Behavior Research Methods, Instruments, & Computers
2003, 35 (3), 369-378

Tools for remote usability evaluation of Web applications through browser logs and task models

LAILA PAGANELLI and FABIO PATERNÒ
Istituto di Scienza e Tecnologie Dell'Informazione "Alessandro Faedo," Pisa, Italy

The dissemination of Web applications is extensive and still growing. The great penetration of Web sites raises a number of challenges for usability evaluators. Video-based analysis can be rather expensive and may provide limited results. In this article, we discuss what information can be provided by automatic tools able to process the information contained in browser logs and task models. To this end, we present a tool that can be used to compare log files of user behavior with the task model representing the actual Web site design, in order to identify where users' interactions deviate from those envisioned by the system design.

Creating a Web site allows millions of potential users, who have diverse goals and knowledge levels, to access the information that it contains. Although a Web site can easily be developed by using one of the many tools available that are able to generate HTML from various types of specifications, obtaining usable Web sites is still difficult. Indeed, when users navigate through the Web, they often encounter problems in finding the desired information or performing the desired task. With over 30 million Web sites in existence, Web sites have become the most prevalent and varied form of human–computer interface. At the same time, with so many Web pages being designed and maintained, without automation there will never be a sufficient number of professionals to adequately address usability issues (Ivory & Hearst, 2001) as a critical component of their approach. For these reasons, interest in automatic support for usability evaluation of Web sites is rapidly increasing (Card et al., 2001; Scholtz, Laskowski, & Downey, 1998).

With the advent of the Web and the refinement of instrumentation and monitoring tools, user interactions are being captured on a much larger scale than ever before. In order to obtain meaningful evaluation, it is important that users interact with the application in their daily environment. Since it is impractical to have evaluators directly observe users' interactions, interest in remote evaluation has been increasing. In addition, recent studies (Tullis, Fleischman, McNulty, Cianchette, & Bergel, 2002) have confirmed the validity of remote evaluation in the field of Web site usability. Some work (Lister, 2003) in this area has been oriented to the use of audio and video capture for qualitative usability testing. In our case, we provide quantitative data and support for their intelligent analysis. The goal of this paper is to present a tool for performing remote usability evaluation of Web applications, without requiring expensive equipment. We describe how it supports analysis of task performance and Web pages accesses by single users and groups of users.

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 behavior is analyzed 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 video analysis also requires time and effort (in our experience, it takes five times the duration of the session recorded), and some aspects of the user interaction can still be missed by the evaluator (such as rapid mouse selections).

In model-based evaluation, evaluators apply user or task models to predict interaction performance and identify possible critical aspects. For example goals, operators, methods, and selection (GOMS) rules (John & Kieras, 1996) have been used to describe an ideal error-free performance. Model-based approaches can be useful, but the lack of consideration for actual user behavior can generate results that do not agree with real user behavior.

It is important to compare models with empirically observed performance. To this end, the main goals of our work are to support remote usability evaluation in which users and evaluators are separated in time and/or space, to analyze possible mismatches between actual user behavior and the design of the Web site represented by its

Correspondence concerning this article should be addressed to F. Paternò, Consiglio Nazionale delle Ricerche, Istituto di Scienza e Tecnologie Dell'Informazione "Alessandro Faedo," Via G. Moruzzi, 56100 Pisa, Italy (e-mail: fabio.paterno@cnuce.cnr.it).

task model, in order to identify user errors and possible usability problems, and to provide a set of quantitative measures (such as execution task time or page download time), for individuals and group of users, useful for identifying usability problems.

Since we perform remote evaluation without direct observation of 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, whose effectiveness is strongly limited because they cannot capture accesses to pages stored in the browser cache. Moreover, server logs completely miss local user interactions with the interface techniques (menus, buttons, fill in text, etc.).

In order to understand what the user goal is, during navigation, a list of all the possible activities supported by the site is displayed in a separate window. The user is supposed to select the target task from the list, which is derived from the task model corresponding to the actual design of the Web site.

Web Remote User Interface Evaluation (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 (Figure 1). The method is composed of three phases: *preparation*, which consists of creating the task model of the Web site, collecting the logged data, and defining the association between logged actions and basic tasks; *automatic analysis*, during which WebRemUsine examines the logged data with the support of the task model and provides a number of results concerning the performed tasks, errors, and loading time, which are displayed in various formats, both textual and graphical; and *evaluation*, during which the information generated is analyzed by the evaluators to identify usability problems and possible improvements in the interface design.

The architecture of our system is mainly composed of three modules: the ConcurTaskTrees (CTT) 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; and WebRemUsine, a Java tool able to perform an analysis of the files generated by the logging tool, using the task model created with the CTT editor tool.

Task models describe the activities a user must perform in order to reach his or her goals. We have used the CTT notation (Paternò, 1999) to specify them. CTT is a notation that provides a graphical representation of the hierarchical logical structure of the task model. In addition, the set of temporal and logical relations among tasks and task attributes that can be specified with CTT is larger than the corresponding set in traditional hierarchical task analysis. In particular, it is possible to specify a number of flexible temporal relations among such tasks (concurrency, choice, enabling, disabling, suspend–resume, order independence, optionality, etc.), and for each task, it is possible to indicate the objects to be manipulated and a number

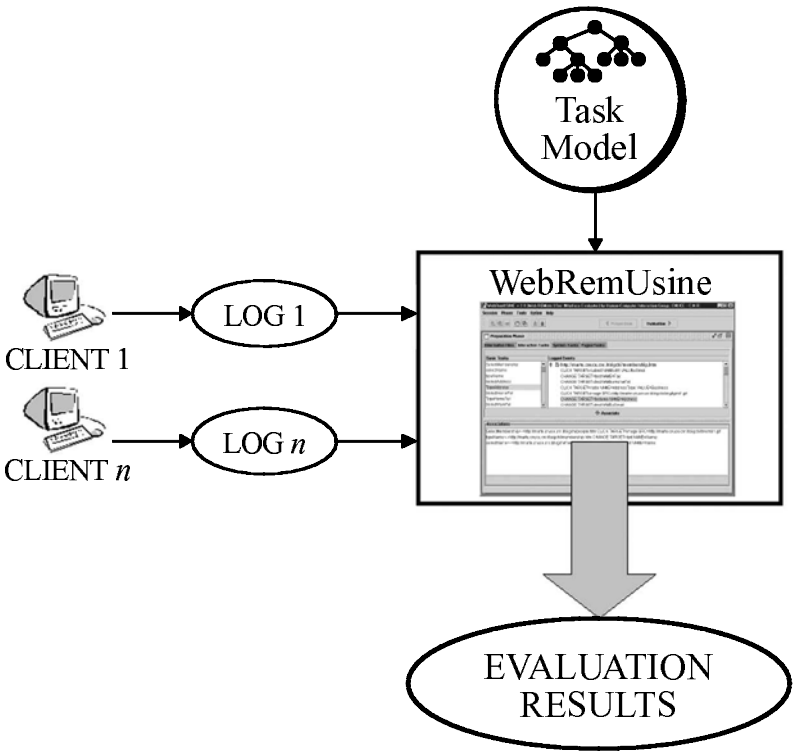

Figure 1. The overall architecture of WebRemUsine.

of other attributes. The notation also allows designers to indicate how the performance of the task should be allocated (to the user, to the system, or to their interaction) through different icons.

The logging tool stores various events detected by a browser, using JavaScripts that are encapsulated in the HTML pages and 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. Then a Java applet stores the log files directly in the application server. Our browser logging tool overcomes the limitations of other approaches to logging: Server logs have limited validity, since various page accesses are hidden to them because of browser cache memory and they do not detect the local interactions with the user interface elements (check-boxes, type in fields, etc.), which also cannot be detected with proxy-based approaches (Hong & Landay, 2001). Our tool works for the two main Web browsers (Microsoft IE and Netscape Communicator).

The WebRemUsine analysis can detect usability problems, such as tasks requiring a long performance time or tasks not performed according to the task model corresponding to the Web site design. The task model describes how activities can be performed according to the current design and implementation. Either the designer or the evaluator can develop it. Since the CTT notation has various temporal and logical operators, it is possible to describe all the various paths the user can follow to accomplish a task. Deviations from the expected paths can be detected in the logs and represent useful information for identifying any pages that create problems for the user.

Moreover, the tool analysis provides information concerning both tasks (such as time performance, errors, etc.) and Web pages (such as download and visit times, etc.). These results allow the evaluator to analyze the usability of the Web site from both viewpoints—for example, by 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 whether 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 which tasks have been completed, which have been started but not completed, and which have never been tried. This information is also useful for Web pages: Never-accessed Web pages can indicate either that such pages are not interesting or that they 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 for understanding whether a certain problem occurs often or is limited to specific users in particular circumstances.

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 analyze the sequence of user interactions. Basic tasks are tasks that cannot be further decomposed, whereas high-level tasks are complex activities composed of subactivities. The log files are composed of sets 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, two events are stored in the log: one associated with the selection of the Submit button and the other with the actual transmission of the form. Thus, in the task model, two basic tasks are required: one interaction task for button selection and one system task for form transmission. Otherwise, the model is incomplete.

In the logs, there are three types of events: user-generated events (such as *click* or *change*), page-generated events (associated with loading and sending of pages and forms), and events associated with a change in the target task by the user, which is explicitly indicated through selection from the list of supported tasks.

Tasks can belong to three different categories, according to the allocation of their performance: User tasks are cognitive activities that are only internal and that, thus, cannot be captured in system logs; interaction tasks are associated with user interactions (*click*, *change*, etc.); 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 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 for simplifying 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.

Analysis of Task Performance

During the test phase, all the user actions are automatically recorded, including those associated to goal achievement. The evaluation performed by WebRemUsine consists mainly in analyzing such sequences of actions to determine whether the user correctly performed the tasks complying with the temporal relationships defined in the task model or whether some errors occurred. In addition, the tool evaluates whether the user was able to reach the goals and whether the actions performed were actually useful for reaching the predefined goals. In order to determine whether the sequence of tasks performed complied with the task model, we use an internal simulator. For each action in the log, first, the corresponding basic task is identified, and next, there is a check to see whether the task was logically enabled. If not, a precondition error is identified; otherwise, a list of the basic tasks enabled after task performance is provided. Then

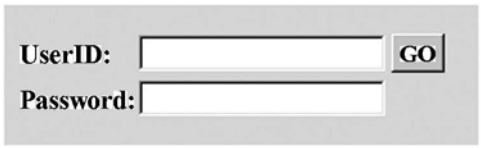

Figure 2. Example of problematic interface.

the list of high-level tasks already accomplished is updated and checked to see whether the target task has been completed.

In the report analyzing the user session, for each action, there is an indication as to whether it was correctly performed or a precondition error occurred. The analysis of user actions allows detection of problems generated by the execution task order. The precondition errors are highlighted when task performance does not respect the relations defined in the system design model, which leads to mismatches between the user's performance and the system task model.

Precondition errors may reveal some underlying usability problems. For example, if people want to access a remote service (such as Web access to e-mails), usually they have to provide username and password and then activate the request through a button. If the user interfaces elements that are not located in such a way that the user can easily realize that both fields have to be filled in before connecting to the mail box, some precondition errors can occur, and they can be detected through WebRemUsine. For example, Figure 2 shows a problematic user interface where the location of the Go button may lead some users to type in the user name and then press the Go button.

The presence of events not associated to any task can indicate parts of the interface that create problems for the user. For example, if the log contains events corresponding to images that are not associated with any link, evaluators will be able to understand that the image confuses the user. In this case, designers can change the page in such a way that it is clear that it has no associated function or can decide to associate a link with it.

In addition to the detailed analysis of the sequence of tasks performed by the user, evaluators are provided with some results that provide an overall view of the entire session considered: the basic tasks that were performed correctly and how many times they have been performed correctly; the basic tasks that the user tried to perform when they were disabled, thus generating a precondition error, and the number of times the error occurred; the list of tasks never performed either because they have never been tried or because of precondition errors; and the patterns (sequences of repeated tasks) that occurred during the session and their frequency. Such information allows the evaluator to identify which tasks are easily performed and which create problems for the user. Moreover, re-

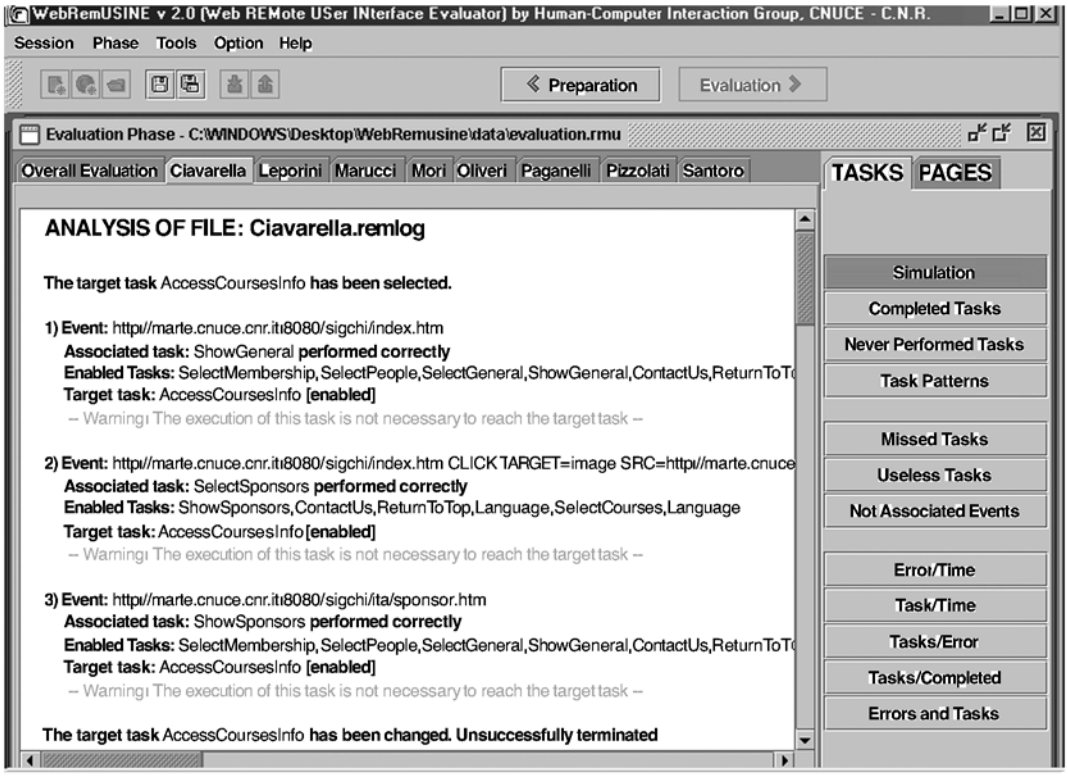

Figure 3. Example analysis of a session log.

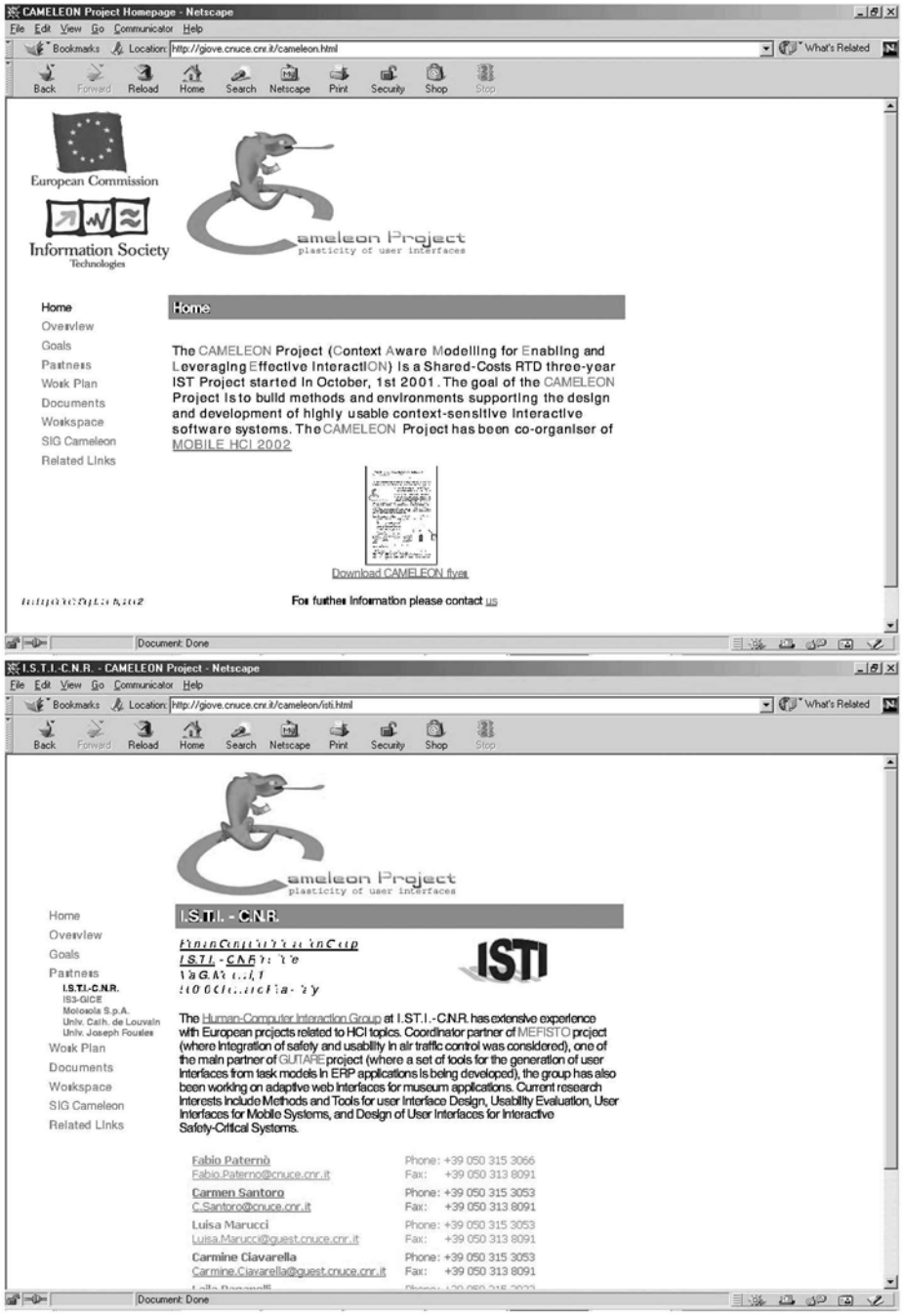

Figure 4. Example of Web application with horizontal structure.

vealing tasks that are never performed can be useful in identifying parts of the application that are difficult to comprehend or reach. On the basis of such information, the evaluator can decide to redesign the site, in order to reduce the number and complexity of the activities to be performed.

The main user goal is associated with a high-level task (called the *target* task), which allows the automatic identification of the basic tasks to be performed to reach it. The target task can be in various states: *enabled*, in which state the user can start performance of the associated basic tasks; *disabled*, in which state the user cannot perform any of the associated basic tasks according to the temporal relations defined in the task model and, thus, has to perform some other tasks in order to enable the target task; *active*, in which state the user has started

the performance of the associated basic tasks; and *completed*, in which state the user has accomplished correctly the target task.

The evaluation of the state of the target task is performed during the analysis of the log and is reported in the log analysis (see an example in Figure 3). Thus, after each user action, this state is reported. This also allows the identification of useless tasks.

During the test session, the user can change the target task at any time by selecting another target task. From the time of the selection until either another change of target task or the end of the session, each basic task performed by the user is analyzed to determine whether it is useful to accomplish the target task.

From the log analysis, the tool can generate various types of results: *success*, where the user has been able to perform a set of basic tasks required to accomplish the target task and, thus, achieve the goal; *failure*, where the user started the performance of the target task but was not able to complete it; *useless uncritical task*, where the user performed a task that was not strictly useful for accomplishing the target task, but did not prevent its completion; *deviation from the target task*, where the target task was enabled and the user performed a basic task whose effect was to disable it, which reveals a problematic situation, since the user was getting farther away from the main goal, in addition to performing useless actions; and *inaccessible task*, where the user was never able to enable a certain target task.

Figure 3 shows a part of the analysis of a user session log. At the beginning of the test, the user selects the target task *AccessCoursesInfo*. Then the user performs a number of actions that are useless for reaching the current goal by selecting a page that does not lead to the information desired. Thus, the user generates a number of events associated with basic tasks useless for the current goal. During this navigation, the user does not make any precondition errors but is not able to find the link for accessing the page containing information regarding the course and, thus, is not able to reach the goal.

Another type of information considered during the evaluation regards the task execution time. In the case of tasks correctly performed, the tool calculates the global time of performance. This information is calculated by examining the temporal information associated with each event and stored in the logs. The duration is calculated for both high-level and basic tasks. The time for high-level tasks is calculated by summing the time required to perform the associated subtasks. The resulting information is useful in that it provides an overall view that allows one to identify which complex activities require longer performance.

In calculating the task duration, the tool takes into account that a task can be iterative and the time performance can be different during different iterations within the

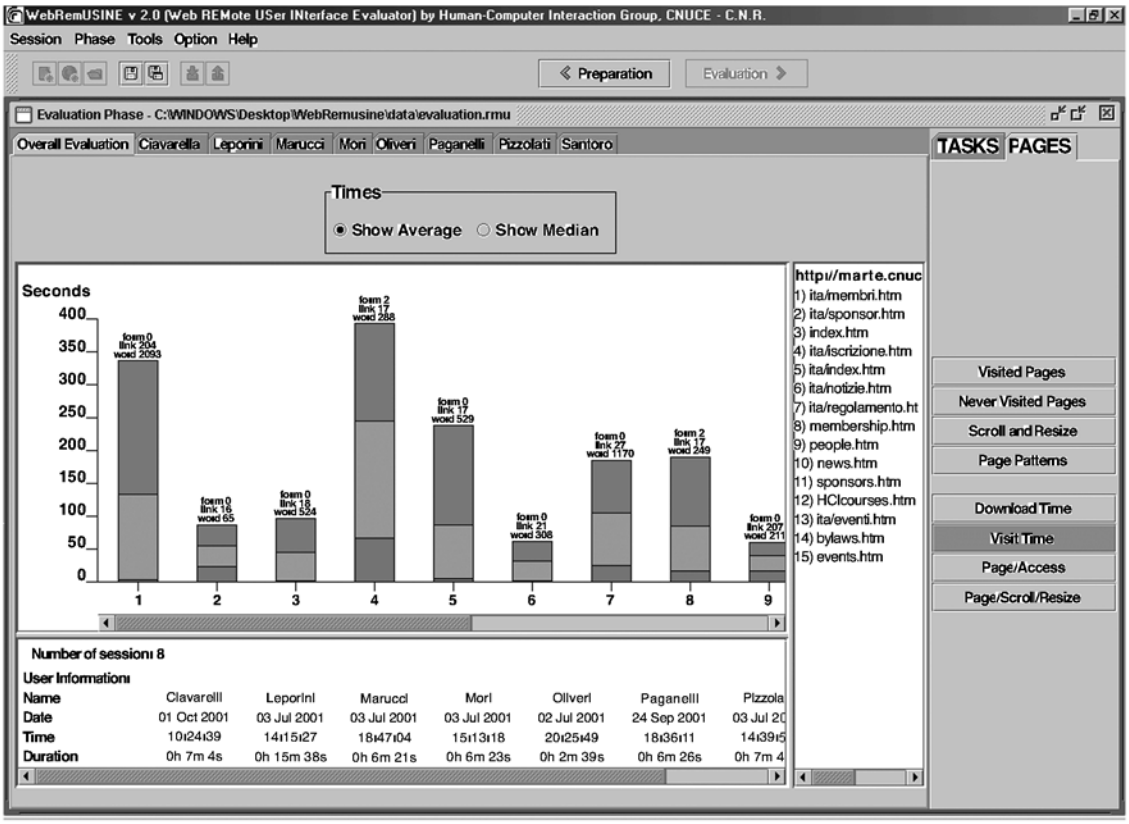

Figure 5. Example of analysis of Web pages.

same session. In case of multiple performance of a task during a session, the tool identifies the minimum, maximum, and average durations. The execution time is represented through bar charts in which each task is associated with a bar and different colors in the bar indicate minimum (blue), average (green), and maximum (red) times. It is also possible to get detailed information on each task performance of a given task by selecting the associated bar with the right button on the mouse. Thus, it is possible to know the number of times it has been performed and the duration of each performance. This can be done for both basic and high-level tasks.

The set of results regarding the execution time can provide information useful for understanding which tasks are the most complicated or which tasks require, in any event, longer performance times. A longer execution time does not always imply a complicated task; for example, in some cases, download time can be particularly high. WebRemUsine also provides detailed information regarding download time so that evaluators can know its influence on performance time.

A further type of evaluation concerns the time associated with actions that generate errors. By analyzing when errors occurred, it is possible to determine whether the user's performance improved over the session. If the errors were concentrated during the initial part of the test and their number decreased over time, we can assume that the user interface is easy to learn.

Analysis of Web Pages Accesses

Regarding the navigation in the pages, in the evaluation process it is possible to determine the following information: the pages accessed by the user and whether he or she made multiple accesses to the same page, the navigation patterns and their frequencies, and the download and visit times for each page.

In addition, for each page, the tool reports the number of times the scrollbar has been used and the number of times the window has changed size. These events are not considered during the log simulation but can provide useful information during the evaluation of the single pages of the site. For example, excessive use of the scrollbar can indicate that the page is not well structured and that the user has to scroll around often in order to find the desired information, or it may indicate that the pages are too long. In the latter case, it would have been better to split the information among multiple pages. Excessively long pages can be annoying because they do not allow users to examine immediately all the possible alternatives.

Understanding how users navigate and what paths they follow can be useful for solving possible design problems. From the analysis of the logs, it is possible to extract information regarding accessed pages and patterns occurring during the session.

Determining what pages are most accessed can be useful for identifying design limitations. For example, consider a site that, in the home page, contains an index that

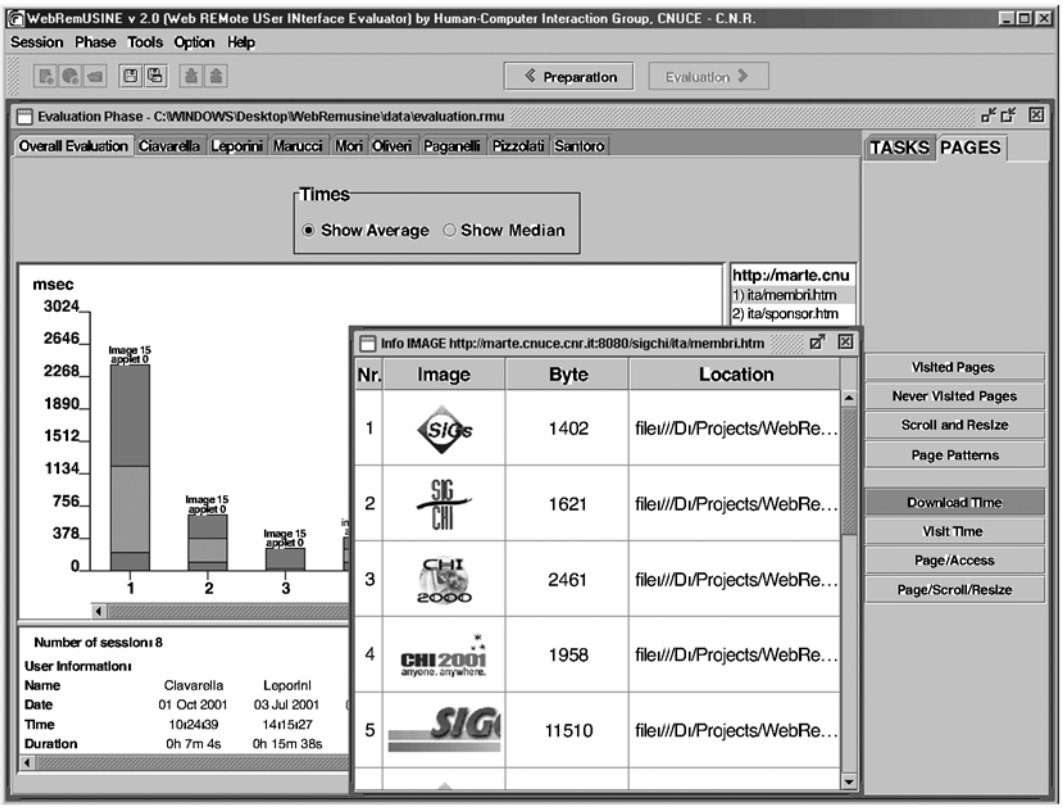

Figure 6. Example of access to details regarding page images through WebRemUsine.

can be used to access various groups of information. If in order to access a type of information, it is always required that the user pass through the home page, from the analysis of the visited pages during a session, the number of accesses to the initial page will be high. In such a case, it is possible to propose a horizontal structure so that the other pages of the site offer the possibility of directly accessing the other types of information, without requiring the user to pass through the home page. Figure 4 shows an example of this solution. In the home page (top panel), there is an index that allows access to the various types of information available. Such an index is provided also in the other pages of the site. In addition, where the current page is located in the structure of the site is also highlighted (ISTI-CNR is highlighted in black in the bottom panel). This allows users to immediately understand where they are in the site, avoiding going through the home page to perform a new access.

Also with respect to accessed pages, the identification of pages never accessed can highlight parts of the site that are either not interesting or difficult to reach. If this occurs for several users, it may be advisable to redesign the site to make access to such pages easier or to improve their presentation or the logical flow.

Lastly, the analysis can point out paths repeated by the user and their frequencies. In these cases, it can happen that users are not able to directly access the information that they are looking for and, each time, follow the same path in the pages while searching for the desired information. Patterns occurring in a single session are not easily interpretable, but if the same patterns occur in several sessions, this information can be useful for having an overall view on the paths preferred by users.

The response time of both the site and the user provide useful hints on various usability aspects. If the transfer time from the server to the browser is too long, it means that files are too large. For example, if image loading is often interrupted (causing introduction of an abort event in the log), it is possible to deduce that the user does not intend to wait so long to see the images. This is an indication that the dimensions should be reduced, thus improving the usability of the site. On the contrary, if the user spends too long a time on a page, it may not be easy to understand the reason: It may be because it is either very interesting or too difficult to understand.

Page visit time indicates the time spent from when the page is completely loaded in the browser until a new page request is sent. The visit time depends on the structure of the page: Very long pages containing a lot of text require longer time for the user to identify the desired information. The visit time is also affected by the number of links included on the page, because the user has to an-

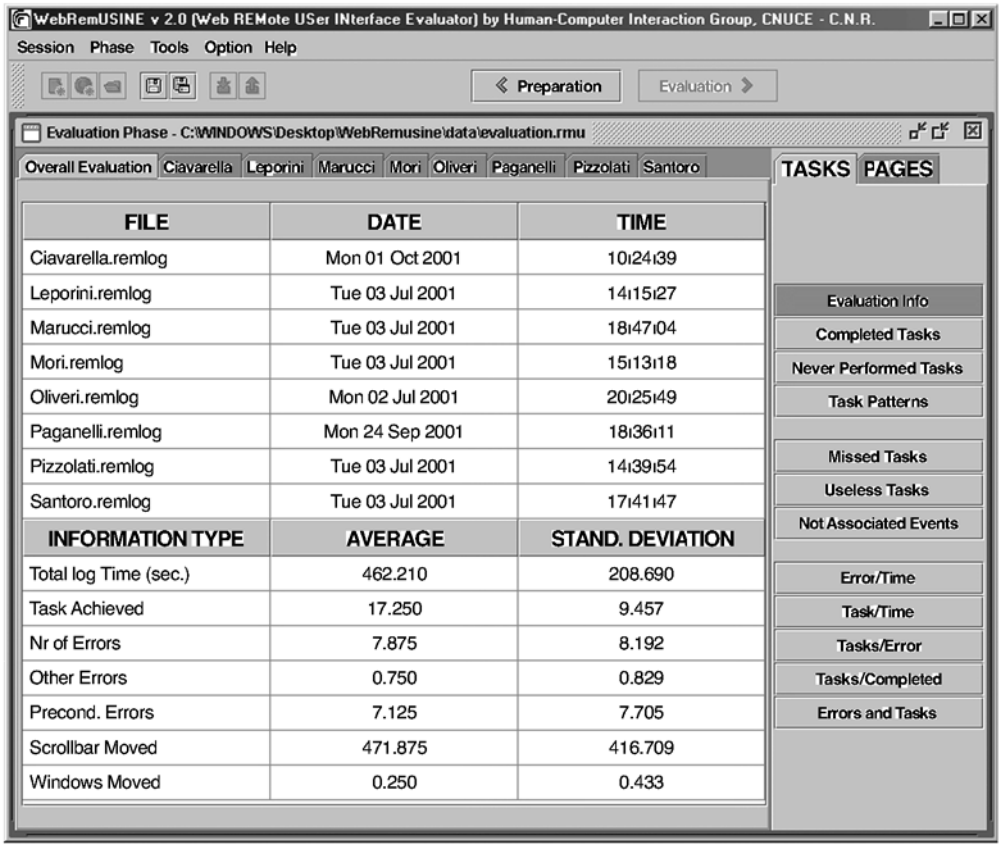

Figure 7. Example of summary information of a user group.

alyze them before deciding how to proceed in the navigation. Also, inclusion of forms in the page can affect the visit time. Indeed, forms require a careful reading and a correct input of data.

In order to allow evaluators to correctly evaluate the visit time, WebRemUsine determines some measures through an analysis of the HTML code, in order to indicate the complexity of the page. Thus, in addition to visit time, the tool calculates the number of words, the number of forms, and the number of links. It is reasonable to think that pages with a higher number of words and links can have longer visit times.

Figure 5 shows information on visit time with a bar chart in which each bar is associated with a page. As was discussed above, it is possible to have multiple accesses to the same page, so the graph indicates minimum, average, and maximum visit times. Each bar is also annotated with the number of words, links, and forms contained in the associated page.

The download time indicates the time spent from when the user asks for the new page until that page is completely loaded in the browser. Excessively long download times affect site usability negatively. It is important to provide information quickly so that users can feel free to move about in the information space. Some factors that can affect the download time is the presence of images or applets. For this reason, we have chosen to label the bars associated with download times with an indication of the number of applets and/or images contained in the page considered. It is also possible to access details regarding the images' sizes in terms of bytes (see Figure 6).

Analysis of Groups of Users

In the previous sections, we have examined the evaluations performed for each single user. The information associated with the various sessions can be grouped and compared, so as to determine a set of statistical data summarizing the set of sessions.

A first evaluation of the overall test results is provided by the average value and the standard deviation of the following (see Figure 7): session duration, number of complete tasks, number of errors, subdivided into precondition errors and errors derived from occurrences of events (in the figure, "Other Error") not associated with any task, and number of scrollbar movements and browser resizing.

When tasks performed at least once during the user sessions are considered, the following average values can be calculated with respect to the total number of users: the average numbers of times that a basic task has been performed correctly and that target tasks have been accomplished successfully, the average numbers of times that the performance of a basic task has generated a pre-

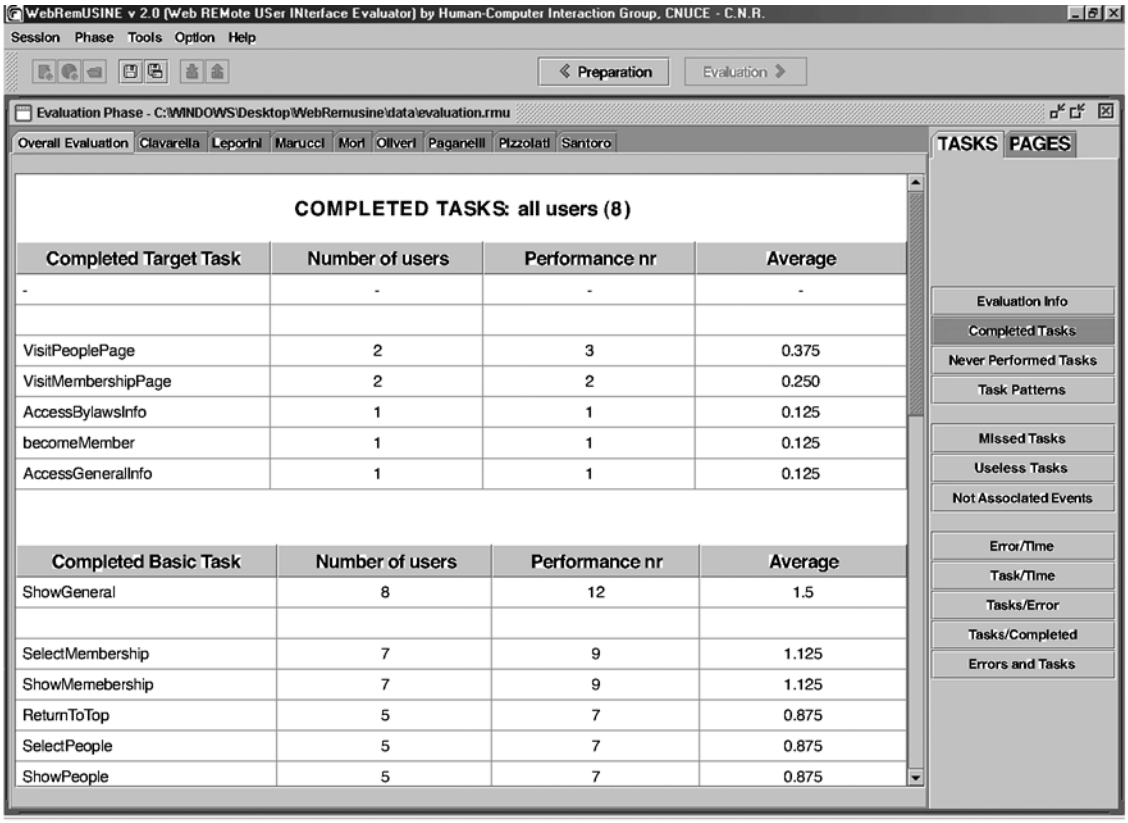

Figure 8. Table providing information regarding task performance in a group of sessions.

condition error and that target tasks have not been accomplished successfully, the average number of basic tasks performed that were useless in reaching the current goal, and the average frequency of a task pattern.

For example, Figure 8 shows two tables that summarize results related to tasks associated with goals successfully achieved and to basic tasks correctly completed in the sessions considered, giving the average number of times they were realized. The results in each table are ordered according to the average value associated with each task. In addition, as Figure 8 shows, the tables are subdivided into two parts: In the first group, tasks performed correctly by all users are reported, and the second group shows tasks executed by at least one user (but not by all of them). Thus, the evaluator can immediately identify which tasks create no problem, because all users manage to accomplish them.

As we noted above, a task in the task model can be performed multiple times. In the case of analysis of groups of sessions, for each task, the average of the execution time is determined as the sum of the average execution times for each single session, divided by the number of users that performed the task correctly at least once. The same type of information can be provided for both basic and high-level tasks.

Regarding the evaluation related to the pages examined in each session, the following average values are calculated with respect to the total number of users: the average number of accesses to a page, the average frequency of a task pattern, and the average download time.

We noted that for the analysis of multiple sessions performed by a group of users, it is often useful to consider the median time. The median time of a set of values is the midway value—that is, half the data is lower and the other half higher than the median. It furnishes more meaningful information than does the mean value. This allows the tool to determine the time spent on a page by users, without having singles values associated with one or a few users to excessively affect the overall evaluation.

Conclusions

The paper has discussed the results that can be obtained through a tool able to automatically analyze the information contained in Web browser logs and task models. Such information regards task performances and Web page accesses of single and multiple users. This allows evaluators to identify usability problems even if the analysis is performed remotely. The overall approach requires some effort in the preparation phase (task model development and association of actions in the logs with basic tasks). However, once the preparation phase has been completed, it is possible to easily analyze even high numbers of user sessions.

Future work will be dedicated to providing more effective representations of the automatic analysis results, through the application of information visualization techniques, and to extending the approach proposed to supporting the evaluation of mobile applications.

REFERENCES

Card, S. K., Pirolli, P., Van Der Wege, M. M., Morrison, J. B., Reeder, R. W., Schraedley, P. K., & Boshart, J. (2001). Information scent as a driver of Web behavior graphs: Results of a protocol analysis method for Web usability. In *Proceedings of ACM CHI 2001* (pp. 498-504). New York: ACM Press.

Hong, J. I., & Landay, J. A. (2001). WebQuilt: A framework for capturing and visualizing the Web experience. In *Proceedings of the 10th International World Wide Web Conference* (pp. 717-724).

Ivory, M. Y., & Hearst, M. A. (2001). The state of the art in automating usability evaluation of user interfaces. *ACM Computing Surveys*, **33**, 470-516.

John, B., & Kieras, D. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer–Human Interaction*, **3**, 320-351.

Lister, M. (2003). Streaming format software for usability testing. In *Proceedings of ACM CHI 2003* (Extended abstracts, 632-633). New York: ACM Press.

Paternò, F. (1999). *Model-based design and evaluation of interactive applications.* New York: Springer-Verlag.

Scholtz, J., Laskowski, S., & Downey, L. (1998, June). Developing usability tools and techniques for designing and testing Web sites. In *Proceedings of HFWeb'98*. Available at http://www.research.att.com/conf/hfweb/proceedings/scholtz/index.html.

Tullis, T. S., Fleischman, S., McNulty, M., Cianchette, C., & Bergel, M. (2002). *An empirical comparison of lab and remote usability testing of Web sites.* Paper presented at the Usability Professionals Association Conference, Orlando, FL.

(Manuscript received October 29, 2002;
revision accepted for publication May 18, 2003.)