私 立 東 海 大 學

資 訊 工 程 與 科 學 所

碩士論文

Utilizing Mobile Agent for Web Testing to Ensure Website Quality

指導教授：張文貴 博士

研 究 生：莊閔翔

中 華 民 國 九 十 一 年 五 月

謝　　誌

感謝恩師　張文貴博士這兩年來在學業及待人處事上的關心與鼓勵，使我可以從國外的學習環境與習慣中很快的適應過來，並且讓我的人際關係更上一層樓。

在撰寫論文期間，我要感謝老師不辭辛勞，一個字一個字慢慢地檢查我的論文，並給予我在論文寫作上許多的指導與建議； 而在學期間，我要感謝林清祥老師、林助興老師、朱正忠老師、蔡清欉老師、呂芳懌老師、羅文聰老師、與朱允執老師在學業上之教導及啟發，讓我得到更多的知識。雖然我因為剛回國不久，所以朋友並不多，但是還好我在學的這段期間，有同實驗室的學長咨博與昕凱給予我照顧和關懷，讓我可以快快樂樂地過完這兩年。

最後，我要感謝我最親愛的父母親，你們辛苦工作，只是要給我跟妹妹一個好的求學環境，雖然我們以前還年輕，不懂得珍惜，但是我們現在都長大了，都很感謝爸媽的辛苦。謹以此論文獻給我最親愛的父母與妹妹，願與他們分享完稿之喜悅。

謹誌於

東海大學資訊工程與科學所

中華民國九十一年七月

Abstract

In this paper, we first investigate the issues related to web testing, which is an important issue for software quality assurance. The issues include web characteristics, web testing metrics and techniques. There are many aspects included in web testing, and therefore huge bandwidth is required if we intend to fully carry out testing for websites. Thus, we introduce the concepts of mobile agent technology, and look at some related research on mobile agent technology. Mobile agent technology is employed as a solution for the critical problem of limited bandwidth we face nowadays. Finally, we combine the verdicts from different viewpoints and propose frameworks that utilize mobile agent technology for performing different aspects of web testing. In addition, we discuss the advantages of the suggest frameworks, such as better bandwidth utilization and improved robustness, and provide several practical demonstration examples to support the underlying rationale.

Keywords: software quality assurance, web testing, mobile agent technology, communication bandwidth, robustness

Index

Figure Index

Table Index

1. Introduction

The World Wide Web (WWW) has become more and more popular and common around the world. It is even more common than television sets in some areas. Therefore, it has a very wide range and deep reach in the human society. Furthermore, with so many web editors available with little cost or even free of charge, making websites are getting easier as time goes by. Not only do people have their own homepages describing their own interests, but more and more companies are also building their businesses web-based. However, not all companies are successful in doing so. There are many factors contributing to this, and the website quality is the major factor. A website without quality will have fewer people visiting them and will not survive under such competitive environment.

First, we look into the issues related to web testing. Web testing can ensure that the websites are at least of some quality. Quality websites are attained through several ways, and under most circumstances, people are only concerned with factors such as the hyperlink validity, load and stress of the server, and the performance of the server. All these factors actually constitute to what we call the reliability of the websites.

In the following, we introduce the concepts of mobile agent technology. This technology is introduced so as to save the already-limited bandwidth we have today. Next we highlight the advantages, such as better bandwidth utilization and improved robustness, of the mobile agent technology, and look at some related researches on it. Finally we combine the essence from these sections to establish several frameworks for utilizing mobile agent for web testing to attain quality websites.

2. Issues related to web testing

2.1 Web environment characteristics

To access the web, you run a browser program. The browser reads documents, and can fetch documents from other sources. Information providers set up hypermedia servers which browsers can get documents from.

The browsers can, in addition, access files by FTP, NNTP (the Internet news protocol), gopher and an ever-increasing range of other methods. On top of these, if the server has search capabilities, the browsers will permit searches of documents and databases.

The documents that the browsers display are hypertext documents. Hypertext is text with pointers to other text. The browsers let you deal with the pointers in a transparent way -- select the pointer, and you are presented with the text that is pointed to.

Hypermedia is a superset of hypertext -- it is any medium with pointers to other media. This means that browsers might not display a text file, but might display images or sound or animations.

2.2 Web testing metrics

Web testing is not only concerned with the overall performance of the website, but also the different metrics that contribute to the making of a quality website. In the following paragraphs we will be looking at some of the common metrics that people will look for when they want to perform testing on the websites for quality.

2.2.1 Workload

Workload defines the amount of work that a machine can produce in a specified time period. In the case of a website, it will be the indicator of the amount of data that the web server is capable of sending and receiving. Under most circumstances, workload and stress are being put together due to their intimate connection. When we need to know the maximum load that the web server can handle, we can look at it from another viewpoint, that is, when the web server will be facing the maximum amount of stress. The workload and stress of the web server depend on numerous factors, such as the time of the day, and most importantly, the number of users. From the maximum workload and stress of the web server, we can determine the maximum number of users that the server can handle simultaneously. Once the maximum number is reached, the server will be facing a performance bottleneck. This is the place where the web server can handle with a decent performance. Once the number of users exceeds that of the bottleneck, the performance of the web server will start to fall dramatically, until it eventually stop to respond.

2.2.2 Throughput

In computer technology [18], throughput is the amount of work that a computer can do in a given time period. Historically, throughput has been a measurement of the comparative effectiveness of large commercial computers that run many programs concurrently. An early throughput measure was the number of batch jobs completed in a day. More recent measures assume a more complicated mixture of work or focus on some particular aspect of computer operation. While "cost per Million Instructions Per Second (MIPS)" provides a basis for comparing the cost of raw computing over time or by manufacturer, throughput theoretically tells you how much useful work the MIPS are producing.

Another measure of computer productivity is performance, the speed with which one or a set of batch programs run with a certain workload or how many interactive user requests are being handled with what responsiveness. The amount of time between a single interactive user requests being entered and receiving the application's response is known as response time. A benchmark can be used to measure throughput.

2.2.3 System response time

Generally, a response time is the time that elapses between a stimulus and the response to it. In a computer system, the system response time will be the time it takes for the CPU to interpret and execute commands. On the other hand, in a network environment, the response time of the network system will be the time it takes for the web server to respond to the requests, and the oblivious choice is to have a response time that is as low as possible. To achieve this, we need to plan out the network, and make sure that the response time is as predictable as possible, because research has shown that users get annoyed and frustrated when the response time from the server is irregular, which could be cause by a network that is overloaded by extra demand. One way to make sure that the response time is as predictable as often as possible is to set a minimum time before the server response to instructions from the user. For example in normal circumstances an instruction could be completed by the server in 1 second and under busy circumstances it might take 5 seconds. Now if 3 second was set for the minimum amount time before the server responses then under normal circumstances the job would be done in 1 sec and then the server has 2 sec left to do something else for a different user before releasing the data back to the first user. The end result is that the network is more predictable and reliable.

2.2.4 Security

Security requirements [18] define the security services that the system must provide and help to guide the allocation of security functionality to components. Security requirements are analyzed to determine whether they comprise a complete set and whether they adequately reflect the concerns of the customers. Requirements analysis also determines whether the functionality has been identified and allocated to a component to satisfy the requirements. Requirements, threats, architecture analysis, and penetration tests will attempt to identify how the system may be exploited and the potential impact of that exploitation.

Weaknesses are generated where no functionality exists to enforce a security requirement or where testing and analysis reveal a deficiency in the effectiveness of security mechanisms. Weaknesses against which a threat is levied are classified as vulnerabilities. Weaknesses and vulnerabilities are differentiated in order to focus and prioritize the decision maker's limited resources on items that will yield the greatest improvement in security when addressed. Finally, security guidance is offered to mitigate or eliminate vulnerabilities through the customer's decision-making processes.

There are security risks that affect web servers, the local area networks that host websites, and even innocent users of web browsers. From the point of view of the network administrator, a web server represents another potential hole in the local network's security. The general goal of network security is to keep strangers out. Yet the point of a web site is to provide the world with controlled access to the network. A poorly configured web server can punch a hole in the most carefully designed firewall system. A poorly configured firewall can make a web site impossible to use. To the end-user, web surfing feels both safe and anonymous. However, active content, such as ActiveX controls and Java applets, introduces the possibility that web browsing will introduce viruses or other malicious software into the user's system. Active content also has implications for the network administrator, insofar as web browsers provide a pathway for malicious software to bypass the firewall system and enter the local area network. Even without active content, the very act of browsing leaves an electronic record of the user's surfing history, from which dishonest individuals can reconstruct a very accurate profile of the user's tastes and habits.

Finally, both end-users and web administrators need to worry about the confidentiality of the data transmitted across the web. The TCP/IP protocol was not designed with security in mind. Hence it is vulnerable to network eavesdropping. When confidential documents are transmitted from the web server to the browser, or when the end-user sends private information back to the server inside a form, someone may be listening in.

2.2.5 Reliability

Reliability [18] has to do with the quality of measurement. In its everyday sense, reliability is the "consistency" or "repeatability" of your measures. Before we can define reliability precisely we have to lay the groundwork. First, we have to learn about the foundation of reliability, the true score theory of measurement. Along with that, we need to understand the different types of measurement error because errors in measures play a key role in degrading reliability. With this foundation, we can consider the basic theory of reliability, including a precise definition of reliability. There we will find out that we cannot calculate reliability - we can only estimate it. Because of this, there are a variety of different types of reliability that each has multiple ways to estimate reliability for that type. In the end, it's important to integrate the idea of reliability with the other major criteria for the quality of measurement - validity - and develop an understanding of the relationships between reliability and validity in measurement.

With so many metrics that we need to watch out for while we perform web testing, we are only concerned with a number of the metrics listed above. In order to measure the performance of a website, we will be focusing on the workload, throughput, and the system response time of the website. From these metrics we are able to determine the peak performance of the web server and the maximum load that it can handle. Furthermore, we are interested in the security aspects of the websites, as the security functions of a website play an important part in fortifying the website from hostile entries. Hence, we will also be putting our attention onto the security matters. Last but not the least, we need to concentrate on the reliability part of the websites as that is what a good website is all about. A reliable website obviously has to have all of the metrics, and it is the ultimate goal that today's websites are trying to achieve.

2.3 Web testing techniques

With the focused metrics in mind, we discuss several web testing techniques that will suit and amplify the importance of the metrics.

2.3.1 Load testing

Load testing is the best method to gauge client-server performance. It involves testing in an environment that may consist of a variety of software applications and hardware platforms. Load testing is necessary in order to determine the suitability of application server, database server, and web server performance.

Load testing requires emulating a milieu where multiple clients access a single-server application. This type of testing is rarely possible without automation. There are several reasons why manual load testing is not usually a viable option. First, it is tremendously difficult to amass the physical resources that are required. Second, a large number of testers are required. Third, it is very difficult to coordinate and synchronize the testers and machines. Fourth, a significant level of organization is required to capture and organize the results. Fifth, manual tests are rarely repeatable.

When we analyze the system, we need to catalog the hardware and software components, describe the system architecture, and develop and expected usage model that includes peak and heavy load periods. The next step is to define a set of common user tasks that can be grouped into load test scenarios. The tasks must be analyzed and a frequency distribution plotted that is based on frequency of occurrence for each task. The test scenarios should reflect the distribution by running the more frequently used more often than the less frequently used tasks.

Next, we should develop a set of load test objectives. The objectives should be stated in clear terms. They should precisely specify which results are acceptable and which, unacceptable. Some common objectives might include:

1. Measuring the length of time to complete an entire task.
2. Discovering which hardware/software configuration provides optimal performance.
3. Tuning database queries for optimal response.
4. Capturing Mean-Time-To-Failure as a measure of reliability.
5. Measuring system capacity to handle loads without performance degradations.
6. Identifying performance bottlenecks.

Based on the test objectives, a set of performance measurements should be described. Typical measurements included:

1. End-to-end response time
2. Network response times
3. GUI response times
4. Server response times
5. Middleware response times

The measurements can then be analyzed and related back to the load test objectives.

Decisions can be made as to what remedies, if any, are necessary in order to achieve the required system performance. Once the necessary steps have been implemented, the load test should be repeated and the results analyzed again, and this cycle should be repeated until the system meets its performance criteria.

2.3.2 Stress testing

The purpose of stress testing [18] is to show that the system has the capacity to handle large numbers if transactions during peak periods. An example of a peak period is when everyone is logging back on-line after the system has been down. In a batch environment a similar situation would exist when numerous jobs are fired up after down time. As an example, a script might require users to login and proceed with their daily activities while, at the same time, requiring that a series of workstations emulating a large number of other systems are running recorded scripts that add, change, or delete from the database, The scripts should include activities purposely designed to compete for server resources.

2.3.3 Security testing

Security testing is carried out to reveal the security loopholes in a website. Generally, no matter how well the scripts are written, they are always prone to errors due to several factors. The security functions of a website are at their weakest either during peak hours or when the server just recovered from failures. A website with good and monitored security functions will obtain higher trust than those with lower security levels. Good security testing can prevent hostile entries that not only try to obtain unauthorized materials from the website, but also even try to alter the content of it.

2.3.4 Reliability testing

Reliability is an attribute of any computer-related component that consistently performs according to its specifications. It has long been considered one of three related attributes that must be considered when making, buying, or using a computer product or component. Reliability, availability, and serviceability are considered to be important aspects for designing any system. In theory, a reliable product is totally free of technical errors. However, in practice, vendors frequently express the reliability proportion of a product as a percentage. In literature, software reliability is defined as the summation of the probabilities of the path between the initial state and the termination state that is totally free of errors [19,21,22]. In the case of website reliability, since we are only concerned with the probabilities of visiting a certain page and the page being available or not, the website reliability will be the summation of the probabilities of each navigation path that is totally free of errors.

There are several issues related to website reliability. These are the factors that will help us to determine whether a website is reliable or not. They are listed and described below in detail [7].

1. Validity

This is to check whether the website has valid information regarding the author(s). A reliable website should have the proper contact information of the author(s). A responsible author will try to make his or her website fulfill the following four credentials so as to make the website reliable, and he will not hesitate to leave the contact information.

2. Currency

This is to ensure that the website is both around for a long period of time and frequently updated. When website has been around for some time, it means that the site is very stable. And hence it indicates that the website has a higher reliability than a recently created website. Also, if a website is frequently revised and improved, it will be more reliable than one that has seldom been updated.

3. Content

Next we should look at the content of the website. A reliable website should have neither too much information nor too little of it. It should be focused on what it is about and try to present the information to a sufficient extent. A reliable website should also contain links to other relevant sites so that it will be considered as resourceful. In addition, a reliable website is the one that can be easily navigated. So the structure of the website is another issue to be considered if we are determining whether a website is reliable or not. Hence, if a website has a navigation bar and we are able to go from any point of the website to other points without much hassle, we can say that the website is reliable in the content area. Last but not least, if the website has overwhelming advertisements either on the site itself or popping up, we can conclude that the website is content-wise unreliable.

4. Purpose

The purpose of the website can also tell us whether it is reliable or not. An educational website should not try to persuade us to buy certain software or anything like that, and vice versa. Therefore, the purpose of the site should be clearly stated and it will be a fair gauge for us to determine whether the website is reliable. Furthermore, a more biased site will, of course, be considered as less reliable.

5. Accuracy

In accuracy, we are looking for precision of the information. In other words, we are looking for references. The author should note the resources that he or she used to build the website, especially the charts, graphs, and diagrams. A reliable website should credit all referred entities to their original owners instead of itself. In addition, we need to check whether the grammar and spelling in the site are correct or not. This is the easiest and most obvious way to tell whether the website is reliable or not. A reliable website should not tolerate this kind of carelessness.

The above five issues related to website reliability are the general issues that people will look for in a reliable website. However, it is quite difficult if we are going to test all the five issues. One major reason is that different people may produce various personal views on the same website with respect to the above five issues. We cannot possible produce numerous frameworks to suit different people. Hence, according to our past researches on web testing [3,4,5], we have produced a general framework that will suit most situations.

3. Trend of mobile agent technology

3.1. Agent characteristics

In essence, an agent [9,15] is a computerized or human entity that has the following properties: [1,16]

1. Autonomy - Agents operate without the direct intervention of humans or others, and have some control over their actions.
2. Social ability - Agents can interact with other agents and possibly humans.
3. Reactivity - Agents respond in a timely fashion to changes they perceive in their environment which may be a representation of the physical world, a user via a graphical user interface, a collection of other agents, the Internet, etc.
4. Pro-activeness - Agents do not simply act in response to their environment; they are able to exhibit goal-directed behavior by taking the initiative.

In addition, the agent can also be considered as an intelligent entity that possesses the ability to learn, cooperate, and move.

Considering a mobile agent as a moving software agent, it may generally consist of the following components as illustrated in Fig. 3-1.

Itinerary – This is where the route of the agent is recorded. It not only records 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. We also have to know its current position so that we can trace it and it will not get lost.

Code – This is 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. Moreover, this part controls all other parts of the agent, making it more important than it seems.

State – The 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, the agent then will be able to report to both the server and client about the status of the hyperlinks.

Host – This is where the server position is stored. It is quite vital since an agent is only running on the Agent Transfer Protocol (ATP) [8]. The agent has to remember where it came from so that it is able to return to the server after the assigned tasks are completed. If not, it will be lost in the network and perhaps jam up the network.

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

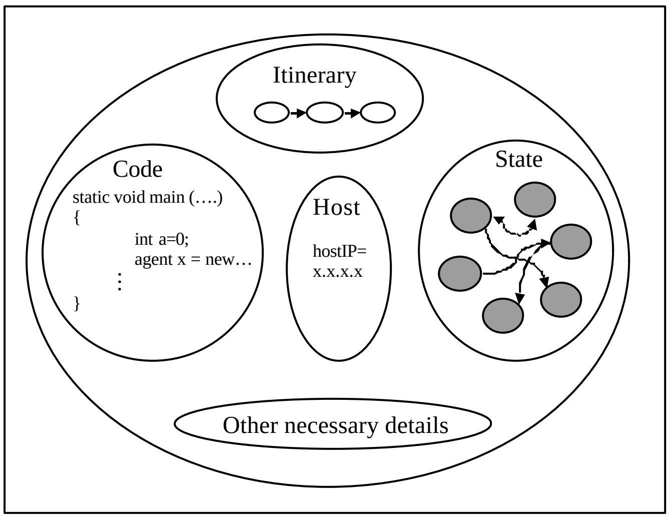


Fig. 3-1. Components of a mobile agent

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) [6] and IBM Aglets [8]. In this research, we choose Aglets due to its availability.

The Aglets Software Development Kit (ASDK) [8], developed by IBM, is a Java-based framework for implementing mobile agents. ASDK provides a network agent classloader that enables mobility of agent code, data and state information.

The aglet runtime layer provides basic functionality such as creation, dispatch and disposal of aglets. An Application Programmer Interface (API) is provided by a graphically enabled aglet server, and it is called Tahiti. Aglets use an event-driven model and event listener objects to listen for aglet mobility events. When an event occurs, a method of the corresponding eventlistener object is invoked to perform housekeeping chores either before or after the aglet movement.

3.2. Mobile agent advantages

Although mobile agent technology is rather new, it has several clear advantages over the conventional approach for designing web applications. The most common advantages [1,10,16] are listed and described below.

3.2.1 Dynamic execution

The main advantage of using the agent approach to design any system is that the agent system will be able to carry out tasks dyna mically. This idea is further enhanced by mobile agent technology. Any system or application designed using the mobile agent approach will be able to carry out tasks even more dynamically. The main reason is that the mobile agent will be able to execute under disconnected environments. Furthermore, if the present environment is breaking down due to factors other than the break down of the machine itself, the mobile agent can halt whatever it is doing at the moment and resume its execution on another computer if it has this built in function. It can also move back to the original machine once the machine is revived.

This advantage of mobile agent technology is very useful nowadays as people and enterprises are seeking ways to introduce products and services that can be used on wireless instruments such as a PDA. Therefore, we can be sure that we will be seeing more and more usage of mobile agents in the near future.

3.2.2 CPU utilization

When we talk about the CPU utilization of the mobile agent approach, we are more concerned about the CPU utilization of the agent server. In the conventional approach, if we want to perform any test, we will require the server to be running simultaneously with the client, so that the server will be able to perform any calculation if needed. In this case, when we have more than one client requesting the test, the server will not be able to handle the huge amount of CPU power needed.

On the other hand, if we switch to the mobile agent approach, the server will be able to handle several requests for tests. The reason is that once the server sends out the mobile agent that is designed to perform a certain test, the mobile agent will not have to be connected to the server. In other words, the mobile agent has the ability to carry out the tasks on its own. This is one of the basic characteristics of a software agent. In this way, the CPU utilization of the agent server will be greatly reduced.

3.2.3 Bandwidth

This is the most important advantage of mobile agent technology. Even though we have bigger and bigger bandwidth with new technologies, due to the applications we use and the requirements of the clients, we constantly face the problem of not having enough bandwidth to satisfy both the applications and the client requirements. Especially in the case of performance monitoring, we need to perform the test procedures continuously for a long period of time, and thus the consumed bandwidth is even bigger than other applications.

By using mobile agent technology for performance monitoring, we are able to decrease the bandwidth to a minimum. The reason is that we only require a small amount of bandwidth while the agent is being compiled and sent out to the client to perform the tasks. Once the agent is on the client side, there will be no connection between and agent server and the agent, and the server will be able to accept other requests without having to sacrifice the previous requests.

3.2.4 Robustness

A common goal for which designers of complex systems strive is robustness. Robustness is the ability of a system to continue to operate correctly across a wide range of operational conditions, and to fail gracefully outside of that range. One of the major characteristics of a mobile agent is its robustness. Agents can work on their own without having to connect to the agent server permanently. Therefore, after we apply mobile agent technology to performance monitoring, we are able to do a better job than with the conventional approach. The reason is that, under the conventional approach, there needs to be constant connection between the test server and the client, since testing the performance of the client involves the sending and receiving of data packets. Once the connection is lost or unstable, the test server will not be able to continue the test from where it has stopped, or even it can continue, the result may be not as accurate as desired.

With the mobile agent approach, the test can be carried out as usual without having a permanent connection, and therefore robustness will be achieved without much trouble.

3.3. Related research

Though mobile agent is quite a new technology, it has been utilized in several areas in computer science [2,10,13]. We will look at the related researches on mobile agent for network management.

The authors of the research note on mobile agents for network management [2] realized as early as 1998 that the mobile agent framework is an emerging standard, and they tried to utilize it in the network management area. Basically they categorized basic network management into three different areas, namely fault management, configuration management, and performance management. They wanted to use mobile agents in these areas, 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 recent new technologies are hoping to achieve.

Furthermore, in another paper [13], the authors realized that they could actually apply the mobile agent technology in another way. They utilized the mobile agent framework in network management back in 1998. Several problems were faced during the implementation of their JAMES project [17]. The major problems were that the application has to be centered on the mobile agents, and a complicated interface between the agents and the application has to be written in order to make the application work. With these problems, the end users will have difficulty in using the application because they not only need to configure security permissions, but also have to manage the troublesome agent platforms.

Hence, the authors developed another approach for utilizing mobile agent technology, which is called the application-centric mobile agent system (ACMAS). With this new framework, the mobile agents become the underlying mechanisms that perform messaging instead of all the necessary tasks. The end users now see only the application without worrying about the complicated configurations. Components are utilized to divide the tasks among the different agents. The advantage of the new framework is that the mobile agents now interact directly with the applications only from the inside, and thus no agent platforms are needed. In addition, users can self-configure the components they need to specialize the application to suit their needs.

4. Proposed mechanism for using mobile agent for web testing

4.1. General framework

An architecture was needed for agents working within the mass environment which effectively isolated the agent-dependent behavior logic from the underlying support code which would be common to all of the agents in the simulation. One goal of the framework was therefore to allow an agent's behavioral logic to perform without the knowledge that it was operating under simulated conditions, e.g. a problem solving component in a simulated agent would be the same as in a real agent of the same type. The framework also needed to be flexible and extensible, and yet maintain separation between mutually dependent functional areas to the extent that one could be replaced without modifying the other.

Before we look at the details of how to utilize mobile agent technology for web testing, we first discuss the framework that we can use to achieve this. Generally the framework can be divided into two different types.

4.1.1 Controllable environments

Controllable environments are those in which we can achieve complete control of the environment. Under this kind of environment, we will not have to worry about the authorities given to us, as we have the maximum authorities. Since we are able to control what we want to do and what is allowed to be performed in the environment, we can do whatever we want. Therefore, the mobile agent platform can be installed in the environment, and we can allow the incoming agent to perform the tasks that it was designed to do.

In this case, we do not need complicated setup of the environment. We can simply install the platform on any computer in this environment, and the mobile agent will be able to move from one computer to another to perform tasks that are allocated to it. The simplified idea is illustrated as Path 1 in Fig. 4-1.

4.1.2. Uncontrollable environments

Uncontrollable environments are those that do not allow us to execute processes without being permitted. This is the category to which the majority of the environments belong, as people do not wish to have hostile entries that possess the potential to harm the system.

When we want to use mobile agent technology to test this kind of environment, we have no choice but to use another test server to test the environment instead of testing the targeted environment physically. The test server can be any computer. It can be a computer situated in the agent server side, or a computer located within the network area of the targeted web server. Path 2 in Fig. 4-1 shows the conceptual layout.

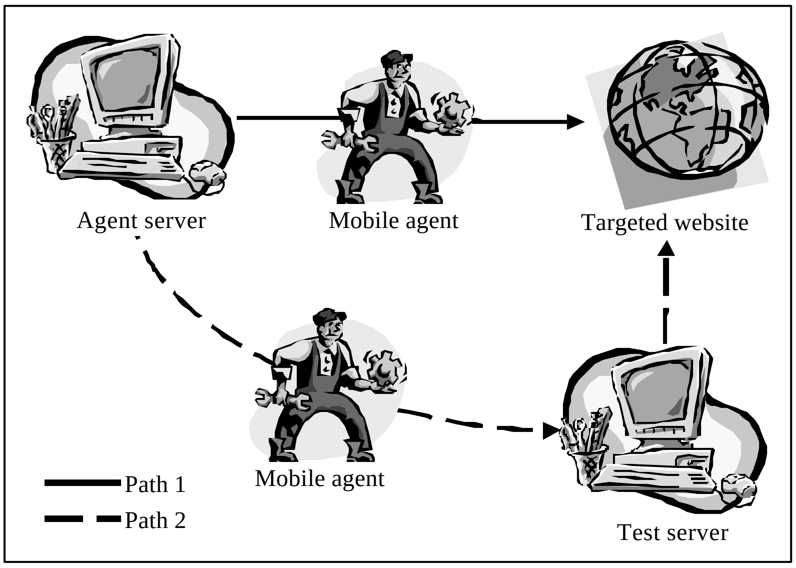


Fig. 4-1. The conceptual evaluation paths

The general framework is somewhat similar for both the controllable and the uncontrollable environments. Fig. 4-2 demonstrates the framework diagrammatically, and the following descriptions tell us how the framework works.

1. Initialization

The target website initiates the test by accessing the web page of the agent server. It will need to decide which test that it wants to perform. Once the test category is chosen, the agent server will compile and create the agent that is designed to perform the certain test.

The agent server will also assign values to the agent so that the agent will perform the test automatically if required.

2. Dispatch

After the agent has been created and assigned with the specification obtained earlier, it will be sent out to the cyber space. After the agent has been sent out, it is on its own. The agent will be responsible for performing the required tests automatically. During the test, the agent needs not to be connected to the server.

3. Perform tests

This is the part that varies as the test category varies. Basically the test categories can be divided into three parts, namely workload and performance testing, security checking, and reliability appraisal. Workload and performance testing can tell us the maximum amount of load and stress the web server can handle, and from the collected data we are able to determine the performance of it as well. We can ensure that the security func tions of the website are working as expected with security checking. In addition, we can estimate the website reliability with reliability appraisal, which determines the reliability from the hyperlink navigation structure. The detailed tasks performed by the agent will be discussed in the next chapter.

4. Report server status

After the agent has collected and possibly compiled the data, the agent will report the results and the tables in various forms to the client, so that the client can have choices of different types of displays. The agent can report to the server as well if needed. It will be disposed after a certain period of time so that it will not take up the CPU resources.

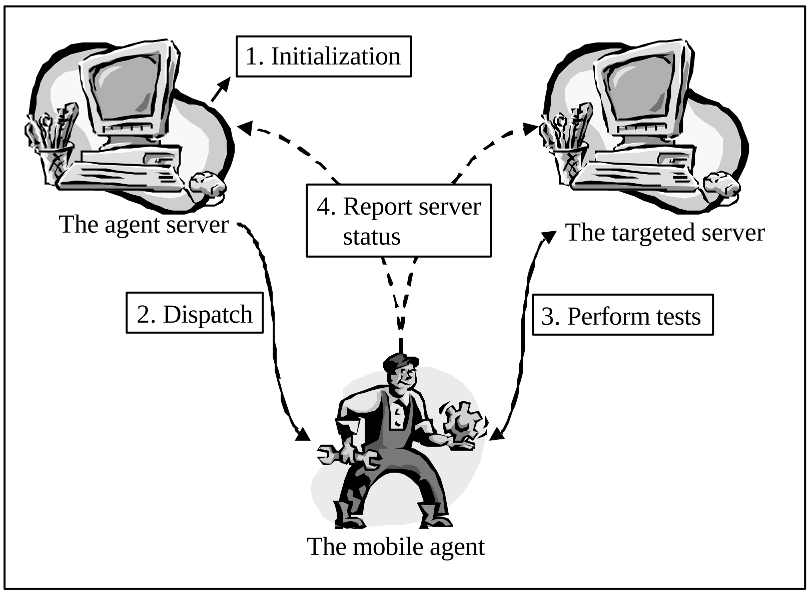


Fig. 4-2. The general framework

4.2. Benefits of utilizing the mobile agent framework

Mobile agent technology is used to solve several problems faced today, and the advantages of using mobile agent technology in our research can be categorized into two categories, namely bandwidth and robustness.

4.2.1. Bandwidth

First, we compare the bandwidth that the mobile agent approach and the conventional web-based approach utilize. The mobile agent is well known for its efficiency under limited bandwidth, and therefore we derive an equation to calculate the agent efficiency by comparing different bandwidth utilizations.

First, we make an assumption:

The complexity of the application is directly proportional to the execution time and the size of code. In other words, a more complicated application will have a longer execution time and a larger code.

To simplify the derivation, we use the following notation. Let

ε = the efficiency of the mobile agent, which is the degree of enhancement of bandwidth utilization by the mobile agent,

BW_{conv} = the bandwidth utilization of the conventional approach

BW_{agent} = the bandwidth utilization of the mobile agent approach

S_{trans} = the size of the conventional transaction, which mainly depends on the size of the web page, therefore we take it to be the size of the web page

S_{agent} = the size of the mobile agent

T_{ex} = the execution time for the conventional approach

T_{transC} = the transmission time for the conventional approach

T_{transA} = the transmission time for the mobile agent approach

Generally speaking, bandwidth is directly proportional to the amount of data transmitted or received per unit time. In a qualitative sense, bandwidth is proportional to the complexity of the data for a given level of system performance. For example, it takes more bandwidth to download a photograph in one second than it takes to download a page of text in one second. Large sound files, computer programs, and animated videos require still more bandwidth for acceptable system performance

By the general definition of bandwidth, we can say that the bandwidth utilized by an approach equals the size of the content multiplied by the time duration. Therefore, we can derive two equations representing the bandwidth utilized by the conventional web-based and the mobile agent approach.

The bandwidth utilized by conventional web-based testing (BW_{conv}) can be defined as follows:

$$BW_{conv} = C_1 \text{ x } S_{trans} \text{ x } (T_{ex} + T_{transC})$$

where C_1 denotes an arbitrary constant.

The bandwidth utilized by the mobile agent approach for testing (BW_{agent}) can be defined as follows:

$$BW_{agent} = C_2 \text{ x } S_{agent} \text{ x } T_{transA}$$

where C_2 denotes an arbitrary constant.

The efficiency of the mobile agent (ε) is the ratio of the bandwidth utilized by the conventional approach to that of the mobile agent approach. Hence, the relationship can be represented as:

$$\varepsilon = BW_{conv} / BW_{agent}$$

$$= C_1 \text{ x } S_{trans} \text{ x } (T_{ex} + T_{transC}) ／ C_2 \text{ x } S_{agent} \text{ x } T_{transA}$$

$$(S_{trans} ／ S_{agent}) \text{ x } 〔 (T_{ex} + T_{transC}) ／ T_{transA} 〕$$

assuming $C_1 ／ C_2 \quad 1$

Because transaction time and execution time are directly proportional to size under our assumption, therefore

$$\varepsilon = BW_{conv} / BW_{agent}$$

$$(S_{trans} ／ S_{agent}) \text{ x} (2S_{trans} ／ S_{agent})$$

Therefore, after we simplify the above equation, we have:

$$\varepsilon = 2(S_{trans} ／ S_{agent})^2 \qquad (1)$$

From equation 1, we can say that the agent efficiency equals two times the square of the ratio of the size of the web page to the size of the agent. Under most circumstances, the conventional web page is about 10 times larger than a mobile agent. Therefore,

å = $2\text{x}10^2 = 200$

From this result, we can say that the mobile agent is 200 times more efficient than the conventional web-based approach to testing. In other words, the mobile agent approach will only utilize 1/200 of the bandwidth required by the conventional approach to achieve the same results.

4.2.2. Robustness

Next, we investigate the robustness of the two different methods. A robust system can be taken to be the same as a system having a long mean period of time between each failure. In most cases, under the web environment, we will focus on the mean period of time between connection failures. A system with a longer mean period between connection failures has greater robustness.

To simplify the derivation of the equation, we use the some of the same variables used in mobile agent efficiency ε. In addition, we define the following:

Let γ = agent robustness, which is the degree of reduction of connection failures by the mobile agent

$$\gamma = (T_{ex} + T_{transC}) / T_{transA} \qquad (2)$$

The reason that we define γ as in equation 2 is that the mean period of time between connection failures of the conventional approach greatly depends on the execution time (T_{ex}) and the transmission time (T_{transC}), whereas that of the mobile agent approach mainly depends on the transmission time of the mobile agent (T_{transA}). The logic is that under normal circumstances, if a web page fails to load and execute within the total of execution time and transmission time, it will be considered as a failure. The same applies to the mobile agent approach. If the mobile agent fails to be transmitted within the transmission time, it is considered a failure.

After we simplify equation 2 with the same assumptions as those in equation 1, we have

$$\gamma = (S_{trans} + S_{trans}) / S_{agent}$$

$$= 2(S_{trans} / S_{agent}) \qquad (3)$$

From equation 3 we can evaluate the robustness of the mobile agent approach. Since as we reported earlier, a normal web page is about 10 times the size of a mobile agent, we can say that the robustness of the application utilizing the mobile agent approach is about 20 times that of the conventional web-based application.

4.3. Application limit

Mobile code denotes the general term of programs that are executed on foreign computers. Examples of mobile code are Java applets, Postscript files, etc. Here, mainly the executor needs guarantees that the execution of the mobile code will not harm its system, i.e. behave in another than the expected way. This part of the problem has two aspects: the mobile program is allowed to execute only permitted statements, and to use resources only in allowed quantities.

Mobile agents are special mobile code entities which have a state that moved with the agent, i.e. persists over time. This state consists of explicit data elements, and, sometimes, implicit execution state. It allows to distinguish between different instances of the same program.

When it comes to protect the executor from malicious programs, state has also to be taken into account as state is one of the factors determining the actions of a program. Nevertheless, the program code restricts the potential of a program if code is not self-modifying, as no state element (e.g. a variable value) can add a statement to a program.

The much bigger problem is that the important part of the state is provided by the owner of the agent. Therefore, the owner is often interested in getting that state protected from attacks. This means that there can be state elements that must not be read by unauthorized parties (e.g. digital cash), or that must not be modified (e.g. public keys). This also means that the owner needs a guarantee that the agent is executed only in the specified way (given the agent is error-free), as the agent acts in the name of the owner.

4.3.1. Agent security

The fact [12] that an execution environment may attack a program under execution hardly plays a role in classical computer security. This is because the party that maintains the execution environment generally also employs the program. This situation is different in the case of mobile code and agent-based systems. In this case, the mobile code owner and the operator of the execution environment are in most cases different parties. This situation automatically leads to the problem of malicious hosts and how to protect mobile code against the corresponding execution environments. In short, a malicious host can be defined in a general way as a party that is able to execute an agent that belongs to another party and that tries to attack that agent in some way. For example, the party may try to spy out or manipulate code, data, or control flow, masquerade another host or execution environment, or return wrong results to system calls issued by the agent.

It is commonly believed that protecting mobile code against malicious hosts is not a "nice-to-have" feature but is essential for agent-based systems to be useful. Against this background, it is often argued that mobile code cannot be effectively protected against the execution environment because the environment has full access to the mobile code and its data. This seems to imply that mobile code requires strong guarantees on the trustworthiness of the executing computer, thus mobile code technology would underline the necessity of a trusted computing base (TCB). So far, little research has been done on protecting mobile code against malicious hosts and corresponding execution environments. The resulting approaches can be classified into four groups:

The first group of approaches tries to circumvent the problem by not allowing mobile code to move to hosts that are not trusted. The main problem of these approaches is that it is not always clear in advance whether a specific host is trusted or not. In addition, trust may not always be a binary and absolute decision (it may happen that some specific information should be hidden from a trusted host).

The second class of approaches tries to circumvent the problem by taking organizational measures. In these approaches, the agent-based system is not open in the sense that everybody can open a host, but only trust-worthy parties can operate hosts.

The third class of approaches uses specialized, tamper-resistant hardware to ensure the integrity of the mobile code. Obviously, these approaches require the usage of special hardware in every host, which is currently a too restrictive requirement.

Finally, the fourth class of approaches tries to solve some of the problems related to mobile code protection by setting up restricted environments, deploying crypto-graphic protocols between the mobile code and a crypto-graphically "safe haven," and by frustrating potential adversaries by making mobile code tampering difficult.

Unfortunately, most of these approaches are insufficient because they are either too restrictive or too unreliable. According to, the main challenge for mobile code protection is to find answers to the following set of questions:

1. Can mobile code protect itself against tampering by a malicious host?

2. Can mobile code conceal the function it wants to have executed?

3. Can mobile code remotely sign a document without disclosing the user’ s private key?

Presently solutions to address the problem of protecting mobile code against attacks originating from their execution environment are aimed at prevention or detection.

4.3.2. Execution environment security

The most obvious concern [12] with regard to the security of mobile code and agent-based systems is related to the environment in which the code is assumed to execute. How can the environment be protected against potentially hostile actions of mobile code under execution? Our first examples of mobile code attacks, namely the christma.exe and Internet Worm incidents, have demonstrated the power of simple replication techniques to cause serious problems in computer networks and distributed systems. The problem and some partial solutions have been known for some time, but no perfect solution exists even when the objects being replicated are passive messages. For example, the original Telescript model of mobile code control involved giving each processes a limited supply of funds, and requiring a process to expend a certain quantity of those funds in order to accomplish specific results, such as spawning new copies of it). On the one hand, this puts some form of restriction on replication by out-of-control programs. On the other hand, it may also prevent desirable behavior, as when a process is requiring more funds that it is authorized to use.

In many respects mobile code resembles potentially malicious static code, such as provided by Trojan horses, computer viruses, and network worms. In theory, protection against malicious code has turned out to be hard. For example, in his 1983 ACM Turing Award Lecture, Ken Thompson showed that deciding whether an arbitrary code segment contains a Trojan horse is a hard problem. Obviously, this statement also holds for code that is mobile, and there is no general routine that can decide for every possible mobile code segment whether it contains malicious code. Consequently, the problem of protecting the runtime environment must be addressed alternatively.

5. Test methodologies in detail

5.1. Framework for testing workload and performance

5.1.1. Detailed steps

The desired testing framework proceeds by the following steps illustrated using flowchart and shown in Fig. 5-1:

1. Initialization

The target website initiates the test by accessing the web page of the agent server, with load/stress testing as the chosen option. The number of simulated simultaneous accesses can be set here, so the agent will know how many times it has to simulate to complete the test.

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 generating the required number of simultaneous accesses automatically. During the test, the agent needs not to be connected to the server.

3. Probe and collect data

The test is supposed to be incremental. The agent will be required to automatically generate increasing amount of accesses to simulate the number of users during a short period of time. It may start at 100 simultaneous accesses and increase at an interval of 50 or even 500. It will only terminate the test when the required amount of simultaneous accesses is reached. During each round of tests, the agent will be able to record necessary information for calculation at the later stage.

4. Compilation and tabulation

Once the agent has finished simulating the required number of simultaneous accesses, it will gather the information obtained earlier and compile the raw data, which is mainly the response time during each round of the tests. Then the agent is supposed to tabulate the results according to the number simulated accesses so as to show the client the load and stress that the server was facing during each round of the tests. The agent will report the results and the tables in various forms to the client, so that the client can have choices of different types of displays. The agent can report to the server as well if needed. It will be disposed after a certain period of time so that it will not take up the CPU resources.

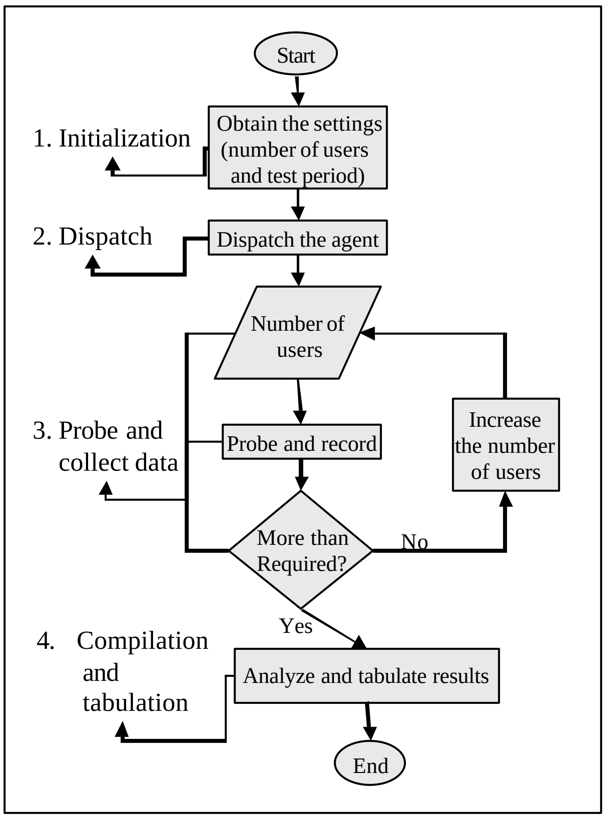

Start
1. Initialization
Obtain the settings (number of users and test period)
2. Dispatch
Dispatch the agent
Number of users
3. Probe and collect data
Probe and record
Increase the number of users
More than Required?
No
Yes
4. Compilation and tabulation
Analyze and tabulate results
End

Fig. 5-1. The desired testing steps

5.1.2. Test process in detail

Now we will look at the testing process in detail. The process is shown in Fig. 5-2. Before we can start the testing process, we have to assume that the mobile agent is already residing at the client side. Since the main objective of the agent is to simulate the number of simultaneous accesses to the website, the agent will be required to repeat the testing process and collect the necessary data.

From here, we can deduce that the agent will have a mechanism to generate numerous simultaneous accesses. This is where we place the information that we gathered while the client accessed the website. The client will determine before hand about the number of access that he wants to simulate and input the number while he accesses the website. The agent will use that number as the upper limit and start the testing process. The testing process will be incremental. It will start from the minimum and start to increase the number of simultaneous accesses by 50, 100, or even 500.

While the test is going on, we will need the agent to collect some information for us so that we are able to determine the load and stress of the server under different number of simulated simultaneous users, the threshold of the server, and finally the performance of the server. The kind of information we will be collecting is the response time. From the response time we will be able to verify whether the server is still able serve more users or the server is already at the peak of its performance.

After we have collected all the response time for different number of simultaneous access, the agent will be responsible for compiling and data the tabulate them into different forms, such as bar charts and pie charts. Then it will send the tabulated results back to the client and display the results to let the client know the load and stress that the server will be facing if given a certain number of simultaneous accesses. The agent will also be able to conclude from the data the performance of the server. In other words, the agent will be able to tell when the server will reach its peak. From the results the client can also learn the efficiency of the server and decide whether to control the number of users that can simultaneously connect to the site.

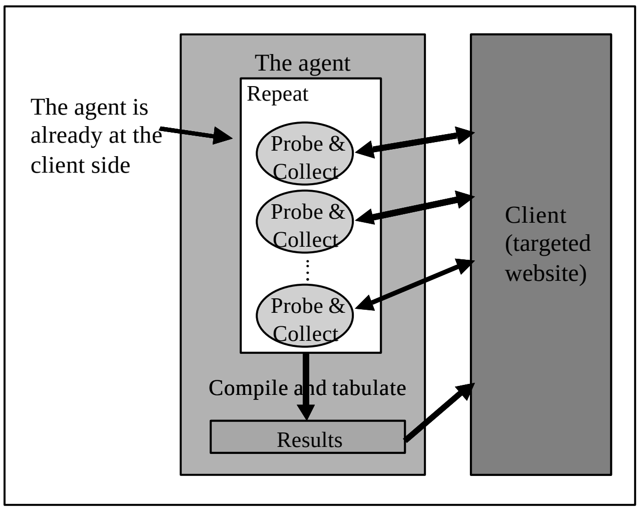


Fig. 5-2. The testing process in detail

5.2. Framework for security checking

5.2.1. Detailed steps

In this section, we describe in detail the steps for utilizing a mobile agent for security testing. Fig. 5-3 illustrates the steps clearly.

1. Initiate the test

The target website initiates the testing process by accessing the web page of the agent server. The agent can determine at this stage where it will go after being dispatched. Since the main purpose of the agent is to test the security scripts of the web server, other attributes that the agent requires will only be obtained just before the agent starts the testing process.

2. Dispatch

The agent server will then create an agent and send the agent out. After the agent has been sent out, it is on its own. The agent will be responsible for testing the security scripts of the targeted server and will report on failures automatically. During the process, the agent need not be connected to the server.

3. Test the scripts

The agent will perform some fairly simple tests to collect data. The agent will first obtain the test data from the system administrator of the web server. The test data includes two sets of user names and passwords. One is the valid set of user names and passwords whereas the other set is the invalid one. The agent will be responsible for applying these two sets of data for testing the validity of the security scripts.

4. Report to client if failure is spotted

The agent will continue testing the scripts alternatively with time intervals that can be set by the system administrator. The main purpose of the agent is to make sure that there is no security breakdown. If the valid set of user names and passwords does not allow the agent to log in successfully or the other way round, this is an indication that the security system is down. The agent will then record the time that the breakdown took place and after a period of time, it will send the log file to the system administrator. The system administrator will then know about the problem and fix it before the system is further damaged.

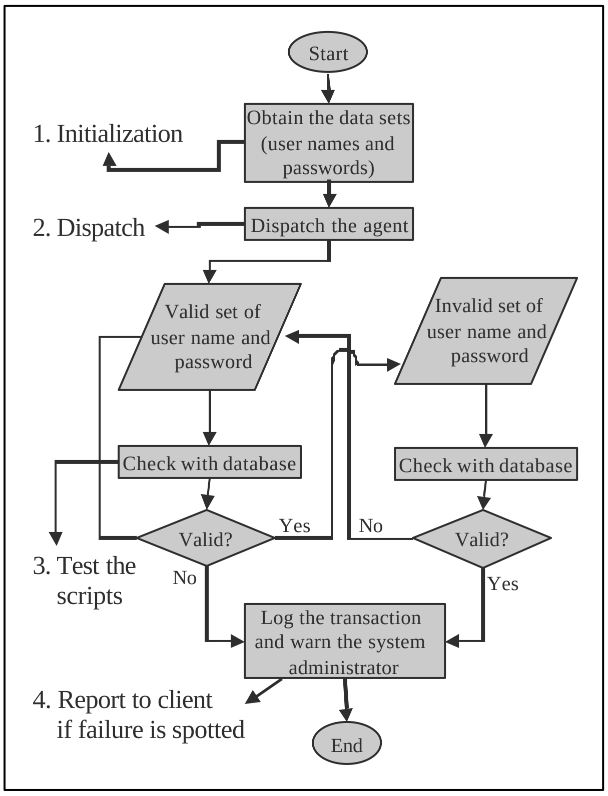

Start
Obtain the data sets (user names and passwords)
1. Initialization
Dispatch the agent
2. Dispatch
Valid set of user name and password
Invalid set of user name and password
Check with database
Check with database
Valid?
Yes
No
Valid?
3. Test the scripts
No
Yes
Log the transaction and warn the system administrator
4. Report to client if failure is spotted
End

Fig. 5-3. The desired checking steps

5.2.2. Checking procedures in detail

We will look at the checking process in detail now. Before the mobile agent carrying out the security checking on the website, it has to be assumed to be already residing at the client side. Since the main objective of the agent is to make sure that the security functionalities of the website is not malfunctioning, we can figure out that the agent will have a mechanism to automatically double-check the user name and password assigned to it with the set of user names and passwords in the database of the website. The valid set of user name and password will allow the agent to log into the server and access the contents, which is not the main function of this testing agent. On the other hand, the invalid set of user name and password will lead the agent to a page that will display the error. Fig. 5-4 shows the checking procedures.

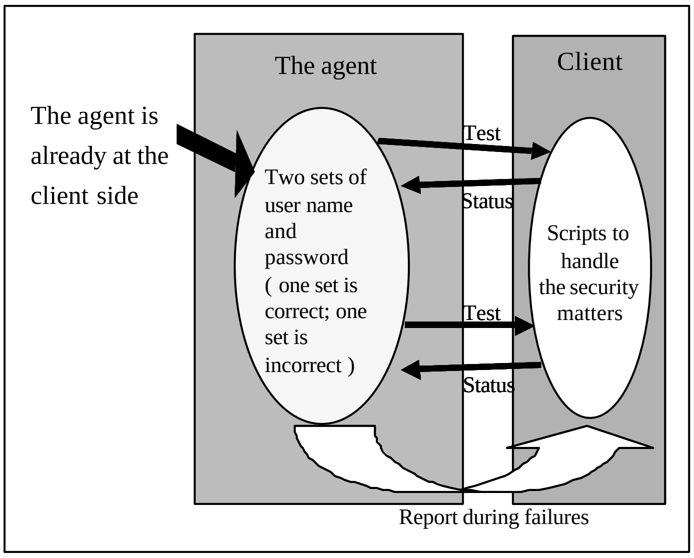

The agent
Client
The agent is already at the client side
Two sets of user name and password (one set is correct; one set is incorrect)
Test
Status
Scripts to handle the security matters
Test
Status
Report during failures

Fig. 5-4. The checking procedures

Under most circumstances, once security functions of any website malfunctions, the indication will be that either the valid set of user name and password cannot be used to log into the system or the invalid set can log in to the system. Once the stated situation occurs, the mobile agent record down the time of the occurrences. The agent will repeat the checking process as required and terminate the checking process according to the settings obtained in the earlier stages. If there are failures recorded in the log file, the mobile agent will send the log file via electronic mails or other means to the system administrator. Once the system administrator receives this log file, he will be able to address the problem as soon as possible, and further security leaks will be prevented.

5.3. Framework for reliability appraisal

5.3.1. Detailed steps

The steps for the framework for reliability appraisal are described in the following paragraphs and are illustrated in Fig. 5-5 using flowcharts.

1. Initialization

The target website initiates the appraisal process by accessing the web page of the agent server. Certain amount of information can be obtained here to aid the appraisal process, such as whether there are special items that the client wants the agent to test.

2. Dispatch

The server will then create an agent with the functional 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 performing the first stage of the appraisal process, which is performing simple tests and collect necessary data. During the process, the agent needs not to be connected to the server.

3. Test and collect data

The agent will perform some fairly simple tests to collect data. The agent will validate the hyperlinks of the website. Each path the agent takes will be considered as one testing chain. The agent will be performing what we call controlled validation. The agent will only validate the hyperlinks of the website to a certain level, in order to prevent the situation where the agent will test endlessly. The detailed mechanism for hyperlinks validation is exemplified in Fig. 5-7 in the next section. The testing chains will be packed together after the agent finished validating the website. The agent is not in charge of performing complete coverage for reliability appraisal as we are trying to keep the size of the agent small. If the size of the agent is too huge, then it will be pointless to utilize the mobile agent technology.

First is that it will still jam up the precious bandwidth that we are trying to save, and secondly, we might as well give out the complete version of our solution to the clients to save the trouble.

4. Data send back to server for processing

The collected data, that is, the testing chains, will then be stored in the agent and sent back to the server. As testing progresses and failures occur, the structure of the testing chain is amplified with "failure states." The website reliability, which may be calculated at any point in testing, is the probability of taking a random walk through the testing chain without encountering a failure state. In other words, reliability is the probability that every event in a complete usage scenario will be processed successfully. The same as the way Whittaker [21] defines software reliability, we define website reliability (R) of a website navigation structure as:

$$R = P_{first,last} + \sum_{j=1}^{n} P_{first,j} \times R_{j,last}$$

where $P_{first,last}$ is the probability of the path that will go straight from the first web page to the last web page, and

$\sum_{j=1}^{n} P_{first,j} \times R_{j,last}$ is the summation of the products of the probabilities of each path

Reliability is calculated from the testing chain with confidence intervals. On the server side, we calculate the website reliability with the aid from ToolSET_Certify by Q-Labs [14]. The compiled numbers will be the indicators for the website reliability.

5. Report

After the data has been compiled by the server, the agent will report the results to the client. The client can then determine from the numbers whether the site is reliable or not. Generally, when the number is closer to 1, the website is said to be more reliable than those that are further away from 1. Since the reliability factor greatly depends on the navigation structure of the website, it will be a good idea for the system administrator to focus on improving the validities of the hyperlinks to achieve higher reliability.

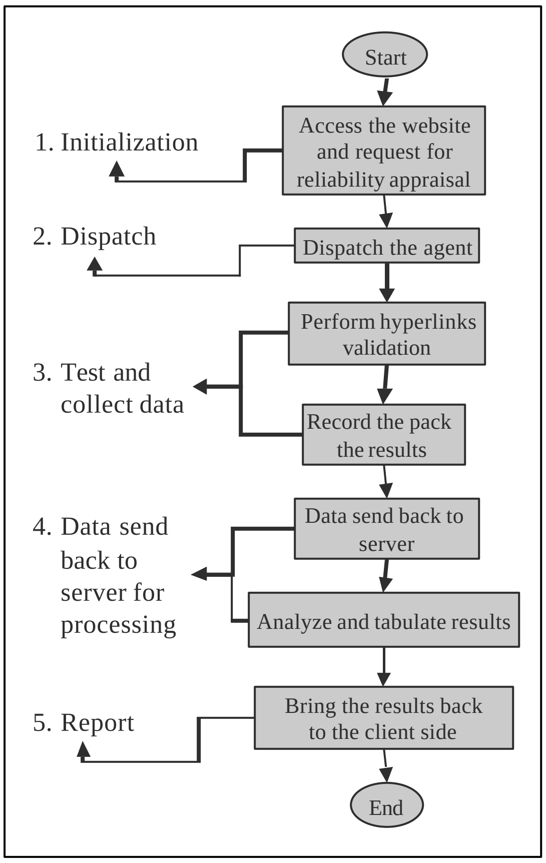

Start
1. Initialization
Access the website and request for reliability appraisal
2. Dispatch
Dispatch the agent
Perform hyperlinks validation
3. Test and collect data
Record the pack the results
Data send back to server
4. Data send back to server for processing
Analyze and tabulate results
Bring the results back to the client side
5. Report
End

Fig. 5-5. Flowcharts portraying the appraisal steps

5.3.2. Appraisal mechanism in detail

To describe the appraisal mechanism, we will have to assume that the mobile agent is already at the test server. Once the agent is at the test server, it will automatically start the appraisal process. The steps are described in the following paragraph and are illustrated in Fig. 5-6. It will first perform tests on two of the reliability issues discussed earlier, the validity and currency of the website. Checking whether there are dates and basic information about the author(s) on the pages does this.

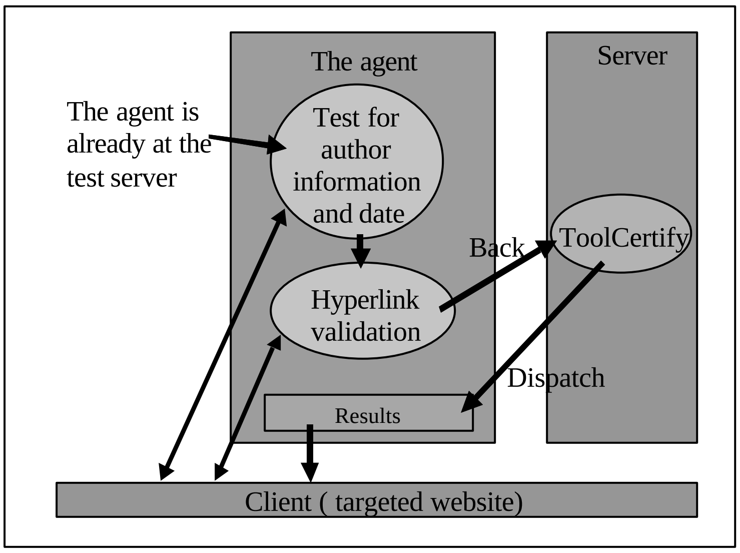

The agent
Server
The agent is already at the test server
Test for author information and date
ToolCertify
Back
Hyperlink validation
Dispatch
Results
Client (targeted website)

Fig. 5-6. Detailed appraisal processes

Next it will first test the status of the hyperlinks. Hyperlinks are the most important components in a website since they are the connections between different web pages. If the hyperlinks are dead, this will simply imply that the website is dead too. This is why we are validating the hyperlinks of the website as the first step. The agent will traverse the hyperlinks of the website to validate the hyperlinks. The agent will first visit the root of the navigation structure, which is the index page, of the website. The index page is considered as the node in level 1. The agent will then perform Depth First Search (DFS) [11] to obtain the rest of the navigation structure. The way that the mobile agent will validate the hyperlinks of the website is illustrated as flowchart in Fig. 5-7.

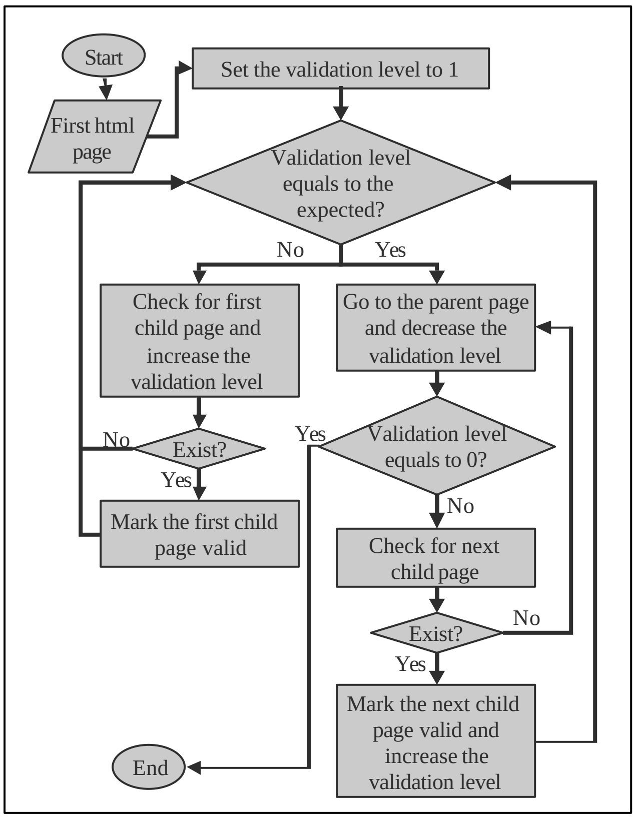

Start
Set the validation level to 1
First html page
Validation level equals to the expected?
No
Yes
Check for first child page and increase the validation level
Go to the parent page and decrease the validation level
No
Exist?
Yes
Yes
Validation level equals to 0?
No
Mark the first child page valid
Check for next child page
Exist?
No
Yes
Mark the next child page valid and increase the validation level
End

Fig. 5-7. Flowcharts describing the hyperlink validation process

Each path the agent takes will be considered as one testing chain which will be used later on for the calculation of the reliability of the website. After the agent has finished with the hyperlink validation, the testing chains will be packed together. As we discussed before, the structure of the testing chain is amplified with failure states as soon as testing progresses and failures occur. The reliability of the website is the probability of taking a random walk through the testing chain without encountering a failure state. In other words, reliability is the probability that every event in a complete usage scenario will be processed successfully and it is calculated from the testing chain with confidence intervals.

On the server side, we calculate the reliability of the website with the aid from ToolSET_Certify by Q-Labs [14]. ToolSET_Certify is a Markov [19,21,22] analysis tool. The tool is designed to support statistical testing in many aspects, such as developing Markov usage models, generating test scripts, performing statistical analysis of models, and providing the test results. In our case, the agent will first bring back the navigation structure obtained with hyperlink validation process. Then ToolSET_Certify will base on the navigation structure obtained by the hyperlink validation process and calculate for the reliability of the website. The compiled numbers will be the indicators for the reliability of the website.

After the results have been calculated, the agent will then bring back the results to the clients and display them in an orderly manner. The clients will then be able to determine whether their websites are reliable or not and decide on any improvements.

6. Demonstration example

6.1. Background

Without loss of generality, we take the MSN Hotmail website as an example for our framework. It is of the one of the most popular Internet email account providers. Hence, it will be an appropriate example to demonstrate the advantages of the mobile agent mechanism. Fig. 6-1 is a screen shot of it.

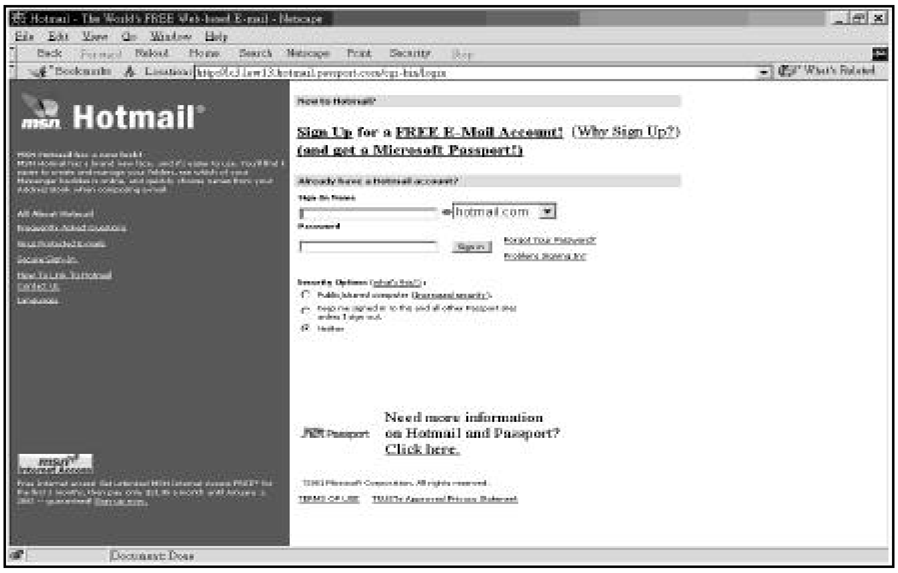

Hotmail - The World's FREE Web-based E-mail - Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://lc1.law13.hotmail.passport.com/cgi-bin/login
What's Related
msn Hotmail
New to Hotmail?
Sign Up for a FREE E-Mail Account! (Why Sign Up?)
(and get a Microsoft Passport!)
Already have a Hotmail account?
Sign-In Name
@hotmail.com
Password
Sign In
Forgot Your Password?
Problems Signing In?
Security Options
Neither
Passport
Need more information on Hotmail and Passport? Click here.
TERMS OF USE
Document: Done

Fig. 6-1. Snapshot of MSN Hotmail

6.2. Categorized demonstration

Since we used the IBM Aglets Software Development Kit (ASDK) as our tool to create our mobile agent, we will need to install the ASDK onto the server for the mobile agent to arrive there and perform the tests. However, at the present moment we are not able to install the agent server onto the Hotmail website, and therefore we decided to take an alternative approach. We installed ASDK on another computer within our control, and let it test the Hotmail website. When the test server makes a request to the agent server to test for the reliability of the website, the agent server will send an agent, with the testing abilities, to it. Fig. 6-2 shows the test path that we utilize.

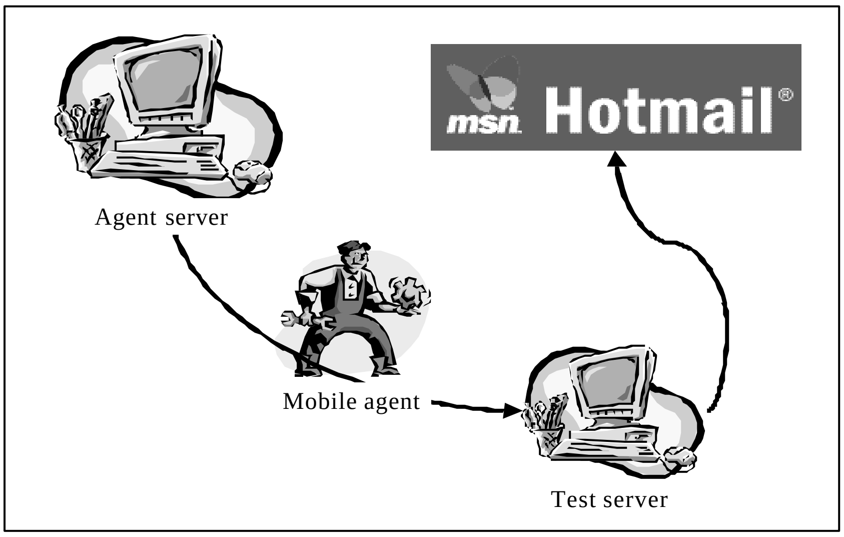


Fig. 6-2. Utilized test path

6.2.1. For workload and performance

1. Demonstration detail

With the MSN Hotmail as the subject, we will now show how exactly our framework will function. After ASDK has been set up and running at the test server, we send an agent out to carry out the test. Since one advantage and also the obvious purpose of using mobile agent for testing is the network bandwidth, we purposefully record the network status of the agent server. We notice that there is only a short period of connection while the agent is being compiled and sent out to the test server, which ideally should be the targeted website. Once the agent is at the client side (the test server), we notice no network connection between the agent server and the client at all. After a while, the agent returns with the data it has collected after testing. The results can be summarized into Fig. 6-3.

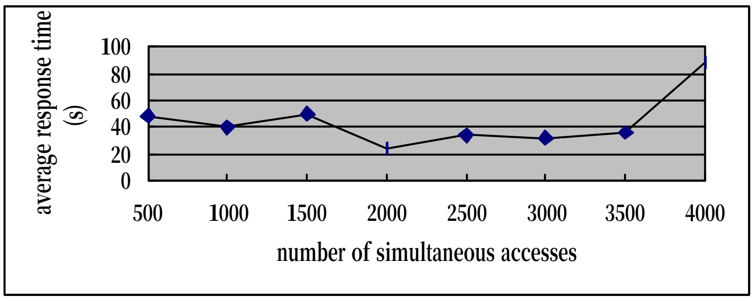


Fig. 6-3. The profile of the results

Fig. 6-3 is the kind of diagrams that we expect the agent to display to the client. It shows the load of the server during different period of the test. Also, from Fig. 6-3 we are able to see that the bottleneck of the web server is around 3500 simultaneous accesses as the average response time increases right up to around 90 at 4000 simultaneous accesses. We are able to determine from the figure that the peak performance of the web server is probably at 2000 simultaneous accesses as the average response time is even lesser than that of 500 accesses. It will be more meaning for if we allow the agent to continue the performance testing for a longer period of time. With this, we are telling the mobile agent to carry out performance monitoring. After one day, the agent returns with the monitoring report of tour faculty website. The default number of users used to simulate the situation is 3500, and hence, according to Fig. 6-3, the upper limit will be around 35.

Fig. 6-4 shows the summary of the monitoring process by the mobile agent approach. From the results the system administrator can determine when the web server is performing at its peak and when it is performing at its poorest.

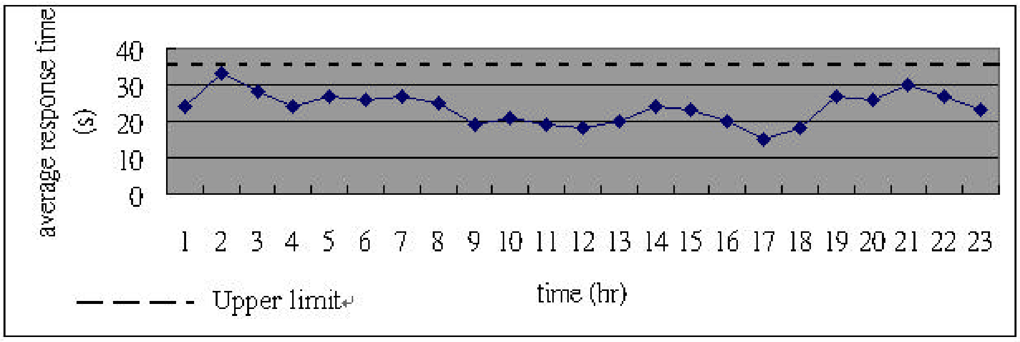


Fig. 6-4. The performance monitoring profile collected by the mobile agent approach

2. Comparison with other tools

We will now compare the differences between the mobile agent approach for performance monitoring and the monitoring tool provided by Web Performance Monitoring [20]. As we access the website of Web Performance Monitoring and requests for a monitoring for our faculty website for 1 day, with a period of 10 minutes, we recorded a total of 14,061,845 packets. On the other hand, when we switched to the mobile agent approach, we only recorded a total of 68,382 packets. From the data here, we can see that the tested agent efficiency å is indeed about 200 as 14,061,845 divided by 68,382 equals to 205.6 to the nearest decimal.

At the same time, on the website of Web Performance Monitoring, we saw a number of example output. We selected the sample output of 7 different websites, namely Amazon, CNET, Geocities, Lycos, Microsoft, Sun, and Yahoo. With the 7 websites, we recorded a total number of 167 failures. Hence, the average number of failures in 1 day equals to 23.857. The mean period of time between each connection failure of the conventional approach is 1.006 hours (24 / 23.857). This result indicates that when we want to monitor any website with Web Performance Monitoring, we will have to expect an average failure rate of 1 hour in 1 day.

As reported previously, we learn that the size of a conventional web page is generally 10 times larger than the size of a mobile agent. Therefore,

$\gamma = 2 \times 10 = 20$

With this, we can calculate for the mean period of time between connection failures for the mobile agent, which is 20.12 hours (1.006 x 20). With this result, we can say that the mobile agent approach can survive continuous performance monitoring for 20.12 hours before it may encounter with a failure. Table 6-1 summarized the comparisons.

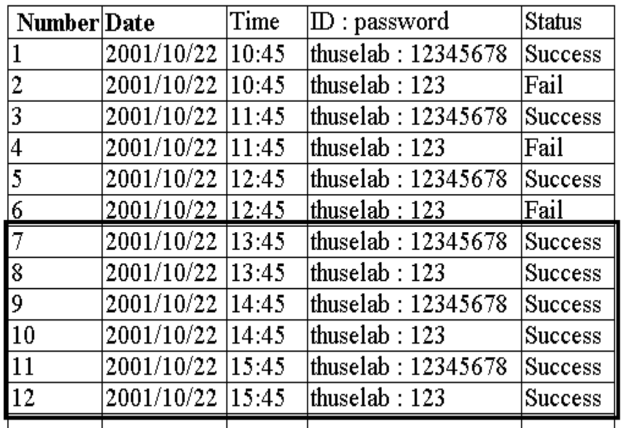
Table 6-1. Comparisons of the two performance monitoring approaches

Methods / Metrics	Web Performance Monitoring	Proposed approach
Total number of packets received in 1 day	14,061,845	68,382
Mobile agent efficiency (ε)	1	205.6
Robustness (γ)	1	20
Mean time between connection failures (hr)	1.006	20.12

6.2.2. For security

1. Demonstration detail

The agent will repeat the testing process as required. In this example, we have set the agent to perform the test every hour and report the results everyday. The sample log file is illustrated in Fig. 6-5.



Number	Date	Time	ID : password	Status
1	2001/10/22	10:45	thuselab : 12345678	Success
2	2001/10/22	10:45	thuselab : 123	Fail
3	2001/10/22	11:45	thuselab : 12345678	Success
4	2001/10/22	11:45	thuselab : 123	Fail
5	2001/10/22	12:45	thuselab : 12345678	Success
6	2001/10/22	12:45	thuselab : 123	Fail
7	2001/10/22	13:45	thuselab : 12345678	Success
8	2001/10/22	13:45	thuselab : 123	Success
9	2001/10/22	14:45	thuselab : 12345678	Success
10	2001/10/22	14:45	thuselab : 123	Success
11	2001/10/22	15:45	thuselab : 12345678	Success
12	2001/10/22	15:45	thuselab : 123	Success

Fig. 6-5. The sample log file

From the log file, we can see that the failure cases start at case number 8. By rights, when the supplied sign-in name (ID) is **thuselab**, the only password that will allow the agent to log in is **12345678**. However, in this case, when the password is **123**, the agent is still able to log into the system without prompting the error message. This will indicate that the security functionalities are not working as expected and therefore failure occurs.

Once the system administrator receives this log file, he will be able to address the problem as soon as possible, and further security leaks will be prevented.

6.2.3. For reliability

1. Demonstration detail

Since the agent is already at the test server, it will automatically start the appraisal process. It will first perform tests on two of the reliability issues discussed earlier, the validity and currency of the website. Checking whether there are dates and basic information about the authors on the pages does this.

Next it will test the status of the hyperlinks. The agent will traverse the hyperlinks of the website to validate the hyperlinks and create the navigation structure of the website. Each path the agent takes will be considered as one testing chain which will be used later on for the calculation of the reliability of the website. After the agent has finished the hyperlink validation with validation level set to 4, the navigation structure of the Hotmail website is shown in Fig. 6-6.

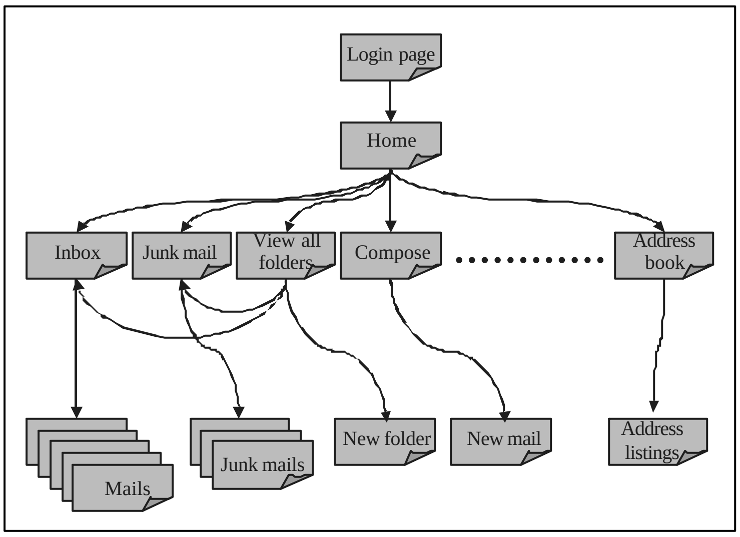


Fig. 6-6. The brief navigation structure of Hotmail website

After obtaining the navigation structure, the agent will pack the paths into the forms of the testing chains. Inevitability, errors such as connection failures may occur as the agent tries to achieve complete coverage of the hyperlinks validation process. Therefore, the structure of the testing chain is amplified with failure states as soon as testing progresses and failures occur. The reliability of the website is the probability of taking a random walk through the testing chain without encountering a failure state. In other words, reliability is the probability that every event in a complete usage scenario will be processed successfully and it is calculated from the testing chain with confidence intervals.

On the server side, we calculate the reliability of the website with the aid from ToolSET_Certify by Q-Labs [14]. ToolSET_Certify is a Markov [19,21,22] analysis tool. The tool is designed to support statistical testing in many aspects, such as developing Markov usage models, generating test scripts, performing statistical analysis of models, and providing the test results. In our case, the agent will first bring back the navigation structure obtained with hyperlink validation process. Then ToolSET_Certify will base on the navigation structure obtained by the hyperlink validation process and calculate for the reliability of the website. The compiled numbers will be the indicators for the reliability of the website.

After the results have been calculated, the agent will then bring back the results to the clients and display them in an orderly manner. The clients will then be able to determine whether their websites are reliable or not and decide on any improvements.

For our case, there are 32 nodes in all. The majority belongs to the mails in the Inbox. The number of nodes varies as the number of mails varies. With the aid of the ToolSET_Certify appraisal tool, we summarize the analyzed result as in Table 6-2.

In Table 6-2, the row labeled "Number of active hyperlinks" represents the numeric count of arcs in the navigation structure model, while the third row "Expected script length" indicates the expected number of hyperlinks in a typical test script. In other words, it provides the average script length among a large number of random scripts from the model. On the other hand, the fourth row "Least likely node-coverage expected at" shows the expected number of test scripts required to cover the minimal set of nodes. And the final row "Least likely hyperlink-coverage expected at" shows the expected number of test scripts required to cover the minimal set of hyperlinks.

Table 6-2. Analysis report of the MSN Hotmail navigation structure

Number of existing nodes	32
Number of active hyperlinks	192
Expected script length	10.2528
Least likely node-coverage expected at	156.4999
Least likely hyperlink-coverage expected at	235.5222

By the analysis result derived from the Markov usage model, partial certification results may be summarized as in Table 6-3, with 95% and 99% confidence intervals (C). In addition, R denotes the website reliability. Both the state coverage and arc coverage never reach 100% at script number 699. This indicates that at script number 699, there are still errors in the hyperlinks, and therefore we are not able to reach complete coverage. Table 6-3 also illustrates as the number of test scripts increased to 699, the test coverage for both states and transitions reached the limit. The website reliability increases as the number of non-failure test scripts increases.

Table 6-3. Appraisal result for the MSN Hotmail

Script #	Result	R	C=95%	C=99%	% State coverage	% Arc coverage
1	Pass	1	0	0	9.836065	4.385965
24	Pass	0.964286	0.012756	0.016765	72.13114	56.14035
128	Pass	0.967742	0.011541	0.015168	73.77049	57.89474
216	Fail	0.960124	0.004824	0.00634	98.36066	92.10526
357	Fail	0.953028	0.004325	0.005684	98.36066	92.10526
423	Fail	0.950128	0.004132	0.005431	98.36066	92.10526
501	Pass	0.962567	0.003452	0.004537	98.36066	92.10526
555	Pass	0.963303	0.003021	0.003971	98.36066	92.98246
617	Pass	0.967871	0.002647	0.003479	98.36066	92.98246
699	Pass	0.970326	0.002013	0.002646	98.36066	97.36842

Fig. 6-7 shows the reliability growth curve in 699 test scripts. From the diagram, we can see clearly that the reliability starts to drop dramatically at script number 216. The trend continues until script number 423 where the reliability is only around 0.95. Starting from script number 433, the reliability again increases, and finally, at script number 699, reaches around 0.97. From the diagram we can reckon that the failures occur between script number 216 and 423. In other words, the hyperlinks are not available during that period of time, which can be caused by many factors such as the server is down or the bandwidth is jammed up.

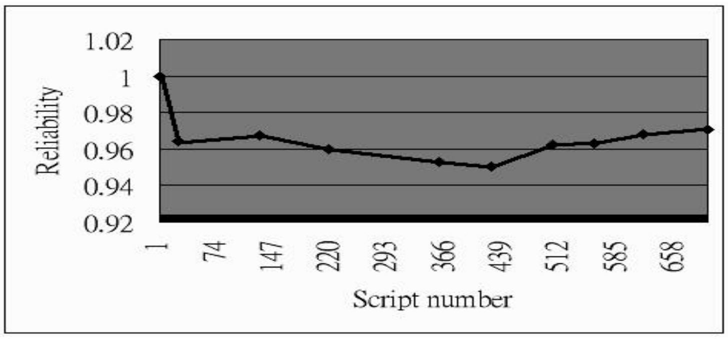


Fig. 6-7. The reliability growth curve of MSN Hotmail

7. Conclusion

Nowadays people are only interested making interactive and efficient websites. Most of them have neglected the quality of the websites. Quality websites attract people, whereas websites of low quality deters potential visitors. It is rare to see the utilization of mobile agent technology on web testing. However, as we study the characteristics of mobile agents, we realize that mobile agent technology is actually a potential candidate in this area. After detailed surveys on the issues related to attaining quality websites, we combine the characteristics of mobile agents to that of a quality environment to produce a framework that is capable carrying out web testing to attain quality using the mobile agent technology. Mobile agent possesses the abilities to reduce the network bandwidth greatly and achieve a higher level of robustness as compared to conventional methods, which is greatly appreciated under the web environment we have today. We have also made an assumption and derived two reasonable equations that produce satisfactory results.

We wish to further enrich our knowledge in mobile agent technology, and contribute our efforts in using mobile agents for web testing.

References

1. Agent Characteristics of Java, homepage: http://agent.cs.dartmouth.edu/workshop/1997/slides/lange/tsld002.htm, (last visited: 2001.8).

2. Andrzej Bieszczad, Bernard Pagurek, and Tony White, Carleton University, Mobile Agent for Network Management, 1998.

3. Wen-Kui Chang and Min-Hsiang Chuang, July 2001, Validating Hyperlinks by the Mobile-Agent Approach, Tunghai Science Vol. 3, p97-112.

4. Wen-Kui Chang and Min-Hsiang Chuang, November 23-24, 2001, Evaluating website reliability by the mobile-agent approach, 4th ROC Symposium on Reliability and Maintainability, Taipei, Taiwan, ROC, p193-202.

5. Wen-Kui Chang and Min-Hsiang Chuang, Investigation on Performance Testing for Website Quality, 37th Annual Conference on Chinese Society for Quality and 7th National Quality Management Symposium, Taipei, Taiwan, ROC, November 3, 2001, pp477-485.

6. Concordia-Java Mobile Agent Technology, homepage: http://www.concordiaagents.com/, (last visited: 2001.5).

7. Evaluate – Website Reliability Checklist, homepage: http://www.hcc.mass.edu/g_k/html/Evaluate.htm, (last visited: 2001.5).

8. IBM Aglets Software Development Kit Home, http://www.trl.ibm.com/aglets/index.html, (last visited: 2001.3).

9. Mark D'Inverno and Michael Luck, Understanding Agent Systems, Springer, 2001.

10. Martin L. Griss and Gilda Pour, Accelerating Development with Agent Components, Computer, May 2001, Volume 34, Number 5, pp37-43.

11. National Institute of Standards and Technology: Dictionary of Algorithms, Data structures, and Problems, homepage: http://hissa.nist.gov/dads/HTML, (last visited: 2001.2).

12. R. Oppliger, Security issues related to mobile code and agent-based systems, Computer Communications, 1999, Vol. 22, pp1165–1170.

13. Paulo Marques, Paulo Simões, Luí s Silva, Fernando Boavida, João Silva, Providing Applications with Mobile Agent Technology, IEEE Openarch 2001.

14. Q-Labs: ToolSET_Certify User Guide, Version 4.0, 1999.

15. P.-A. Queloz and A. Villazon, "Composition of Distributed Services with Mobile Code", Autonomous Agents and Multi-Agent Systems Journal, Volume 4, pp311-337, December 2001.

16. Murch, Richard, and Johnson, Tony, Intelligent Software Agents, Prentice Hall, 1998.

17. L. Silva, P. Simoes, G. Soares, P. Martins, V. Batista, C. Renato, L. Almeida, N. Stohr, "JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks", in Proceedings of IATA' 99, Stockholm, 1999.

18. Stefan P. Jaskiel and Steven Splaine, The Web Testing Handbook, Software Quality Engineering, 2001.

19. Tzupo Wang, master thesis for the Department of Computer Science & Information Engineering, Tunghai University, "A Study on Software Reliability Measurement of Distributed Systems", 2001.

20. Web Performance Monitoring, http://www.webperf.org/, last visited: 2001.11

21. J. A. Whittaker and J.H. Poore, "Markov analysis of software specifications." ACM Transactions on Software Engineering and Methodology, vol. 2(1), pp. 93-106, 1993.

22. J.A.Whittaker, K. Rekab, and M.G. Thomason, "A Markov chain model for predicting the reliability of multi-build software", Information and Software Technology vol. 42 pp. 889-894, 2000.