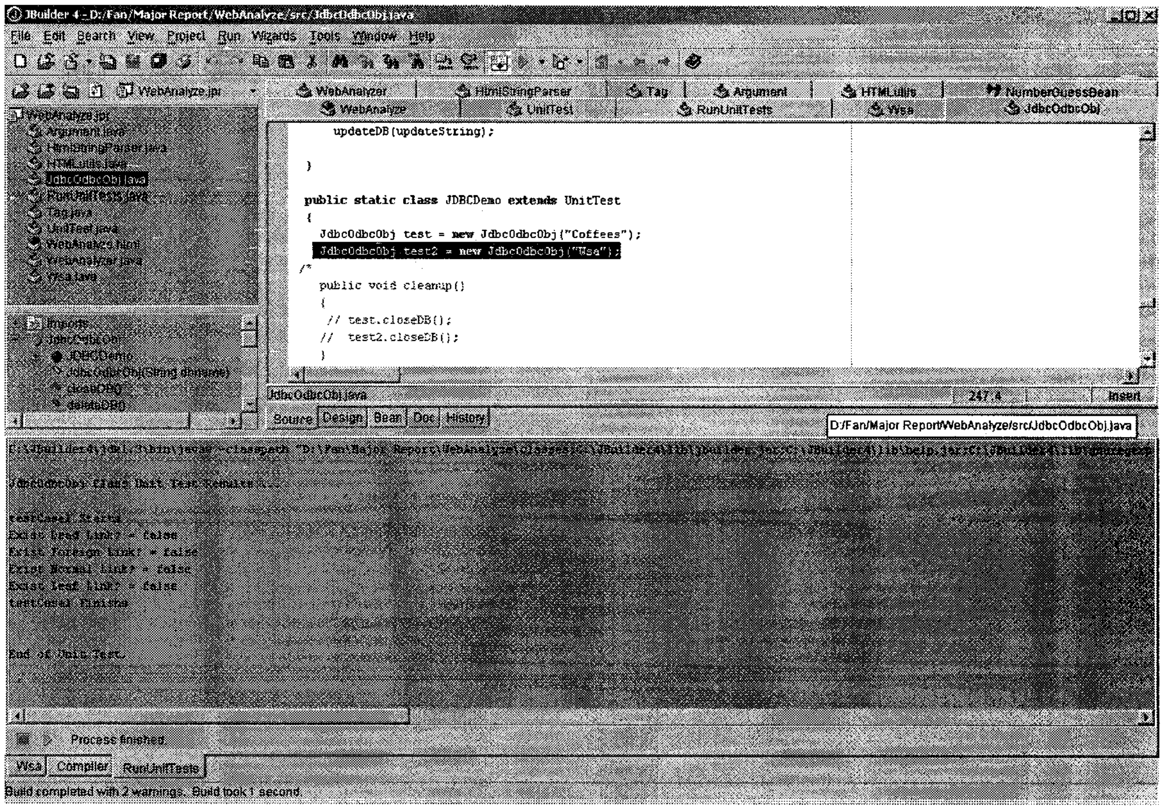
This function is inside JDBC demo class. And test2 object is declared in this demo class:

```
JdbcOdbcObj test2 = new JdbcOdbcObj("Wsa");
```

Our IDE environment is Jbuilder 4. Go to Project setting to change the property:

- Define main class of this project as unitest.RunUntiTest
- Define application parameter as JdbcOdbcObj

Now we can run this application. The result is:


JBuilder 4 - D:/Fan/Major Report/WebAnalyze/src/JdbcOdbcObj.java
File Edit Search View Project Run Wizards Tools Window Help
WebAnalyze.jpx
WebAnalyzer HtmlStringParser Tag Argument HTMLutils NumberGuessBean
WebAnalyze UnitTest RunUnitTests Wsa JdbcOdbcObj
WebAnalyze.jpx
Argument.java
HtmlStringParser.java
HTMLutils.java
JdbcOdbcObj.java
RunUnitTests.java
Tag.java
UnitTest.java
WebAnalyze.html
WebAnalyzer.java
Wsa.java
Imports
JdbcOdbcObj
JDBCDemo
JdbcOdbcObj(String dbname)
closeDBO
deleteDBO
JdbcOdbcObj.java
247:4
Insert
Source Design Bean Doc History
D:/Fan/Major Report/WebAnalyze/src/JdbcOdbcObj.java
Process finished.
Wsa Compiler RunUnitTests
Build completed with 2 warnings. Build took 1 second.

```
      updateDB(updateString);

   }

   public static class JDBCDemo extends UnitTest
   {
    JdbcOdbcObj test = new JdbcOdbcObj("Coffees");
    JdbcOdbcObj test2 = new JdbcOdbcObj("Wsa");
  /*
    public void cleanup()
    {
     // test.closeDB();
    //  test2.closeDB();
    }
```

```
C:\JBuilder4\jdk1.3\bin\javaw -classpath "D:\Fan\Major Report\WebAnalyze\classes;C:\JBuilder4\lib\jbuilder.jar;C:\JBuilder4\lib\help.jar;C:\JBuilder4\lib\gnuregexp

JdbcOdbcObj Class Unit Test Results ...

testCase1 Starts
Exist Dead Link? = false
Exist Foreign Link? = false
Exist Normal Link? = false
Exist Leaf Link? = false
testCase1 Finishs


End of Unit Test.
```

**Figure 12 IDE GUI for Unit Test**

I just highlight the message inside the output of this IDE as following:

C:\JBuilder4\jdk1.3\bin\javaw -classpath "D:\Fan\Major Report\WebAnalyze\classes;C:\JBuilder4\lib\jbuilder.jar;C:\JBuilder4\lib\help.jar;C:\JBuilder4\lib\gnuregexp.jar;C:\JBuilder4\jdk1.3\demo\jfc\Java2D\Java2Demo.jar;C:\JBuilder4\jdk1.3\jre\lib\i18n.jar;C:\JBuilder4\jdk1.3\jre\lib\jaws.jar;C:\JBuilder4\jdk1.3\jre\lib\rt.jar;C:\JBuilder4\jdk1.3\jre\lib\sunrsasign.jar;C:\JBuilder4\jdk1.3\lib\dt.jar;C:\JBuilder4\jdk1.3\lib\tools.jar" unittest.RunUnitTests JdbcOdbcObj

JdbcOdbcObj Class Unit Test Results ...

testCase1 Starts

```
Exist Dead Link? = false
Exist Foreign Link? = false
Exist Normal Link? = false
Exist Leaf Link? = false

testCase1 Finishs


End of Unit Test.
```

### 7.2.4 Explanation

The above example shows that how a unit test case can be run automatically every time when IDE compiles the source code. We can see that unit test is embedded into IDE environment and we can check the semantic error along with syntax error. We build up a test frame and can put as many test cases as we like into specific unit test class (ex. JDBCDemo). And I modified Bruce Eckle RunUnitTest source code to let it to accept multiple application parameters to run multiple target classes.

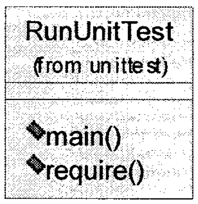


**Figure 13 RunUnitTest Class**

The source code RunUnitTest.java (See attached source code) shows how all test cases of all classes needed test are executed in a main function.

All the classes should have nested class as Unit test container and these nested classes are inherited from Unit test class:

```
package unittest;
import java.util.*;

public class UnitTest {
  static String testID;
  static List errors = new ArrayList();

  // Override cleanup() if test object
  // creation allocates non-memory
  // resources that must be cleaned up:
  protected void cleanup() {}
  // Verify the truth of a condition:
  protected final void affirm(boolean condition, String errorMsg){
    if(!condition)
      errors.add(testID + " failed: " + errorMsg);

  }
} ///:~
```

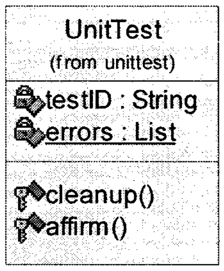


**Figure 14 Unit Test Class**

The "errors" is attribute for all the error messages that might be caught during unit test. Like compiler error message that let user easily identify what type of errors occur, unit test frame should have same mechanism.

Since all the test cases (code) are executed by this "Extend compiler" automatically, the developer can modify his code with more confidence especially in a large project. That way, the errors can be caught in the early stage.

## 7.3 Test driven programming

### 7.3.1 System Test Level

A good habit is that before coding and after design phase, properly document the test plan, and write the test cases based on the requirement. Then between requirement and test, we have good traceability. This is part of standard process recognized by CMM institute -- Capability Maturity Model® for Software (SW-CMM®)[6]. This is system test level test-driven programming.

### 7.3.2 Unit Test Level

We can go further with Extreme Programming in detail design phase, document the test cases for each class and develop these test code first and then functional code.

There are some benefits from high priority test code according to Bruce Eckle:

1. *Describe what the code is supposed to do, not with some external graphical tool but with code that actually lays the specification down in concrete, verifiable terms.*
2. *Provide an example of how the code should be used; again, this is a working, tested example, normally showing all the important method calls, rather than just an academic description of a library.*
3. *Provide a way to verify when the code is finished (when all the tests run correctly).*

I would like to mention the second point. The example of how code used is part of good on-line documentation and at same time it also can be a good API document for another developer. Maybe one day Javadoc [7] will incorporate this test code as part of javadoc documentation. MSDN normally provides the function example in its help file, and Javadoc could have the same example mechanism.

# 8 System Test (Including Installation Guideline)

## 8.1 System Requirement

Before trying to install and running WSA tool, computer system needs meet following requirements.

### 8.1.1 Hardware

Pogrom shall operate with the following hardware requirements:

- CPU 486 or later
- Monitor – SVGA (800x600) or latter
- RAM – 16 MB
- Disk Free Space 16M or more
- Mouse or equivalent pointing device

### 8.1.2 Software

Following software should be setup in the machine

- JDK 1.3 or later
- Microsoft Access 98 or later
- Tomcat 3.2.4 or later

### 8.1.3 **Platform**

The program can run on the following platforms

- Windows 98
- Windows NT
- Windows 2000

## 8.2 Setup and Installation

Step1: Installing Jakarta-Tomcat, if it is already in the machine, go to step 2.

- Download Jakarta-Tomcat from:

  http://jakarta.apache.org/builds/jakarta-tomcat/release/v3.2.4/bin/jakarta-tomcat-3.2.4.zip.

- Unzipped it to Drive D:\ and new folder D:\jakarta-tomcat-3.2.4 will appear.

Step2: Copy WSA files:

File location should be:

| ***Directory*** | ***Files*** |
|---|---|
| D:\jakarta-tomcat-3.2\webapps\ROOT | JSP, HTML, DB file |
| D:\jakarta-tomcat-3.2\webapps\ROOT\WEB-INF\classes | classes files |
| D:\jakarta-tomcat-3.2\webapps\ROOT\images | image files |

Step3: Set windows system environment variables

- Run start->settings->control panel, control panel window appears.
- Click 'system', and 'System Properties' window appears. Go to “advance” tab

- Click 'Environment variable' button, go to “System Variable”. (or User Variable if no administration rights, but environment variables should be reset every time when re-login).
- Set the jdk home: type 'JAVA_HOME' in 'Variable' field, type 'd:\jdk1.3' (or jdk directory in PC) in 'Value' field and click 'Set' button to add.
- Set Tomcat home: type 'TOMCAT_HOME' in 'Variable' field, type “D:\jakarta-tomcat-3.2.4” (Tomcat directory) in 'Value' field and click 'Set' button to add.
- Set PATH: click 'Path' on the 'System Variables" zone (or (or User Variable if no administration rights), append '; %JAVA_HOME%\bin' to the Value string, and click 'Set' button to make change.

Step4: DSN Setup (JDBC- ODBC)

- Start→ setting→ control panel→ administrative tools→ data source (ODBC)
- Set DSN name as URLs, and path as D:\jakarta-tomcat-3.2\webapps\ROOT\URLs.mdb

Step5: Start Tomcat Server

- Go to D:\jakarta-tomcat-3.2.4\bin, execute startup.bat.
- Open the browser, and type http://localhost:8080/ (Tomcat has its own HTTP server which listens on port 8080: default value).
- Go to http://localhost:8080/examples/jsp/num/numguess.jsp to test if Tomcat server works fine.
- Step6: Run WSA

- Run http://localhost:8080/MainPage.jsp , the user interface will show up.

## 8.3 Analyze a website

After successful installing and starting WSA, the main page of WSA appears. User is required to input an URL (Uniform Resource Locator) into the text box in the top frame, and then press the Submit button to start searching.

The format of URL that WSA can analysis includes HTML, ASP, JSP, PHP pages and folders. The example of URL could be:

http://www.cs.concordia.ca

http://www.cs.concordia.ca/programs/grad/diploma/courses.html

http://www.newatlanta.com/products/servletexec/index.jsp

http://www.cs.concordia.ca/~faculty/grogono/

https://www.mywebsite.org

www.mail.yahoo.com

When inputting the URL, the usual protocol http:// may be omitted. It is the default protocol of WSA. But the https:// cannot be omitted. After clinking on the submit button, it may take a few seconds or even several hours to analysis the website and save the results into the database. It depends on the size of the website being analyzed.

Once the progress bar on the right bottom of explorer shows that the search is finished, user can click on Refresh button on the top of explorer to update the search result in the tree menu. Clicking on the + or – can span or collapse the folder to navigate the tree.

One of the advantage of WSA is it allows user to partly analyze a website. User can Stop the analyze process any time they want by clicking on the icon Stop on the top of explorer. The results obtained so far will be stored in the database safely and displayed in the report tree correctly.

## 8.4 Test Cases and Experimental Results

In order to test the performance and capability of this tool, we choose various range of websites to test in term of size, from hundreds links size website to huge website engines. The experimentation proved that WSA could adapt small site as well as big size of web site with fast performance. One of the advantages of it is for very huge website, WSA allow user to partly analyze the website, that means user can stop the searching process anytime by clicking stop button in explore. The results will exactly show the results of the finished part robustly.

1. http://www.cs.concordia.ca/programs/grad/diploma/courses.html

   Size: 476,　　　　Time: few seconds

2. http://www.cs.concordia.ca/~faculty/grogono/

   Size: 4082　　　　Time: 2 minutes

3. http://www.yahoo.com

   Since yahoo is a web sites engine, it is impossible to finish in short periods. We stop the test after 10 minutes and get the result:

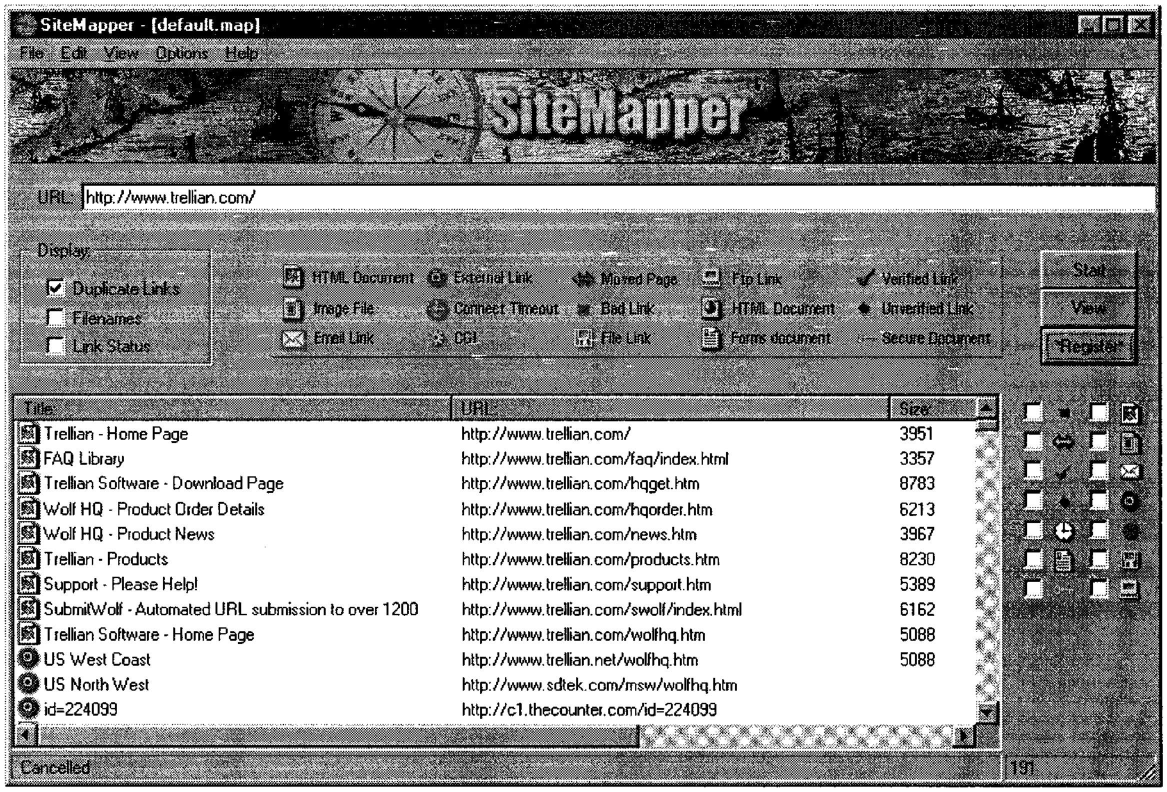
   Size: around 10,000　　　　Time: 10 minutes

# 9 Commercial Product

There are many existing commercial Web Site analyzer tools. Some of them analyze the contents of a web site by giving the detailed map with an indexed listing of all resources by page and category; some of them analyze the structure; some of them analyze the traffic of the web site for commercial propose. In this section, we just compare our work with Sitemapper product.

Site Mapper is a commercial Website Analyzer tool developed by company Trellian, it analyze the contents of a website, and create a detailed map with an indexed listing of all resources by page and category.

The below figures shows the main user interface of SiteMapper and an example report of Sitemapper:



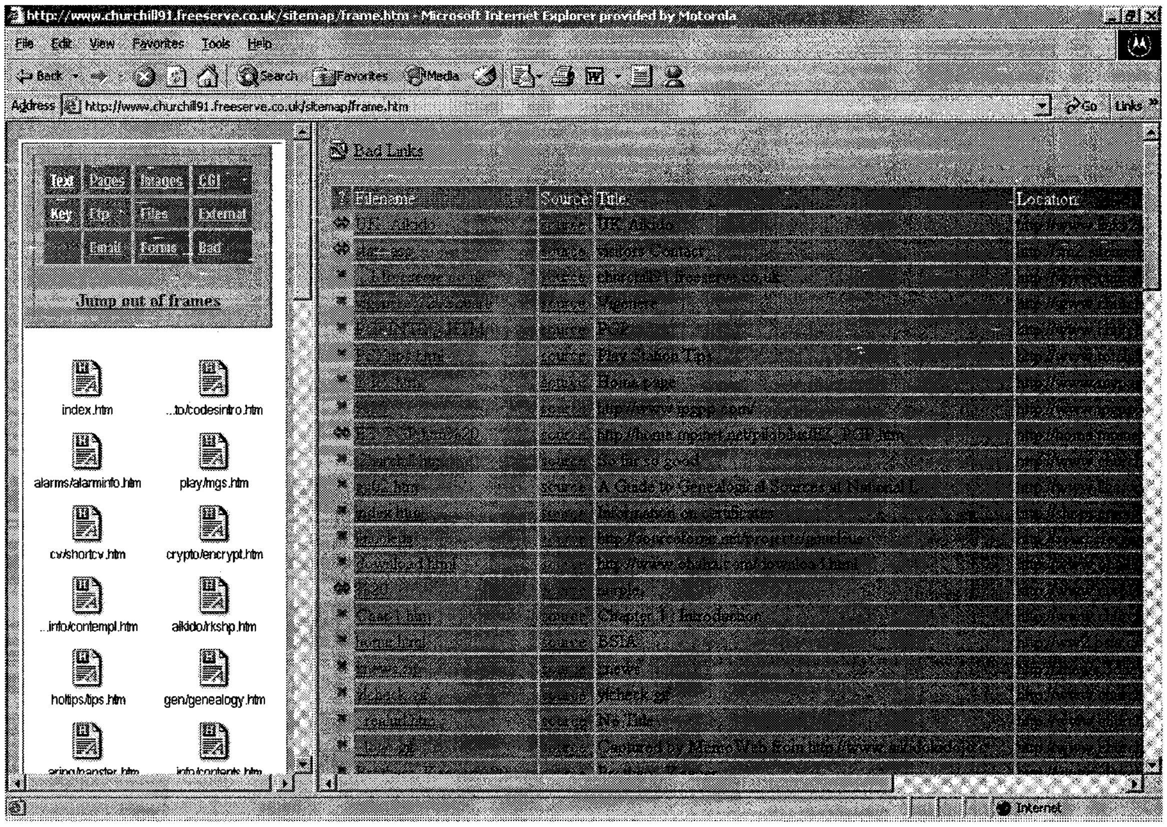
URL: http://www.trellian.com/

Display:
[✓ Duplicate Links]
Filenames
Link Status

| Title | URL | Size |
|---|---|---|
| Trellian - Home Page | http://www.trellian.com/ | 3951 |
| FAQ Library | http://www.trellian.com/faq/index.html | 3357 |
| Trellian Software - Download Page | http://www.trellian.com/hqget.htm | 8783 |
| Wolf HQ - Product Order Details | http://www.trellian.com/hqorder.htm | 6213 |
| Wolf HQ - Product News | http://www.trellian.com/news.htm | 3967 |
| Trellian - Products | http://www.trellian.com/products.htm | 8230 |
| Support - Please Help! | http://www.trellian.com/support.htm | 5389 |
| SubmitWolf - Automated URL submission to over 1200 | http://www.trellian.com/swolf/index.html | 6162 |
| Trellian Software - Home Page | http://www.trellian.com/wolfhq.htm | 5088 |
| US West Coast | http://www.trellian.net/wolfhq.htm | 5088 |
| US North West | http://www.sdtek.com/msw/wolfhq.htm | |
| id=224099 | http://c1.thecounter.com/id=224099 | |

Cancelled 191

**Figure 15 SiteMapper: User Interface**



Bad Links

| ? | Filename | Source | Title | Location |
|---|---|---|---|---|
| ⇔ | UK_Aikido | source | UK Aikido | [illegible] |
| ⇔ | [illegible] | source | seniors Contact | [illegible] |
| ✱ | [illegible] | source | churchill91.freeserve.co.uk | [illegible] |
| ✱ | [illegible] | source | Vigenere | [illegible] |
| ✱ | [illegible] | source | PGP | [illegible] |
| ✱ | PSTips.html | source | Play Station Tips | [illegible] |
| ✱ | [illegible] | source | Home page | [illegible] |
| ✱ | [illegible] | source | http://www.pgpp.com/ | [illegible] |
| ⇔ | [illegible] | source | http://home.mpinet.net/pigbblue/BZ_PGP.htm | [illegible] |
| ✱ | [illegible] | source | So far so good | [illegible] |
| ✱ | [illegible] | source | A Guide to Genealogical Sources at National L | [illegible] |
| ✱ | [illegible] | source | Information on certificates | [illegible] |
| ✱ | [illegible] | source | http://sourceforge.net/projects/gnucleus | [illegible] |
| ✱ | download.html | source | http://www.[illegible]/download.html | [illegible] |
| ⇔ | [illegible] | source | surplus | [illegible] |
| ✱ | Chap1.htm | source | Chapter 1 : Introduction | [illegible] |
| ✱ | home.html | source | BSIA | [illegible] |
| ✱ | [illegible] | source | news | [illegible] |
| ✱ | [illegible] | source | ylcheck.gif | [illegible] |
| ✱ | [illegible] | source | No Title | [illegible] |
| ✱ | [illegible] | source | Captured by MemoWeb from http://www.aikidokai.[illegible] | [illegible] |

Figure 16 SiteMapper: Report Interface

We compare this product to our application and identified the advantages and disadvantages of sitemapper, which respectively are WSA's weak and strength.

## 9.1 Advantages of Sitemapper

These advantages information are from Site Mapper's home page [9]:

- Spell check documents as they are mapped
- Better interface
- Built in document preview functionality
- Save files to local directory

- View file properties (includes list of site URL referrers).
- Proxy and Firewall Support

## 9.2 Disadvantages of Sitemapper

- Because the reports lists are shown in the flat indexed listing of all resources, the end user may not easily grasp the file hierarchy structure of the analyzed website.
- Since Mapper is a commercial Website Analyzer tool, the price of SiteMapper2.0 electronic download version is $39.95 US.

# 10 Conclusion and Future work

## 10.1 Conclusion

This major report emphasizes the aspect of development methodologies, which are part of extreme programming practices. Pair programming and test driven programming are two valuable experiences I get from this project.

WSA is a Website analysis tool with user friendly and high performance. Its "all in once" use interface maximum simplify user's operation, as well as presents the idea of "what you see is what you get". Users don't need to learn how to use it.

It demonstrates an example of JSP three-tier web applications design methodology. The main advantage is to hide the cooperative business logic from the user interface presentation.

In this tool, the user input and presentation output can work independently. The user can input and perform search a URL once, and analyze the result as many times as he/she wishes without redoing the search again. This also allows the off line analysis. The reason for that is because the search results are stored in the database when user input the URL. After the search is finished, the application goes to the database to get the results and load them to the tree menu database.

## 10.2 Future works

1. There is a gap between user input and application output in this tool, that is after user input the desire URL, the search results will be loaded in to the database. After it finished to load the database, user need to press *Refresh* icon in the top of the IE to trigger application to read database and refresh the tree menu. A suggestion solution is to implement a function behind the scene that audit the status of database loading and automatically trigger the application to read database and refresh the tree menu.

2. In this tool, the Orphan links have not been searched and reported yet. Our traverse algorithm is started from a base link, which is input by user. After parsing this base page, we get all the links referenced by this page, and record them into database. The process continues to apply the links found in the previous search until no link exists in last searched page. Since no page refers to orphan link, our algorithm cannot reach it.

3. As we mentioned in our design, to simplify the interface, we designed a "All in One" interface which means the user input and application output are all built in a frameset web page. But everything is a tradeoff, the shortage of this design might be for very large websites, it may take very long time to perform the search, though there is a progress bar showing the status in the right bottom which built in by IE browser, but

it is not obvious, user may not be informed that how many times it will take and how many percentage exactly been finished.

4. It would be nice to provide the ability to Pause and Resume the analysis.

## 11 References

[1] Yuan Xu, "WEB SITE ANALYZER – A Graduate Major Report of Concordia University," 2003

[2] Alistair Cockburn and Laurie Williams, "The Costs and Benefits of Pair Programming," presented in Humans and Technology Technical Report, Jan. 2000.

[3] Bruce Eckel, *Thinking in Patterns with Java*, Revision 0.6, Bruce Eckel's MindView, Inc: Free Electronic Book, 2001

[4] Wiki, "Extreme Programming Roadmap," in *Portland Pattern Repository*, http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap

[5] Unknown, *Junit,* http://www.junit.org/index.htm

[6] S.L. Pflegger, *Software Engineering: Theory and Practice*. Upper Saddle River, NJ: Prentice Hall, 1998.

[7] Unknown, *Javadoc*, http://java.sun.com/j2se/javadoc/, by Sun Microsystems, Inc.

[8] Peter Grogono, "Website Requirement" Concordia University, 2001, http://www.cs.concordia.ca/~faculty/grogono/webreqs.pdf.

[9] Unknown, " SiteMapper", http://www.trellian.com/mapper/index.html, published by Trellian Inc.