Consiglio Nazionale delle Ricerche

Remote Analysis of User Sessions for Usability Evaluation of Web Sites

Laila Paganelli, Fabio Paternò

Rapporto
CNUCE-B4-2001-013

CNUCE

Pisa

Remote Analysis of User Sessions for Usability Evaluation of Web Sites

Laila Paganelli, Fabio Paternò

CNUCE - C.N.R.
Pisa, Italy

Abstract
This paper describes how it is possible to automatically support usability evaluation of Web sites using user session logs and task models describing how the interface design assumes that activities should be performed in order to reach user goals. The approach is tool supported: there is a logging tool able to store events generated during user sessions at the client side and another tool, WebRemUSINE, that takes the logs and the task model and is able to provide evaluators with information regarding task performance and pages of the site. To this end, the tool exploits the hierarchical structure and the flexible set of temporal relationships that are specified in the task model. This approach supports remote evaluation where evaluators and users are separated in time and space.

Keywords: Model-based evaluation, Task Models, Remote evaluation, Automatic tools for usability evaluation.

Introduction

Creating a Web site allows millions of potential users with various goals and knowledge levels to access the information that it contains. For this reason, interest in usability evaluation of Web sites is rapidly increasing. In general terms, many methods for usability evaluation have been proposed: they range from inspection-based methods based on the ability of the evaluators (such as heuristic evaluation or cognitive walkthrough) to methods based on users involvement (such as usability testing or cooperative evaluation) or to others that use modelling techniques for example to predict time performance (such as GOMS-based methods).

There are many motivations for automatic tools able to support evaluation process. The total or partial automation of usability evaluation can reduce the time required and costs and release evaluators from repetitive and tedious tasks. A number of tools for usability evaluation of traditional graphical applications have been proposed, however, the different nature of Web interfaces requires specific tools. In this paper we present a method and a relative tool to detect usability problems in Web interfaces through a remote evaluation. Our approach combines two techniques that usually are applied separately: empirical testing and model-based evaluation. The reason for this integration is that models can be useful to detect usability problems but their use can be much more effective if they can be related to the actual use of a system. Our tool is able to analyse the possible inconsistency between the actual user interactions and the task model of the Web site that describes how its concrete design assumes that activities should be performed. To support remote evaluation, we have developed a technique that allows recording user actions during a site visit. The analysis of the logged data is based on the comparison of the traces of actions performed with the temporal constraints described in the task model. This analysis provides evaluators with a number of results that are related to the tasks that users intend to perform and the Web pages and their mutual relationships.

In the paper, we first discuss related works in the area of automatic support for usability evaluation of Web sites, next we describe the method supporting our approach and the underlying architecture.

Then, we move on to provide more detailed description of the logging tool that we have developed to collect data during user sessions, the preparation phase that is required to apply the method and the automatic evaluations that can be performed by our tool. During the description we discuss how the results provided can be helpful for evaluators to detect usability problems in Web sites.

Related Works

In recent years, interest in automatic support to usability evaluation of Web sites has been an increasing. The methods for usability evaluation of Web sites can be classified into two types of approaches: empirical evaluation, where users are directly involved to some extent, and analytical evaluation where various combinations of criteria, guidelines and models are applied to the evaluation of the site directly by the evaluators.
In the former group there are techniques based on the analysis of Web server logs whose effectiveness is strongly limited by the validity of the data (that cannot capture the accesses to the pages stored into the browser cache) and the impossibility to capture local user interaction with the user interface techniques (menus, buttons, fill in text, …). To overcome these limitations another approach, WebVip (Web Visual Instrumenter Program) [SLD98], has been developed at NIST. This tool allows logging of user interactions and the resulting log files can be analysed through VISVIP [CS99] a graphical tool that visualises the paths followed by the users during the site visit. The log files generated by WebVIP include user interactions (such as checkboxes, menu selections), interactions with the browser windows and events relative to loading the web pages. The logging tool proposed requires a number of modifications in the HTML pages that must be evaluated because each tag representing a user interface component calls for adding Javascript code to record the interaction. Because of the many modifications required, WebVip needs a copy of the entire site. Unfortunately copying the entire site can generate many errors: if relative paths have not been used then some paths can be no longer valid; in case of interactive sites some functionality of the site can be lost (such as CGI scripts or other structures that were in the original Web server), inserting code for recording logs may interfere with the normal behaviour of the user interface components that may already have specific event handlers created by their designers. Also WET [EC99] considers client-side logs but they are obtained more efficiently without requiring copy of the entire site. In this case it is sufficient to include the javascript file in the heading of the page. This javascript file includes the specification of the events that can be detected and the handling functions able to capture them. In WET only the click, change, mouseover and page load events are recorded. This limitation is due also to the lack of automatic tools able to analyse the data. Since the analysis is performed manually it is important to have readable log files with a content useful for the evaluator. Adding all possible events in the log file increases the complexity of its content since one user interaction can correspond to many events. For example, sending a form implies the generation of a sequence of various events (mouseover, mousedown, mouseup, click, submit). Other techniques are based on the use of questionnaires but they have limited ability to find detailed usability problems.
In the latter group we have tools such as Bobby [BURL] that aims to support verification of application of accessibility guidelines. WebSat [SLD98] provides a usability evaluation by analysing the HTML code through six categories of usability guidelines (accessibility, form use, performance, maintainability, navigation, readability). Design Advisor [F00] is based on guidelines derived from the use of eye-tracking techniques that identify which interface elements attract user attention (animations, images, colours, …) by identifying the scanning path on the Web page. In Web Criteria [WebCriteria99] an expert user model is applied. Instead of real users, the simulated user always follows an ideal path where no errors are performed and the shortest path is always selected.
In our case we follow a hybrid approach because our environment is able to analyse data relative to user interactions and then compare them to the task model corresponding to the design of the Web site. To this end, the first issue we addressed was what type of log files to consider. To detect user

actions we have used a technique similar to that used in Wet. Our tool exploits the possibility of defining handlers of browser events in order to record user interactions. The differences between Wet and our logging tool are two: Wet records only some event types because the analysis of the log files is manual whereas in our case the analysis is automatic and it is possible to record a wider set of events; in addition, to save log files WET uses cookies but with this technique it is possible to save only a limited amount of information (about 4K per cookie, with a maximum of 20 cookies per site) whereas our method uses a Java applet that is able to save log files in the server without limiting the amount of data, thus even with long user session there is no loss of data.
Differently from NetRaker [NRURL], during the session users are free to navigate as they wish: our method requires only that they clearly express what their current goal is from the list of supported tasks. The analysis of Web interfaces based only on heuristics can be useful to remove usability problems but the lack of user involvement can be a strong limitation and other problems can not be detected. In any event, our environment has already been integrated with functionality that perform a static analysis of the Web page. Our approach is strongly different from WebCriteria that considers the performance of an expert user who selects the shortest path from the homepage to the target page whereas in our case we consider the actual navigation path followed by users whatever level of experience they have.

The Method

Our approach combines two types of evaluation techniques that usually are applied separately: empirical testing and model-based evaluation. In empirical testing the actual user behaviour is analysed during a work session. This type of evaluation requires the evaluator to observe and record user actions in order to perform usability evaluation. Manual recording of user interactions requires a lot of effort thus automatic tools have been considered for this purpose. Some tools support video registration but also video analysis requires time and effort (usually it takes five times the duration of the session recorded) and some aspects in the user interaction can still be missed by the evaluator. In model-based evaluation, evaluators apply user or task models to predict interaction performance and identify possible critical aspects. For example GOMS (Goals, Operators, Methods and selection rules) has been used to describe an ideal error-free behaviour. Model-based approaches have proven to be useful but the lack of consideration for actual user behaviour can generate results that can be contradicted by the real user behaviour. It becomes important to identify a method that allows evaluators to apply models in evaluation still considering information empirically derived. To this end the main goals of our work are:

- To support remote usability evaluation where users and evaluators are separated in time and/or space [HCK96];
- To analyse possible mismatches between actual user behaviour and the design of the Web site represented by its task model in order to identify user errors and possible usability problems;
- To provide a set of quantitative measures (such as execution task time or page downloading time), regarding also group of users, useful for highlighting some usability problems.

The starting point was RemUSINE [PB00], an automatic tool based on the use of task models to support evaluations of graphical applications. This tool was not suitable for web applications whose specific aims are to support tasks related to retrieving and accessing information, and navigation is based on links to remote pages. In RemUSINE to identify errors (useless actions for the current task), the possible enabling and disabling of user interface actions was considered. Then, if users try to perform an action, this means that they want to perform the associated task, and if the action is disabled, then an error is performed. For example, suppose the user has to perform some actions and then save the data. If the user tries to save the data before terminating the sequence of actions planned, then this action would be disabled, and the error can be automatically detected. During Web site evaluation it is not possible to apply this concept because usually links are always enabled. Thus, in this context it is difficult to automatically identify user intentions. The solution that we have adopted to capture this information is to display the high-levels tasks that are supported by the Web site asking the user to indicate explicitly what task they want to perform. During the testing, since we perform remote evaluation without direct observation of the user interactions, it is important to obtain logs with detailed information. We have designed and implemented a logging tool able to record a set of actions wider than those contained in server logs. WebRemUSINE compares the logs with the task model and provides results regarding both the tasks and the Web pages supporting an analysis from both viewpoints.

The method is composed of three phases:

- *Preparation*, it consists in creating the task model of the Web site, collecting the logged data and defining the association between logged actions and basic tasks;
- *Automatic analysis*, where WebRemUSINEs examines the logged data with the support of the task model and provides a number of results concerning the performed tasks, errors, loading time, …
- *Evaluation*, the information generated is analysed by the evaluators to identify usability problems and possible improvements in the interface design.

Figure 1 shows the architecture of our system where rectangles represent transformation modules and ovals the input and output of such modules. The environment is mainly composed of three modules: the ConcurTaskTrees editor (publicly available at http://giove.cnuce.cnr.it/ctte.html) developed in our group; the logging tool that has been implemented by a combination of Javascript and applet Java to record user interactions; WebRemUSINE, a java tool able to perform an analysis of the files generated by the logging tool using the task model created with the CTTE tool.

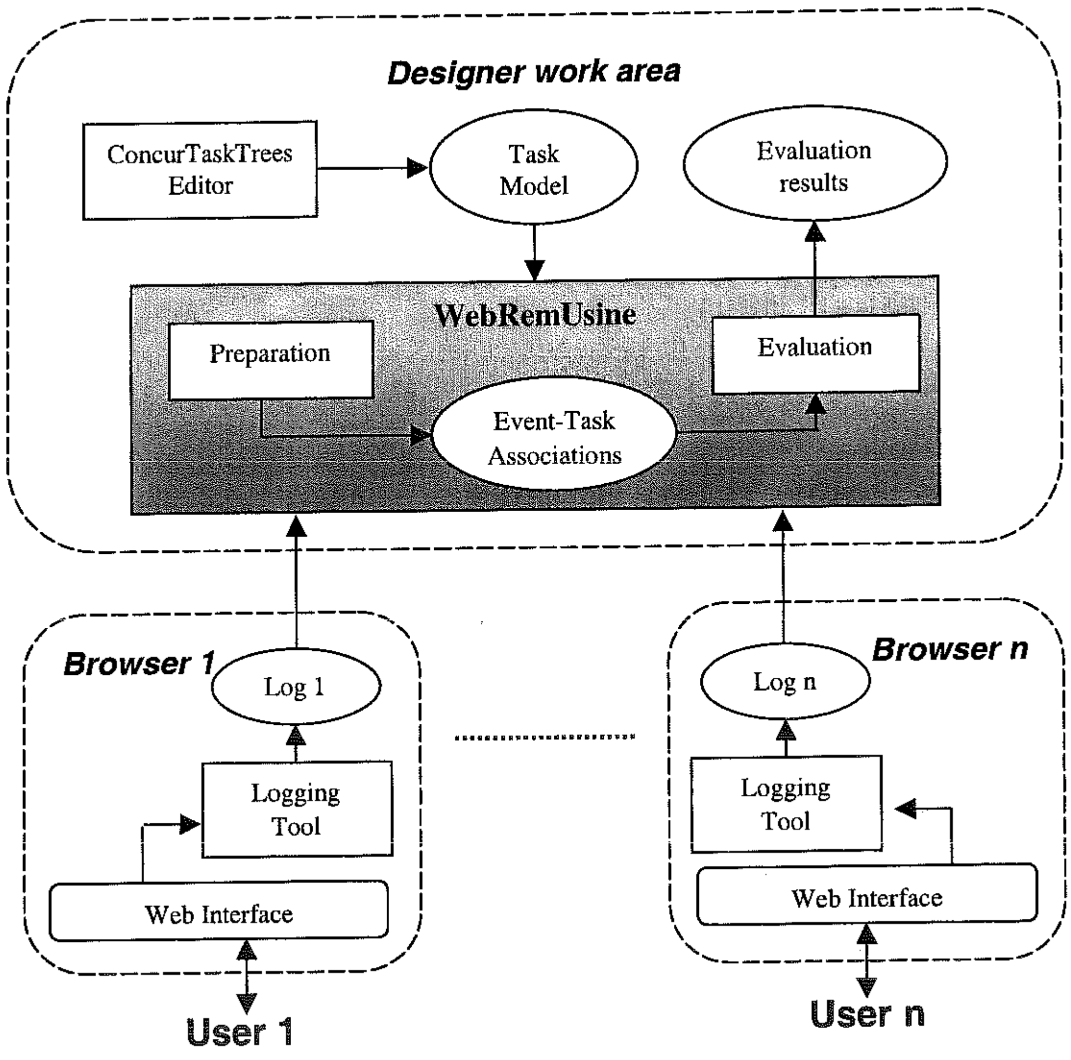


Figure 1: the architecture of our environment for remote evaluation.

Task models describe the activities to perform in order to reach user's goals. We have used the ConcurTaskTrees (CTT) [P99] notation to specify them. This is a notation where it is possible to graphically represent the hierarchical logical structure of the task model. It is possible to specify a number of flexible temporal relationships among such tasks (concurrency, enabling, disabling, suspend-resume, order-independence, optionality, ...) and for each task it is possible to indicate the objects that it manipulates and a number of attributes. The notation also allows designers to indicate how the performance of the task should be allocated (to the user, to the system, to their interaction) through different icons.

The logging tool is able to store various events detected by a browser. The Javascripts are encapsulated in the HTML pages and are executed by the browser. When the browser detects an event, it notifies the script for handling it. By exploiting this communication, the script can capture the events detected by the browser and add a temporal indication. Our tool works for the two main Web browsers (Micorosft IE and Netscape Communicator). Then, a Java applet stores the log files directly in the application server.

WebRemUSINE performs an automatic evaluation of a Web site providing the evaluator with a set of measures, concerning also group of users, useful to identify usability problems. The input for the tool are the task model and the log files recorded during the test sessions. As Figure 1 shows WebRemUSINE is composed of two submodules:

- *The preparation module*, this module filters the information recorded during the testing, then the evaluator has to associate each basic task with the corresponding event. All the event-basic task associations are recorded in a file.
- *The evaluation module*, it has three inputs: the task model, the log files and the event-basic tasks associations. This information is useful to analyse the logs with the support of the task model and identify errors performed by the user during the navigation. By following the sequence of events stored in the log it is possible to identify the corresponding tasks (through the event-basic tasks association) and comparing the sequence with the temporal relationships among the tasks it is possible to identify the tasks performed correctly and those that generate errors. It is also possible to calculate the completion time for the relative tasks. All results are displayed by WebRemUSINE in various formats both textual and graphical.

The WebRemUSINE analysis can point out usability problems such as tasks with long performance or tasks not performed according the task model corresponding to the Web site design. These elements are useful to identify the pages that create problems to the user. As previously explained, log files store both user interactions (mouse movements, keyboard input, link selection) and browser behaviour (start and end of page loading). The events corresponding to user interactions are associated with interaction tasks whereas the internal browser events are associated with system tasks. Thus the evaluation performed provides information concerning both tasks and Web pages. These results allow the evaluator to analyse the usability of the Web site from both viewpoints, for example comparing the time to perform a task with that for loading the pages involved in such a performance. WebRemUSINE also identifies the sequences of tasks performed and pages visited and is able to identify patterns of use, to evaluate if the user has performed the correct sequence of tasks according to the current goal and to count the useless actions performed. In addition, it is also able to indicate what tasks have been completed, those started but not completed and those never tried. This information is also useful for Web pages: never accessed web pages can indicate that either such pages are not interesting or that are difficult to reach. All these results can be provided for both a single user session and a group of sessions. The latter case is useful to understand if a certain problem occurs often or is limited to specific users in particular circumstances.

The Logging tool

Our logging tool is able to extend the browser behaviour by associating a script with the event handlers. Thus, it is possible to capture the user interactions with a Web site. All the pages should include this script. Unfortunately, these scripts are not persistent thus the visibility of the variables is limited to the page where they are defined whereas users can navigate across multiple pages within a web site. The collection of the data relative to multiple pages is performed through a Java applet that is activated at the beginning of the test and is active for all the session. Figure 2 shows the architecture of this module. Each page of the site includes the script for logging user interactions. All events are communicated to the applet that concatenates them. Lastly, at the end of the session the applet provides the server with all the logged events. For this purpose, in the server there is a servlet able to collect the data and save them into a file.

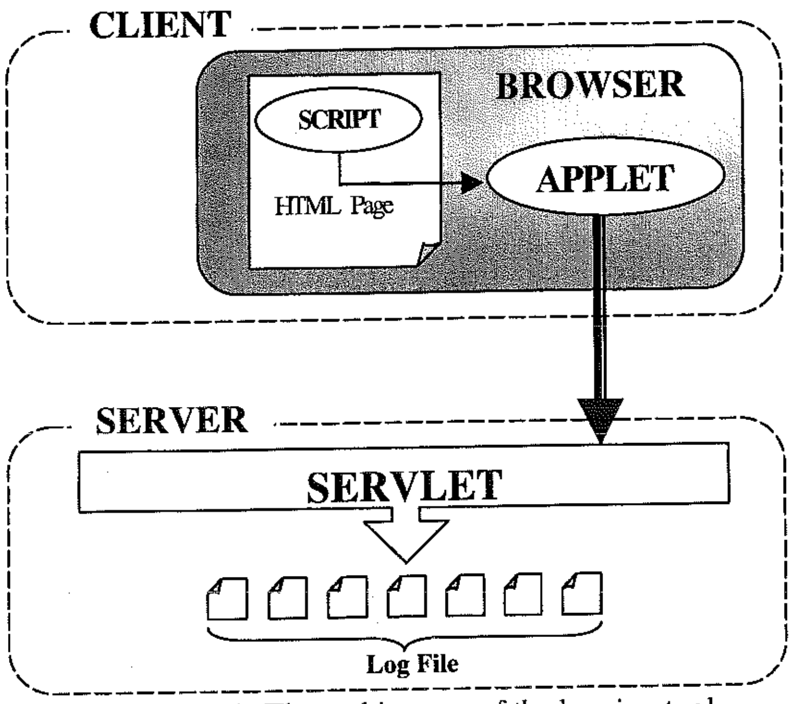


Figure 2: The architecture of the logging tool.

The script redefines event handlers in order to support recording of the following events: *abort* and *error* on images; *change* on form elements, *click* and *dblclick* on links, images and form elements, *load* and *unload* of pages, *submit* and *reset* of forms, *resize* and *scroll* of browser windows (see Figure 3). An event handler is a piece of code associated with an interaction object. When the user performs an interaction, such as button or link selection, the corresponding event handler is performed. For example, it is possible to implement an event handler that when the *mouseOver* event occurs on a link then shows a corresponding message at the bar level. To this purpose it is required to redefine the function predefined as event handlers. Our tool works with both Netscape and Internet Explorer even if they have different ways to analyse user-generated events. Figure 4 shows an excerpt from an example of log.

```
var isNetscape = (navigator.appName.indexOf("Netscape") != -1);
var isIE = (navigator.appName.indexOf("Microsoft") != -1);

if (isNetscape)
{
  document.captureEvents(Event.CLICK |Event.DBLCLIK);
  window.captureEvents(Event.LOAD|Event.UNLOAD|Event.RESIZE);
}

function time(){...}
function handler(e)
{
    . . . .

 //when the "load" event occurs then the defined event handlers
 //onChange,onAbort,onError,onSubmit and on Reset
 //are activated on the associated objects
  for(var i=0;i<document.forms.length;i++)
  {
     document.forms[i].onsubmit = handler;
     document.forms[i].onreset = handler;
     for(var j=0;j<document.forms[i].elements.length;j++)
       document.forms[i].elements[j].onchange = handler;
  }
  for(var i =0; i<document.images.length;i++)
     document.images[i].onabort=handler;

  . . . .
}

document.onclick = handler;
document.ondblclick = handler;
window.onload = handler;
window.onunload = handler;
window.onresize= handler;
if(isIE) window.onscroll = handler;
```

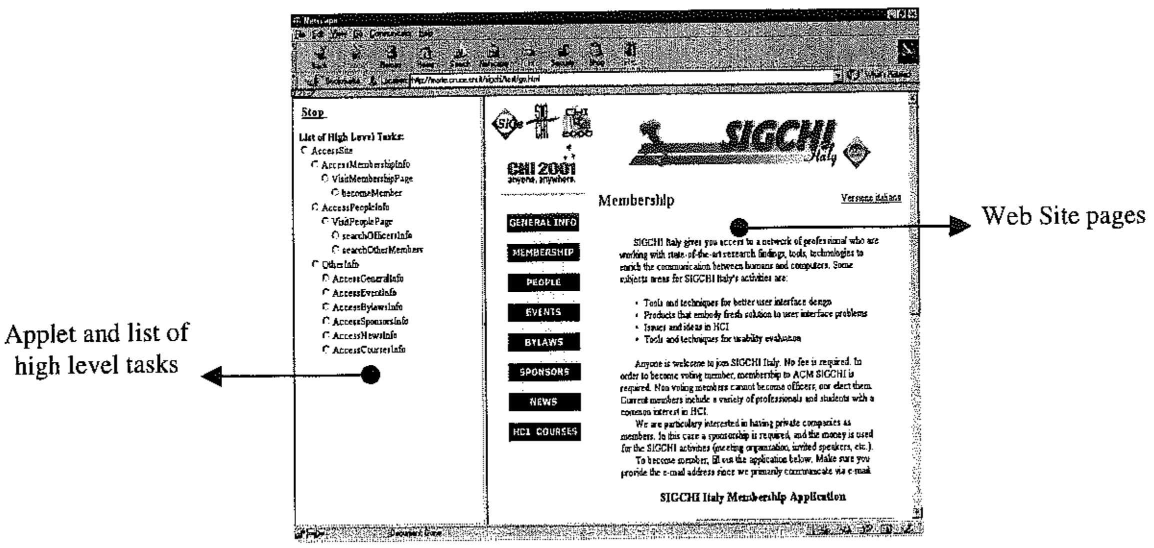
Figure 3: Structure of the code of the logging tool.

It is more difficult to understand user intentions from an automatic analysis when Web applications are considered. We aim to automatically determine if the user is able to reach the information desired and if he is able to follow the best navigational path. For example, if the user wants to download a program from a Web site he can access various pages of the site without finding the download page. Our goal is to highlight this problem, which reveals a usability problem since the user is not able to reach his goal. To this end, we have decided to provide the list of high-level tasks that are in the task model of the web site. This list represents the activities that can be performed during the site visit and the user has to select which one is the current goal. At any time, the user can change the current goal and select another task. To implement this, during the site test the browser window is divided into two frames: one to show the list of possible target tasks and the other showing the site pages. A radio button implements the possibility of selecting the target high-level task with labels indicating the task names (see Figure 5).

```
time:993813656762
LOAD http://marte.cnuce.cnr.it/sigchi/membership.htm
time:993813661649
CHANGE TARGET=textarea NAME=Address:
time:993813666096
CLICK TARGET=radio NAME=AddressType: VALUE=BUSINEss
time:993813668399
CLICK TARGET=submit NAME=B1 VALUE=Send
time:993813668479
SUBMIT TARGET=form
time:993813673757
CLICK TARGET=link NAME=home HREF=http://marte.cnuce.cnr.it/sigchi/index.htm
time:993813673887
UNLOAD
```

Figure 4: An excerpt of log.

The selection of one target high-level task activates a specific event handler that, as it happens with user-generated events, creates an element in the log file composed of a temporal indication.
Our solution has taken into account that when a Web page with a script is accessed then also the script is executed. However, the variables of the script are visible only from when the page is downloaded until a new page is loaded. This lack of data persistency during loading of multiple pages was the first issue to address. Cookies represent one possible solution to keep information regarding the session in a persistent manner, with the possibility of sharing then across multiple pages. Usually, cookies are a mechanism that can be used from server-side connections (such as CGI scripts) to store and retrieve information on the client side of the connection. Javascripts are able to access and save information in cookies thus making it accessible also from the server. This mechanism has some limitation on the amount of information that can be stored in the client system and for long sessions some data can get lost. We have chosen another solution to overcome the stateless of the Web [G98]. The page appearing in the browser is composed of two frames (see Figure 5). The first frame contains the applet while the second frame shows the web site pages containing the script.



Stop

List of High Level Tasks:
- AccessSite
  - AccessMembershipInfo
    - VisitMembershipPage
      - becomeMember
  - AccessPeopleInfo
    - VisitPeoplePage
      - searchOfficersInfo
      - searchOtherMembers
  - OtherInfo
    - AccessGeneralInfo
    - AccessEventInfo
    - AccessBylawsInfo
    - AccessSponsorsInfo
    - AccessNewsInfo
    - AccessCoursesInfo

Membership

SIGCHI Italy gives you access to a network of professional who are working with state-of-the-art research findings, tools, technologies to enrich the communication between humans and computers. Some subjects areas for SIGCHI Italy's activities are:

- Tools and techniques for better user interface design
- Products that embody fresh solution to user interface problems
- Issues and ideas in HCI
- Tools and techniques for usability evaluation

Anyone is welcome to join SIGCHI Italy. No fee is required. In order to become voting member, membership to ACM SIGCHI is required. Non voting members cannot become officers, nor elect them. Current members include a variety of professionals and students with a common interest in HCI.
We are particulary interested in having private companies as members. In this case a sponsorship is required, and the money is used for the SIGCHI activities (meeting organization, invited speakers, etc.).
To become member, fill out the application below. Make sure you provide the e-mail address since we primarily communicate via e-mail.

SIGCHI Italy Membership Application

Figure 5: Layout structure of Web application during testing.

While the user interacts with the site, the script captures all the events and communicates them to the applet. All the information is kept by the applet during the session and only at the end is saved into a file. The end of a session is explicitly indicated by the user selection of the stop button in the frame with the applet and the high-level tasks. Since applets can store information on the server, the log files are transmitted to the server, where the evaluators can access all of them. The communication between applet and server is performed through a servlet that stores the information received from the applet into a file. Servlets are able to access some information regarding the client system and, in some cases, the user. In our case the servlet inserts in each file a heading with the following information: date and time of log reception, IP address and host name of the client system. User name is provided directly from the user. The last operation of the servlet is to save the file in a predefined directory in the server system.

The Preparation Phase

The main goal of the preparation phase is to create an association between the basic tasks of the task model and the events that can be generated during a session with the Web site. This association allows the tool to use the semantic information contained in the task model to analyse the sequence of user interactions. In this phase, the frame with the list of high-level tasks supported is also created. This is performed automatically through a depth-first analysis of the task model and the generation of the corresponding HTML code supporting the possibility of selecting one of them. Once the association file is created, it can be used to analyse as many user sessions as desired without any additional effort.

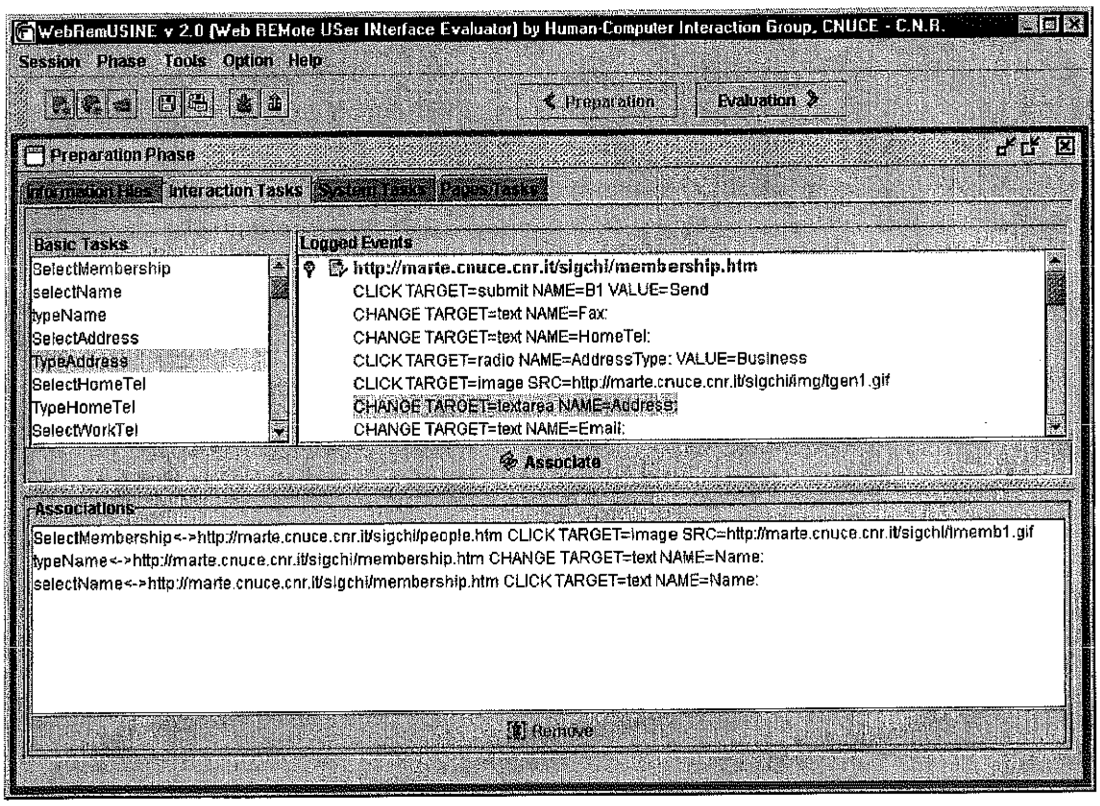
Basic tasks are tasks that cannot be further decomposed while in high-level tasks we have complex activities composed of sub-activities. The log files are composed of set of events. If an event is not associated with any basic task, it means that either the task model is not sufficiently detailed, or the action is erroneous because the application design does not call for its occurrence. For example, when a user sends a form then two events are stored in the log: one associated with the selection of the Submit button and the other one with the actual transmission of the form. Thus, in the task model two basic tasks are required one interaction task for the button selection and one system task for the form transmission otherwise it is uncompleted. Whereas if the user selects a non interactive image it means that an error has been performed which also points out a usability problem since it shows that the user does not understand that the image is static with no functionality associated.

In the logs there are three types of events: user-generated events (such as *click*, *change*), page-generated events (associated with loading and sending of pages and forms) and events associated with the change of the target task by the user.

Tasks can belong to three different categories according to the allocation of their performance: user tasks are only internal cognitive activities that thus cannot be captured in system logs, interaction tasks are associated with user interactions (*click*, *change*, ...) and system tasks are associated with the internal browser generated events. In addition, the high-level tasks in the model are those that can be selected as target tasks by the user. Each event is associated with a single task whereas a task can be performed through different events. For example, the movement from one field to another one within a form can be performed by mouse, arrow key or Tab key. The one-to-many association between tasks and events is also useful to simplify the task model when large Web sites are considered so that we need only one task in the model to represent the performance of the same task on multiple Web pages.

The main activity supported by the WebRemUSINE tool during the preparation phase is the creation of the association files for interaction and system tasks. The list of basic tasks and the events contained in the log files considered are loaded in two separate lists (see Figure 6). In the figure, the list with task names contains the names of all the basic interactive tasks while the list of events contains the list of events that appear in the log considered. If the user performs multiple times one event, that event appears only once in the list. Each event is composed of its description and the indication of the corresponding page. All the events associated with one page are grouped in the presentation. The association is created by selecting one element in each list and pressing the

Associate button. The events associated are removed from the list while tasks remain visible because they may be associated with other events. When a task is associated then it is shown by a different colour. All the associations performed can be visualised and edited for removing previously created associations. In this case, the removed event will be shown again in the list of events.



WebRemUSINE v 2.0 (Web REMote USer INterface Evaluator) by Human-Computer Interaction Group, CNUCE - C.N.R.
Session Phase Tools Option Help
Preparation
Evaluation
Preparation Phase
Information Files
Interaction Tasks
System Tasks
Pages/Tasks
Associate
Remove

Basic Tasks

```
SelectMembership
selectName
typeName
SelectAddress
TypeAddress
SelectHomeTel
TypeHomeTel
SelectWorkTel
```

Logged Events

```
http://marte.cnuce.cnr.it/sigchi/membership.htm
    CLICK TARGET=submit NAME=B1 VALUE=Send
    CHANGE TARGET=text NAME=Fax:
    CHANGE TARGET=text NAME=HomeTel:
    CLICK TARGET=radio NAME=AddressType: VALUE=Business
    CLICK TARGET=image SRC=http://marte.cnuce.cnr.it/sigchi/img/tgen1.gif
    CHANGE TARGET=textarea NAME=Address:
    CHANGE TARGET=text NAME=Email:
```

Associations

```
SelectMembership<->http://marte.cnuce.cnr.it/sigchi/people.htm CLICK TARGET=image SRC=http://marte.cnuce.cnr.it/sigchi/imemb1.gif
typeName<->http://marte.cnuce.cnr.it/sigchi/membership.htm CHANGE TARGET=text NAME=Name:
selectName<->http://marte.cnuce.cnr.it/sigchi/membership.htm CLICK TARGET=text NAME=Name:
```

Figure 6: Tool support for the preparation phase.

The Analysis and Evaluation Phase

Once the task-event association has been created then it is possible to move on to the evaluation phase. The evaluation provides a number of results regarding both tasks and pages allowing evaluators to perform an analysis from both viewpoints. For example, during the evaluation the tool calculates both the time to perform a task on a page and the time of visit of the same page. Thus, the evaluator can deem if it is the entire page to create problems to the user (for example, because it contains too much information) or if it is the task performance to require too long time. Through the log analysis WebRemUSINE identifies:

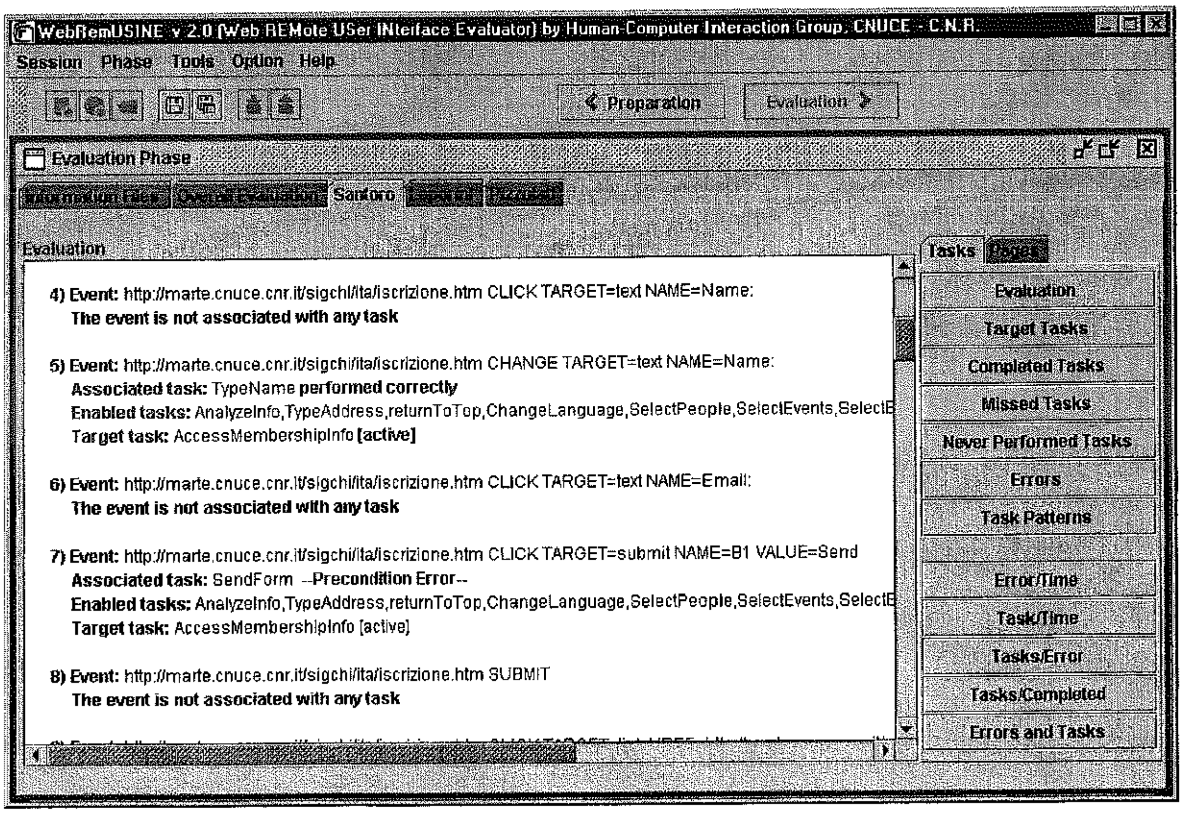
- The sequences of tasks performed and pages visited;
- The patterns existing in such sequences;
- If the user has performed the correct sequence of actions for reaching the current goal, if the goal has been achieved or if useless actions have been performed;
- Correctly performed tasks, those that have caused user errors and those never tried;
- The pages of the site never visited;
- Time taken to perform tasks;
- Time to download and visit pages.

The above results can be calculated both for single logs than for all the sessions available. In addition, WebRemUSINE is able to calculate summary information and statistics regarding the set

of sessions considered. This evaluation is performed by exploiting the basic tasks/events association that allows the tool to analyse user behaviour with the support of the task model.
As we explained beforehand, the logged events are associated with basic tasks and the target tasks with high-level tasks in the model. The analysis performed depends on the type of task:

- For basic tasks the tool checks that the temporal relationships defined in the task model are not violated; if a disabled basic task is performed then a precondition error is indicated otherwise the task is considered correctly performed;
- For high-level tasks the tool determines if all the corresponding basic tasks have been correctly performed (and thus the goal has been correctly achieved) and if some useless basic tasks has been performed.

During the analysis, the WebRemUSINE tool internally uses a simulator that was implemented for the CTTE tool. This simulator takes a task and is able to indicate what the next enabled tasks are according to the constraints indicated in the task model. Thus, at the beginning WebRemUSINE activates the simulator that returns the list of initially enabled tasks. Then, for each event in the log WebRemUSINE asks the simulator to perform the corresponding basic task and return the enabled tasks after its performance as well as the high-level tasks that have completed their performance.
WebRemUSINE also shows analysis of log files. In the readable list (see Figure 7), for each event three types of information can be given: the event is associated with a basic task and the performance was either correct (number 5 in the figure) or a precondition error occurred (number 7 in the figure), alternatively the event was not associated with any basic task (number 4 or 6 in the figure). In addition to the event description and the name of the corresponding basic task (if any), the tool also shows the basic tasks enabled after the performance of the basic task considered and the name of the current high-level target task. This information is useful for evaluators for an interactive analysis of the sequence of actions performed.



```
4) Event: http://marte.cnuce.cnr.it/sigchi/ita/iscrizione.htm CLICK TARGET=text NAME=Name:
   The event is not associated with any task

5) Event: http://marte.cnuce.cnr.it/sigchi/ita/iscrizione.htm CHANGE TARGET=text NAME=Name:
   Associated task: TypeName performed correctly
   Enabled tasks: AnalyzeInfo,TypeAddress,returnToTop,ChangeLanguage,SelectPeople,SelectEvents,SelectE
   Target task: AccessMembershipInfo [active]

6) Event: http://marte.cnuce.cnr.it/sigchi/ita/iscrizione.htm CLICK TARGET=text NAME=Email:
   The event is not associated with any task

7) Event: http://marte.cnuce.cnr.it/sigchi/ita/iscrizione.htm CLICK TARGET=submit NAME=B1 VALUE=Send
   Associated task: SendForm --Precondition Error--
   Enabled tasks: AnalyzeInfo,TypeAddress,returnToTop,ChangeLanguage,SelectPeople,SelectEvents,SelectE
   Target task: AccessMembershipInfo [active]

8) Event: http://marte.cnuce.cnr.it/sigchi/ita/iscrizione.htm SUBMIT
   The event is not associated with any task
```

Figure 7: Simulation of log file with the support of the task model.

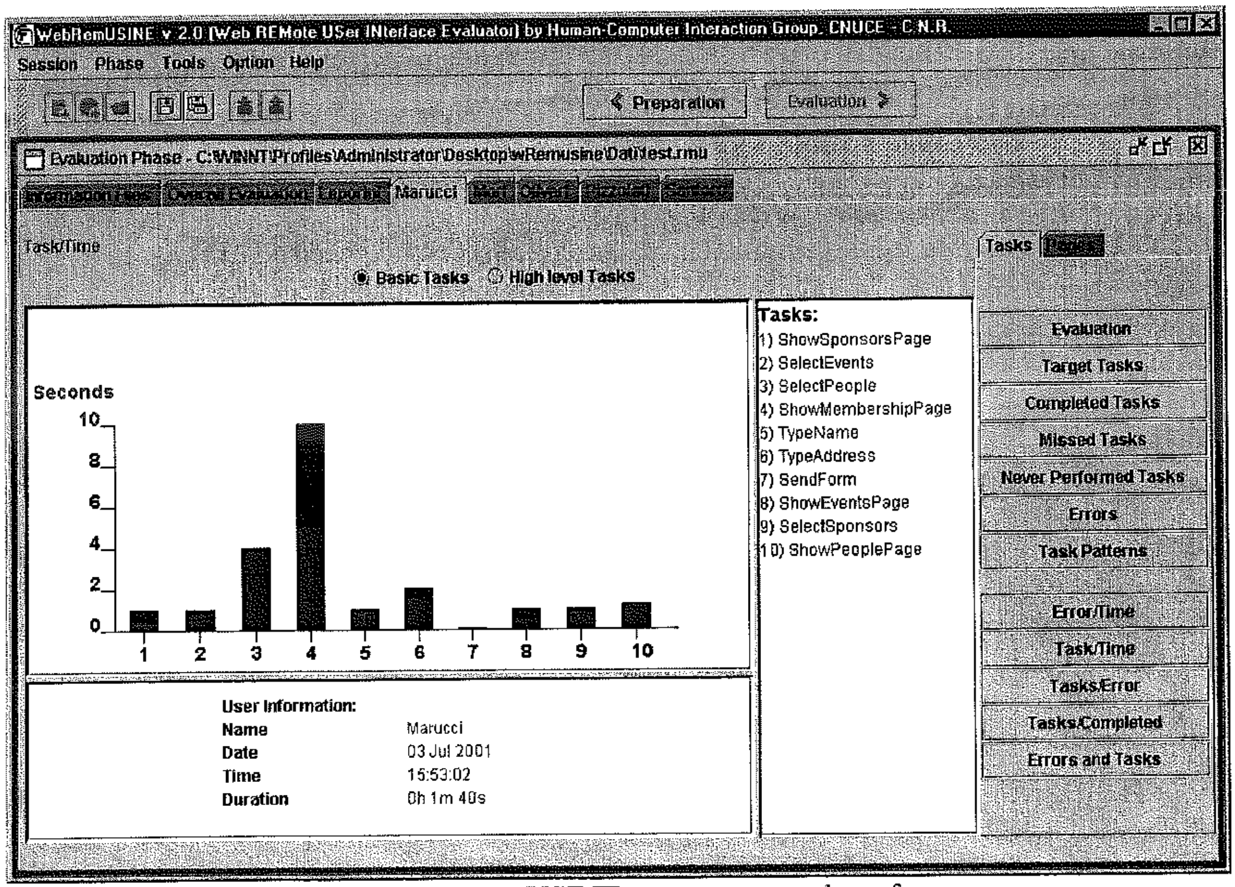
In the automatic analysis of the target high-level tasks five possible results can be achieved:

- Success, the user is able to perform a sequence of basic tasks that allows achieving complete performance of the high-level task;
- Failure, the user starts to perform the required basic task to achieve the current goal but he is unable to complete its performance;
- Warning type 1, it is possible to detect performance of basic tasks useless for the current goal but that do not preclude its completion;
- Warning of type 2, the user starts to perform correctly a high-level tasks but then some useless basic tasks are performed that disable the possibility of reaching the goal;
- Warning of type 3: the user has not been able to enable the performance of the target high-level task.

Regarding single sessions the tool provides various information:

- The list of basic tasks performed correctly with indication of the number of times they have correctly been performed;
- The list of basic tasks that have wrongly been performed for a precondition error with the indication of the number of times the error has been made;
- The list of tasks never performed correctly;
- The pattern of tasks (frequent sequence of tasks) that have correctly been performed during the session and indication of their frequency.

This information allows evaluators to easily identify what tasks create problems and what tasks are efficiently performed. The indication of tasks never tried is useful to identify parts of the user interface that are difficult to reach. In the case of frequent task patterns, the designer can decide to change the design in order to make their performance simpler and faster.



Tasks:
1) ShowSponsorsPage
2) SelectEvents
3) SelectPeople
4) ShowMembershipPage
5) TypeName
6) TypeAddress
7) SendForm
8) ShowEventsPage
9) SelectSponsors
10) ShowPeoplePage

User Information:

Name	Marucci
Date	03 Jul 2001
Time	15:53:02
Duration	0h 1m 40s

Figure 8: WebRemUSINE output: user task performance.

The tool is able to provide the time of task performance for both basic and high-level tasks. The time for the high-level tasks is given by the sum of the performance of all the composing basic tasks. Since one task can be performed multiple times during a session, the tool shows minimum, maximum and average time of performance through different colours in the bar associated with the task (see Figure 9). By selecting one bar it is possible to get detailed information on the time performance in each execution.

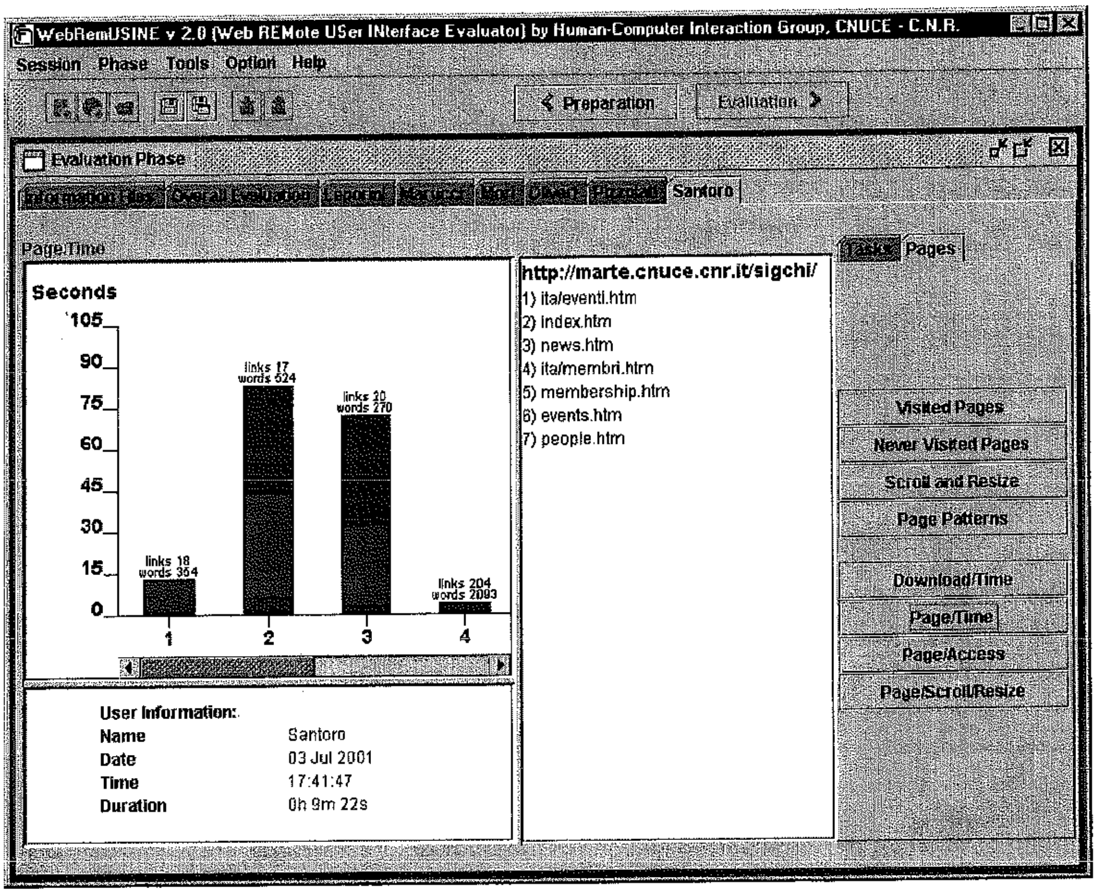
The tool also provides temporal information regarding when errors occur. This is useful to understand whether user performance improves during the test. For example, if errors are concentrated in the initial phase of the test it can mean that the user interface is easy to learn to use.

Regarding the navigation among pages the tool is able to determine the following information:

- The visited pages and the number of accesses;
- The visit patterns during navigation and their frequency;
- The time of downloading and visit of each page.

Analysing the number of accesses to the web pages is interesting. Pages accessed very frequently may indicate a rigid design. For example, if the access to the various parts of the site requires always selection of the home page, this page will have a high number of accesses and this solution would be inefficient since it would be faster to provide the list of the possible parts of the site available at any time. On the other hand, pages rarely accessed indicate parts of the site that are either not interesting or difficult to reach. If the same problem occurs for many users then it becomes important to redesign the site to better support access to this information.
Patterns of pages accessed are another important aspect to analyse especially if they contain errors that occur frequently in various users' sessions.

The analysis of the time can indicate many usability problems. If transferring a page takes too long then it is possible to identify too large files. For example, if the loading of images is often interrupted (generating an abort event captured by the logging tool) it is possible to understand that users do not like to wait too long to see them. In these cases it is better to reduce the dimensions to improve the site usability. The downloading time is calculated from when the user asks for the new page until the new page is completely loaded.
Time visit of a page is calculated from when a page is completely loaded in the browser until the user asks for a new page. The visit time depends on the structure of the page. Long pages containing a lot of textual information require from the user longer time to identify the required information. The visit time is affected also by the number of links in the page because users have to consider them to decide how to carry on the navigation. To allow evaluators to better analyse the visit time for each page the tool is also able to provide some measures (number of words and links contained) obtained through a static analysis of the HTML code to determine the complexity of the structure of the page as Figure 9 shows.



http://marte.cnuce.cnr.it/sigchi/
1) ita/eventi.htm
2) index.htm
3) news.htm
4) ita/membri.htm
5) membership.htm
6) events.htm
7) people.htm

User Information:

Name	Santoro
Date	03 Jul 2001
Time	17:41:47
Duration	0h 9m 22s

Figure 9: WebRemUSINE output: page visit times.

The tool is also able to analyse not only single sessions but also group of sessions and then provide statistical and summary information concerning them. For example, it provides both the average and the standard deviation of:

- Total time taken by the user session;
- Number of completed tasks;
- Number of errors subdivided into precondition errors and other errors;
- Number of scrollbar movements and change dimensions events.

Regarding basic tasks, the following averages are calculated on the group of users and showed listed in decreasing order:

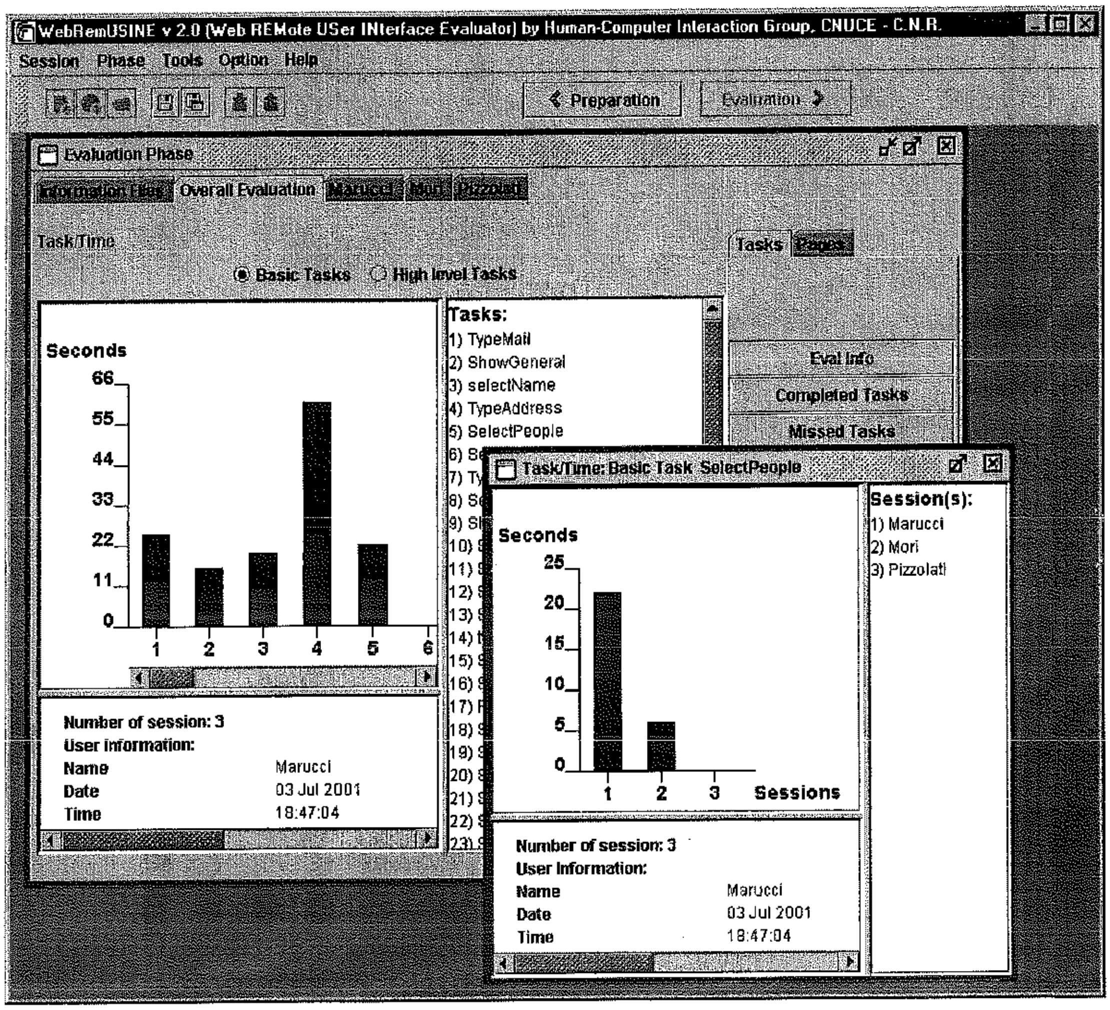
- Number of correct performance, here first tasks completed correctly from all users are shown and then the others to better highlight those that do not create problems to any user;
- Number of times a precondition error has been generated;
- Frequency of a task pattern.

Figure 10 shows the performance time regarding a group of sessions and in detail the performance time of the *SelectPeople* task.

Regarding evaluation of single pages the following average values are calculated on the number of users:

- Average number of accesses to each page;
- Average frequency of patterns;
- Average downloading time;

- Average visit time.



Tasks:
1) TypeMail
2) ShowGeneral
3) selectName
4) TypeAddress
5) SelectPeople

Number of session: 3
User information:
Name Marucci
Date 03 Jul 2001
Time 18:47:04

Session(s):
1) Marucci
2) Mori
3) Pizzolati

Number of session: 3
User information:
Name Marucci
Date 03 Jul 2001
Time 18:47:04

Figure 10: WebRemUSINE output: task performance in a group of users.

Conclusions

In the paper we have shown how it is possible to perform remote testing of Web sites and analyse the results with the support of automatic tools. We have implemented the logging tool in Javascript while WebRemUSINE has been implemented in Java.

Once the initial preparation phase has been completed, this approach allows evaluators to analyse large number of sessions without additional effort.

Future work will be dedicated to the application of our tool to the evaluation of large web sites to determine whether the preparation phase in these cases may be inordinately long, thus calling for improvements in order to make the tool easier to apply.

References

[CS99] J. Cugini, J. Scholtz *VISVIP: 3D visualization of paths through web sites*. Proceedings of the International Workshop on Web-Based Information Visualization (Florence, Italy, September 1999), pp. 259-263. Institute of Electrical and Electronics Engineers. http://www.itl.nist.gov/iad/vug/cugini/webmet/visvip/webvis-paper.html

[EC99] M. Etgen, J. Cantor *What does getting WET (WebEvent-logging Tool) mean for web usability?*. Proceedings of HFWeb'99 (Gaithers-burg, Maryland, June 1999). http://zing.ncsl.nist.gov/hfweb/proceedings/etgen-cantor/index.html.

[F00] P. Faraday *Visually critiquing web pages*. Proceedings of HFWeb'00 (Austin, TX, June 2000). http://www.tri.sbc.com/hfweb/faraday/faraday.htm

[G98] D. Goodman *Client-side persistence without cookies*. May 1998.

http://developer.netscape.com/viewsource/goodman_nocookies/goodman_nocookies.html
[HCK96] R. Harton, J. Castillo, J. Kelso, J. Kalmer, W. Neale *The Network as an estension of the usability laboratory.* Proceedings of CHI'96 (Vancouver, Canada, April 1996), pp. 228-235.
http://www.acm.org/sigchi/chi96/proceedings/papers/Hartson/hrh_txt.htm
[IH00] M. Ivory, M. Hearst *State of the Art in Automated Usability Evaluation of User Interfaces.* 2000. University of California, Berkeley
[LPT99] G. Lynch, S. Palmiter, C. Tilt *The max model: A standard web site user model.* Proceedings of HFWeb'99 (Gaithers-burg, Maryland, June 1999). http://www.nist.gov/itl/div894/vvrg/hfweb/proceedings/lynch/index.html
[PB00] F. Paternò, G. Ballardin *RemUSINE: a bridge between empirical and model-based evaluation when evaluators and users are distant.* Interacting with Computers, Vol.13, N.2, 2000, pp. 151-167.
[P99] F. Paternò, Model-based design and evaluation of interactive applications, Springer Verlag, 1999. ISBN 1-85233-155-0.
[SLD98] J. Scholtz, S. Laskowski, L. Downey *Developing usability tools and techniques for designing and testing web sites.* Proceedings HFWeb'98 (Basking Ridge, NJ, June 1998).
http://www.research.att.com/conf/hfweb/proceedings/scholtz/index.html
[TecEd99] *Assessing Web Site Usability from Server Log Files.* Prepared by Tec-Ed, Inc. December 1999
[WebCriteria99] WebCriteria. *Max, and the objective measurement of web sites.* December 1999.
http://www.webcriteria.com/pdf/max_102.cfm
[BURL] Bobby. http://www.cast.org/bobby
[NRURL] NetRaker Suite. http://www.netraker.com/nrinfo/products/index.asp
NIST Web Metrics. http://zing.ncsl.nist.gov/WebTools