Tunghai Science Vol. 3: 97–112
July, 2001

Validating Hyperlinks by the Mobile-Agent Approach

Wen-Kui Chang* Min-Hsiang Chuang*

Abstract

In this paper, we first lay emphasis on the importance of hyperlink validation. Due to the variety of the structures and the sizes of today's websites, validating hyperlinks has become quite a difficult task. Therefore, the mobile agent technology is introduced as a solution for the critical problem of limited bandwidth we face nowadays. In this research, we survey on different approaches and characteristics that a mobile agent can inherit, so as to achieve better efficiency and performance.

We will first introduce the concepts of the mobile agent technology, and then look at some related researches on the mobile agent technology. Next we will study how we can arrange our online information with different types of navigation structures. In addition, we will compare the browsing algorithms, which depend on the nature of current web sites.

Last but not least, we will combine all the verdicts from different sections of this paper and come out with the actual mobile agent for validating the hyperlinks. We will also discuss about the expected output and finally, by comparing our approach with current methods to provide evidence for our choice. We also wish to extend mobile agent technology to software quality assurance in the future.

Keywords: hyperlink validation, mobile agent, navigation structure, software quality assurance

1. Introduction

The web has become something very common in the life today. Everything nowadays can be put on the web. With all kinds of web page editors available with or without additional cost, making web pages is as easy as flipping the pages of a book. Almost all organizations, whether big or small, have websites of their own, and many people are building their own homepages for various purposes. Different people have different ways in organizing the data on web pages, and using hyperlink is the only way allowing us to do so.

Thus, hyperlinks are actually very important, especially for companies that are doing their

* Department of Computer Sciences & Information Engineering, Tunghai University, Taichung 407, TAIWAN

business online. Dead links mean no business. Since most companies have huge scaled websites, it will be quite impossible for them to manually look at each and every link in their sites everyday. Accordingly, by creating a tool for validating the hyperlinks and perhaps monitoring them as well, the companies can save a lot of time and cost [6].

There are already so many different kinds of software that do the hyperlinks validation [4], why the mobile agent approach? The shortest possible way to answer this is the bandwidth. In general, most of the current validating techniques used require the servers to be connected to the targeted clients all the time while the test is going on. Consequently, it will result in the bandwidth of the testing server to be taken up totally, which may effectively diminish the performance of the server. By utilizing the agent technology, in other words, the communication bandwidth can be saved up for other more essential applications.

Other than just implementing a mobile agent for testing purpose, we actually look into several aspects that will produce a better agent. First we look into the way in which the agent should perform the test. We compare the pros and cons of different browsing methods and come out with the most suitable method for our purpose.

In this research, we develop a mobile agent with the purpose of validating the correctness of hyperlinks in a web site and having the discussed properties.

2. Mobile agent technology in a nutshell

2.1 Basic structure

In essence, a software agent is a computational entity that acts on behalf of others. It is also autonomous, proactive, and reactive, and exhibits capabilities to learn, cooperate, and move [2].

Furthermore, considering a mobile agent as a moving software agent, it may generally consist of the following components as illustrated in Fig. 1.

Itinerary – This is the place where the route of the agent is recorded. Not only does it record the route, it also records the current position, so that we know exactly where the agent is. This is an essential part of a mobile agent since the agent is moving, it needs to know where to start and where to go next and ultimately, where to end. Besides, we also have to know its current position so that we are able to trace it and thus it will not get lost.

Code – This is the part where fragments of the program code are stored. By definition, an agent is a computational entity, which also means it is a program that can be executed. Therefore, this is the part that cannot be neglected. Without this code part, an agent will not be able to perform any tasks, not even traveling. Moreover, this part controls all other parts of the agent, making it more important than it seems.

State – Status of the agent is recorded here. In our case it means the status of the tested hyperlinks will be recorded here. With this part of the agent, the agent then will be able to report to both the server and client about the status of the hyperlinks.

Host – This is the place where the server position will be stored. It is quite vital since an agent is only running on the Agent Transfer Protocol (ATP) [9]. The agent will have to remember where it came from so that it will be able to return to the server after the tasks assigned are completed. If not, it will be lost in the network and perhaps jamming up the network.

Other necessary details – The agent needs to show who created it, and this is the place to put it. Other information related to this agent is stored in the same place so that people will know what this agent does and who the owner is.

In addition, there are quite a number of different interfaces for implementing mobile agents today, and the most profound ones are Concordia by Mitsubishi Electric Information Technology Center America (ITA) [5] and IBM Aglets [9]. In this research, we choose Aglets due to its availability.

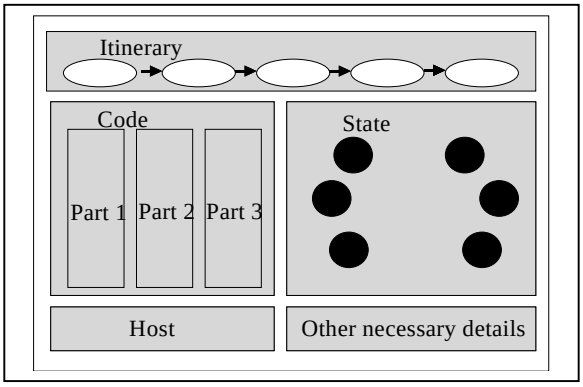

Figure 1. The basic structure of a mobile agent

2.2 Related research

Though mobile agent is quite a new technology, it has been utilized in several areas in computer science. We will look at the related researches on mobile agent for network management [2], and the NASA human planetary exploration plan [12].

The authors of the research note on mobile agent for network management [2] realized that mobile agent framework is an emerging standard as early as 1998, and they tried to utilize it in the network management area. Basically they categorized the basic network management into three different management areas, namely fault management, configuration management, and performance management. They wanted to use mobile agents in the areas mentioned above, so as to automate what we would normally do manually. The ultimate goal is to have a plug-and-play network that will automatically configure itself under different circumstances. This is also what the new technologies are hoping to achieve recently.

Next is the NASA human planetary exploration plan. Future NASA missions will require robust and flexible Web-based information systems that designers can rapidly develop, extend, and customize. Therefore they adopt agent-oriented software engineering and agent components to allow them to achieve it. A successful agent-based system can enhance human performance during planetary exploration in hostile environments. This is a very specialized research on agent systems.

3. Navigation structures of a web site

There are many ways to create a website. If we do try to categorize, we will have the following ways of organizing a website [7,10].

3.1 Waterfall structure

As the name implies, the structure is like the waterfall. It falls from the very first page to the very last page. The traversal direction is one way only. This is considered as one of the simplest ways of building a web page. The waterfall structure is illustrated in Fig. 2.

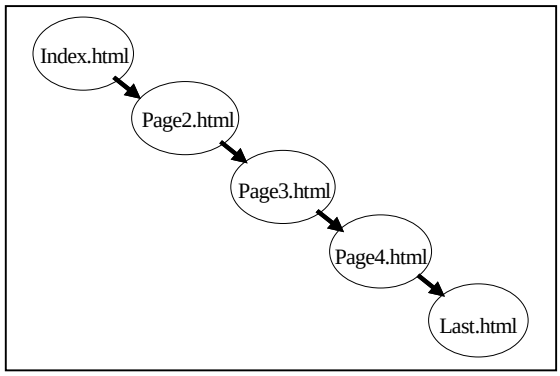

Figure 2. The waterfall structure

3.2 Radiation structure

Radiation implies spreading out from one single point to all directions. This is how websites of this category look like. With the index page as the center, the hyperlinks spread out into all directions. Normally the hyperlinks are bi-directional to increase the accessibility of the site. Fig. 3 shows how a radiated website looks like normally

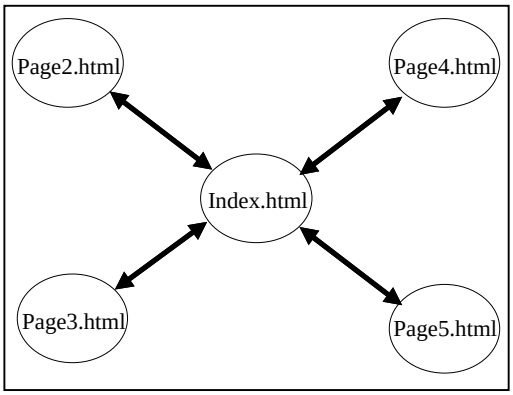

Figure 3. The radiation structure

3.3 Web/interconnected structure

This structure does look like a spider web, and therefore it is called the web structure. Every page in the site is connected to every other page. A user who wants to browse the site will be able to go from any page to all other pages. This structure is considered as quite popular as people are looking for accessibility nowadays. However, this is limited to small-scaled websites

most of the time as once the site gets big, the structure will be too messy to manage. Fig. 4 shows a web/interconnected structure clearly.

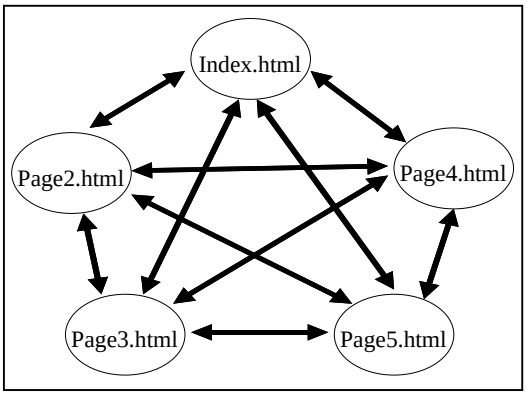

Figure 4. The web/interconnected structure

3.4 Hybrid structure

There is always a so-called best solution for everything, and the hybrid structure is the solution. As we can tell, this structure is the combination of all the possible and suitable structures. If certain parts of a website need to be inter-linked, then the navigation structure can be used. If there are places where only simple structure is needed, then maybe the waterfall structure will be quite useful.

This is actually how modern-day websites are organized. No one can really arrange her or his website fully in one single way. This seems to be the best arrangement for now. Fig. 5 roughly shows how a hybrid structure will be like. It combines only a web/interconnected structure and a waterfall structure.

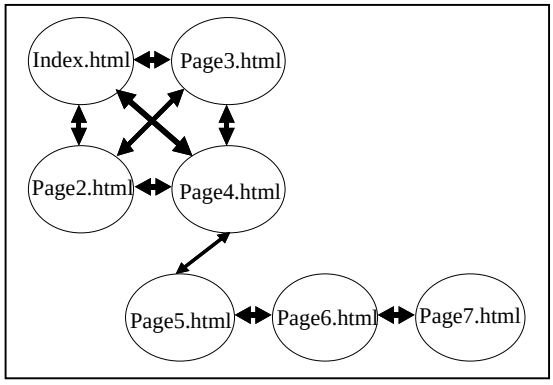

Figure 5. The hybrid structure

4. Discussion of browsing methods

With so many browsing algorithms [14], such as Breadth First Search (BFS), Depth First Search (DFS), DFS Iterative Deepening, Iterative Broadening, British Museum, Hill Climbing, and so on, we have to come out with one single approach that can be utilized for most navigation structures. This is what we are discussing in this section. We will only consider two of the most popular methods, namely the Breadth First Search and Depth First Search. The reason behind this is that both Breadth First Search and Depth First Search are considered as the primary browsing methods. Most of the other browsing methods are the enhanced versions of them.

4.1 The Breadth First Search (BFS) method

The Breadth First Search method [11] is a browsing algorithm that considers neighbors of a vertex, that is, outgoing edges of the vertex's predecessor in the search, before any outgoing edges of the vertex. Extremes are searched last. This is typically implemented with a queue. The idea behind Breadth First Search is illustrated in Fig. 6.

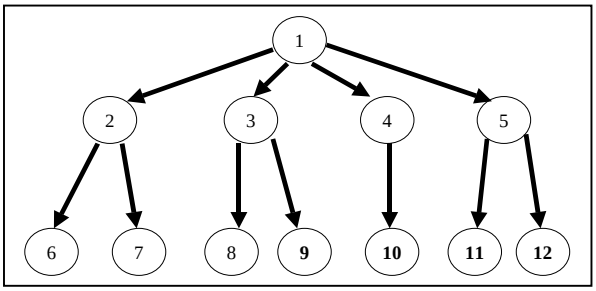

Figure 6. The Breadth First Search method

4.2 The Depth First Search (DFS) method

The Depth First Search method [11] is a browsing algorithm that considers outgoing edges of a vertex before any neighbors of the vertex, that is, outgoing edges of the vertex's predecessor in the search. Extremes are searched first. This is easily implemented with recursion. It is also an algorithm that marks all vertices in a directed graph in the order they are discovered and finished,

partitioning the graph into a forest. Fig. 7 roughly describes Depth First Search.

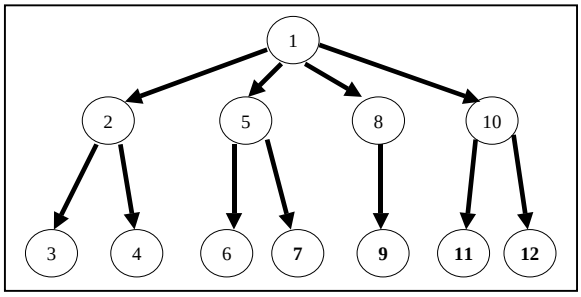

Figure 7. The Depth First Search method

4.3 The verdict

Due to the nature of most websites, we decide to use Depth First Search in our mobile agent. We are not really concerned about the width of the site. Instead, we are more afraid of the depth of the site. Generally people will not arrange their sites in such a manner that they have an indefinitely wide single layer. That is why we will need to apply depth control to our testing agents so that we will not be worried about the possibility of indefinite testing. Furthermore, if we adapt the Depth First Search, we are able to organize the data in such order that we can reuse them when we want to represent the actual layout of the website in the tree structure discussed above.

5. Using a mobile agent for validating hyperlinks

With the use of a mobile agent, we are able to decrease greatly the bandwidth needed for hyperlink validation. The mobile agent is able to perform the test on its own on the client side without having the server connected to it. Hence, the server is able to accept other requests at the same time, which significantly enhanced the utilizing rate of the server.

5.1 Proposed validating process

The proposed validating processes proceeds by the following steps as shown in Fig. 8:

(1) Initialization

The target website initiates the test by accessing the web page of the agent server. Several settings can be obtained here to aid the validating process, such as the depth of search.

(2) Dispatch

The server will then create an agent with the specification obtained earlier, and send the agent out. After the agent has been sent out, it is on its own. The agent will be responsible for validating the hyperlinks of the requested site. During the validation process, the agent needs not to be connected to the server.

(3) Testing

The agent will first visit the root of the targeted web page, and then it will start the testing process. This step is recursive, and this is the place that controls have to be enforced so that the test will not go on indefinitely.

(4) Report

The agent will report the status of the hyperlinks to both the client and server if required. It will basically carry back the information obtained from the validation. If the agent is not required report to either side, then it will be disposed after a certain period of time so that it will not take up the CPU resources.

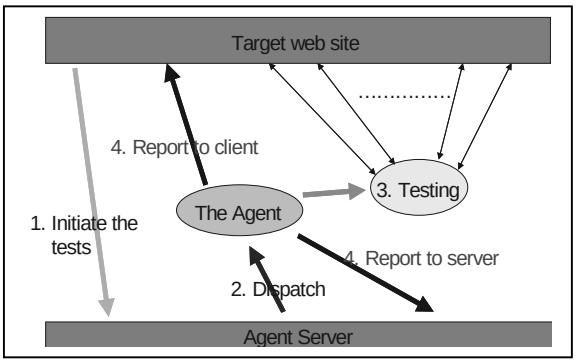

Figure 8. The proposed validating process

The client and server can do several things with the status report. They can analyze it and determine what kind of navigation structure the site belongs to, and they can even suggest how to make the site to be more efficient. The status report is the final product of our testing agents.

5.2 Implementation details

At first, the agent server needs to be set up. The Aglets agent server is called *Tahiti* [9]. It is also an agent viewer, which will let us know whether the agent has been sent out and back or not. With it we can also create our very own mobile agent.

Then we will take a look at what functions this agent is supposed to have. Since it is for validating hyperlinks, then it will definitely have the testing function. On the other hand, we need to employ the controlled testing to avoid testing to infinity. Accordingly, we will have the test-control components.

If we want to send an agent to a certain place, we use the following code [9].

```
public synchronized void startTrip(Message msg) {
    String destination = (String)msg.getArg();
        try {
            itinerary.go(destination, "");
            } catch (Exception ex) {
                ex.printStackTrace();
              } //end of catch
  } //end
```

The *destination* field is the target. Since mobile agents run on the Agent Transfer Protocol, most of the time the *destination* will be something like **atp://your.name.com**. As described previously, we can see that the *itinerary* controls the movement of an agent. This is a simple part of the implementation as there are many source codes available online.

Next is the most critical part of a testing agent – the hyperlink validation. No one really knows in advance how many links there are in a website, and therefore there is no definite number of agents that need to be sent out to completely cover the hyperlinks in a site. Therefore, we decide to send one single agent out to the destination. The way we perform the test is by checking the local side for the existence of the file. Normal hyperlink validation relies on the HTTP reply. For example, when we cannot get the HTTP response in time, we will encounter the HTTP 404 error. That indicates that the requested page cannot be displayed. However, in most cases, that does not mean that the server is down. Sometimes the requested pages cannot be displayed simply because the bandwidth is too small, and therefore the data cannot reach the

clients in time. With respect to this point, our testing strategy will be able to stand out.

Since most of the websites have their web servers running on the machine that they store their HTML files and related materials, having an agent to check the availabilities of the files will certainly tell us whether the requested pages can be displayed or not. If the file does not exist, surely the page cannot be displayed. On the other hand, if the page cannot be displayed due to heavy Internet traffic or other factors, it does not mean that the site is down, and therefore we cannot invalidate the link.

Next, we have to define our end-conditions for the agents. There are two end-conditions in our case. First is when all the hyperlinks have been tested. However, this is usually not impossible, and therefore we need to look at the next one. The next end-condition will be the depth of a test.

5.3 Demonstration example

We will look at one example to illustrate this clearly. Without loss of generality, we take the website from our Software Engineering Laboratory (SEL) as an example. Fig. 9 shows the skeleton structure of the homepage of our laboratory.

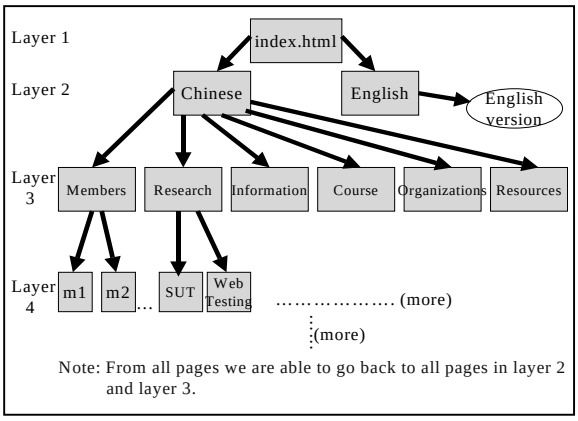

Figure 9. The homepage structure of the demonstration example

Obviously, even though the homepage of our laboratory is considered as small-scaled, it is still quite complex. However, we are still able to categorize all pages into different layers as shown in the Fig. 9. So by applying the second end-condition – the depth of a test, we are able to come out with a result rather than letting the agents run until the network is jammed up. For example, if we set the depth to the layer 3, with the first page (index.html) as layer 1, we will

have a total of 14 hyperlinks to be tested.

5.4 The expected results

We traverse and evaluate the hyperlinks to see whether they are alive or not. In summary, the results include mainly the status of all links as shown in Table 1, reporting to both the client and the server.

Table 1. The status report produced by the hyperlink validation process

PAGE NODE	STATUS
Index.html	Alive
Chinese.html	Alive
English.html	Dead
Members.html	Alive
M1.html	Alive
M2.html	Alive

What is more significant, since we have traversed all or the interested parts of a website in detail, then we are able to draw out the architectural structure of the website like Fig. 9. This may interest and benefit the client as he can decide on upgrading the site according to the browsing architecture.

5.5 Comparison with normal hyperlink validation

It will be very helpful if we compare the performance of our proposed method for validating hyperlinks with the conventional ways. Therefore, we decide to use WebLoad Version 4.6 by RadView Software [13] and LinkRunner by Alert Internet Tools [1] as the target for comparison.

First, we will look into the ease of installing all three software is as simple as clicking a few buttons. Windows will automatically set up the environment variables suitable for the software execution. Next we will look at the ease of activation for the concerned software products. WebLoad so far is the user-friendliest software of the three. Actually it is rather unfair to compare two tools with the IBM Aglets Software Development Kit (ASDK) [9] since ASDK is a development platform. However, we still can compare them if we really want to. WebLoad requires little effort in activating the software, while LinkRunner is actually quite difficult to set up and run. ASDK, on the other hand, only requires one-time effort in setting up the correct environment variables in the beginning, and we can activate mobile agents developed with it

without extra efforts from next time onwards. Therefore ASDK is also considered as easy to use.

As for the speed of testing, both WebLoad and the proposed mobile agent approach can be considered as fast. Both applications are able to return the results within five minutes while testing the SEL homepage. However, using LinkRunner to perform the validation process takes quite some time as compared to the other two approaches.

Also, we will look at the robustness of the three approaches for hyperlink validations. The mobile agent approach clearly out-performs the other two approaches. This has got to do with the characteristics of agents. Agents are supposed to be able to automatically perform tasks and to be robust. If we encounter network problems, such as network failures, while we are performing the validation, processes carried out by WebLoad and LinkRunner will halt immediately. We will have to restart the processes if we have not gotten the expected results. On the other hand, if we are using mobile agent to perform the validation, the agent will still be able to carry out the process on its own. It will only need to connect to the host and return if required and if the network has recovered.

Last but not least, we will look at the bandwidth utilization by both approaches. When we are using both WebLoad and LinkRunner to test a certain website, we can see that there are constant connectivity between the targeted website and the terminal that we started the applications on. The length of connectivity for both WebLoad and LinkRunner equals to the length of testing period. If the website is small, the connectivity period can be only a few minutes. On the other hand, if the website is too huge, the time of connectivity can even last to a few hours. Once we switch to the mobile agent approach, we see different things. We only notice a short period of connectivity during the time that the agent is sent out and returned. Under most circumstances, this period of time more or less equals to 5 seconds, including the time for the agent to be compiled during creation and disposed upon returning to the server. In addition, we are not able to run two simultaneous testing with single activation of WebLoad and LinkRunner. We have to either activate another new executable or wait until the validation is done, to be able to test another website. However, we are able to launch several mobile agents to test different websites with single activation of the Tahiti server.

The above comparisons can be summarized into the Table 2.

Table 2. The comparison table of the related hyperlink validation approaches

Characteristics	RadView WebLoad Version 4.6	Alert LinkRunner	Proposed Mobile agent
Installation	Easy	Easy	Easy
Setting up/activation	Easy	Tedious	More efforts needed only for the first time; easy for the rest
Speed of testing	Fast	Slow	Fast
Robustness	Low	Low	High
Time for connectivity	Depends on the size of the website, ranging from a few minutes to a couple of hours	Depends on the size of the website, ranging from a few minutes to a couple of hours	For 5 seconds when the agent is compiled and sent out and when it returns to the server to be disposed

6. Conclusions and future work

Testing has always been an issue for discussion for a long time. Good testing can save a huge amount of development time and cost. On the contrary, mobile agent technology is pretty fresh to the computing scene. Only recently did mobile agent technology start to show its potential. Mobile agent technology has been utilized in many diverse areas, ranging from network management to NASA space plans. However, only in a number of papers did we see that mobile agent has actually been utilized in the field of testing. Hence there are limited resources in this area.

As for the future work, we would like to strengthen our knowledge in mobile agent technology as much as possible, and would like to extend the testing functionalities of our mobile agent to load/stress testing, performance testing, security testing, and even reliability testing. On top of that, we will want to study the security issues related to mobile agents so as to create mobile agents that are both secure and harmless to the execution environments. As we can see, there are more and more researches on mobile agent technology, and we can say that it will become a rising star in the future.

References

[1] Alert LinkRunner, homepage: http://www.alertbookmarks.com/lr/index.htm, last visited:

2001.6

[2] Bieszczad, A., Pagurek, B., and White, T. (1998) "Mobile Agent for Network Management," *IEEE Communications Surveys* **1**(1), pp. 2-9.

[3] Chang, W.-K., and Hon, S.-K. (2000) "Certifying Hypermedia Links for Internet Applications through Statistical Usage Testing," *2nd World Congress for Software Quality*, Yokohama, Japan, September 25-29.

[4] Chang, W.-K., Hon, S.-K., and Fu, C.-C. (2000) "A System Framework for Ensuring Link Validity under Web Browsing Environments," *13th Annual International Software/Internet Quality Week*, San Francisco, California USA, May 30-June 2.

[5] Concordia-Java Mobile Agent Technology, homepage: http://www.concordiaagents.com/, last visited: 2001.5.

[6] Mosley, D.J. (2000) *Client-Server Testing on the Desktop and the Web*, Prentice Hall.

[7] Lagon, D., Deb Lagon Home on the Web Page, homepage: http://www.deblogan.com/index.html, last visited: 2001.3.

[8] Nwana, H., and Ndumu, D. (1996) "An Introduction to Agent Technology," *BT Technol. J.* **14**(4), pp. 55-67.

[9] IBM Aglets Software Development Kit Home, homepage: http://www.trl.ibm.com/aglets/index.html, last visited: 2001.3.

[10] Lowe, D. (1999) *Hypermedia and the Web-An Engineering Approach*, Wendy Hall.

[11] National Institute of Standards and Technology: Dictionary of Algorithms, Data Structures, and Problems, homepage: http://hissa.nist.gov/dads/HTML, last visited: 2001.6.

[12] Griss, M.L., and Pour, G. (2001) "Accelerating Development with Agent Components," *Computer* **34**(5), pp. 37-43.

[13] Radview Software, homepage: http://www.radview.com/default.asp, last visited: 2001.6.

[14] Search methods, homepage: http://www.cs.dartmouth.edu/~brd/Teaching/AI/Lectures/Summaries/search.html, last visited: 2001.6.

利用行動代理人方法驗證網站超鏈結之正確性

張文貴* 莊閔翔*

摘 要

本文重點在超鏈結的測試驗證。隨著網路的盛行與多變化，超鏈結的測試變得越來越複雜。由於網路頻寬的限制，本研究將評估一個高效能行動代理人之特性，並進一步探討以行動代理人為基礎的測試方法以減少頻寬的使用量。

本文首先介紹行動代理人的基本概念，並且參考其他行動代理人的相關研究，接著討論不同的瀏覽架構及方法，最後我們研究提出一個用來作超鏈結測試的行動代理人，並與相關軟體工具做比較，以說明證實我們所提出方法的優點；此外，我們也希望將來能把行動代理人的技術延展到軟體品質保證方面。

關鍵詞：超鏈結驗證、行動代理人、瀏覽架構、軟體品質保證。

* 東海大學資訊工程與科學系