Tools for Remote Usability Evaluation of Web Applications through Browser Logs and Task Models

Laila Paganelli, Fabio Paternò
ISTI-CNR
Pisa, Italy
{laila.pagnelli, fabio.paterno}@cnuce.cnr.it

Abstract
The dissemination of Web applications is enormous and still growing. This raises a number of challenges for usability evaluators. Video-based analysis can be rather expensive and may provide limited results. In this paper we discuss what information can be provided by automatic tools able to process information contained in browser logs and task models.

Introduction
Creating a Web site allows millions of potential users with various goals and knowledge levels to access the information that it contains. While developing a Web site is an activity that can be easily performed using one of the many tools available 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 many problems in finding the desired information or performing the desired task. For this reason, interest in usability evaluation of Web sites is rapidly increasing.
In order to obtain meaningful evaluation it is important that users interact with the application from their daily environment. This means that it is nearly impossible to have evaluators physically beside users to observe their interactions. Thus, interest in remote evaluation has been increasing.
The goal of this paper is to present a tool that shows how it is possible to perform remote usability evaluation of Web applications without requiring expensive equipment. In the paper we describe how it supports analysis of task performance and Web pages accesses of single 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 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 (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 GOMS (Goals, Operators, Methods and Selection rules) [3] 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;
- 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.

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 and, in order to understand what the user goal is during navigation, we have made always available the list of possible activities supported by the site from which the user can select the target task. 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*, 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. All results are displayed by WebRemUSINE in various formats both textual and graphical.
- *Evaluation*, the information generated is analysed 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 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.

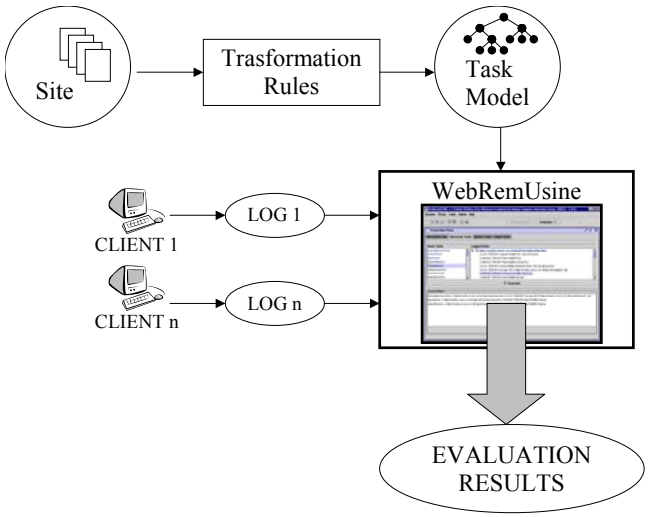

Figure 1: The Overall Architecture of WebRemUSINE.

Task models describe the activities to perform in order to reach user's goals. We have used the ConcurTaskTrees (CTT) [6] 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. 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, …) that also in the case of proxy-based approaches [4] cannot be detected.

The WebRemUSINE analysis can point out usability problems such as tasks with long performance or tasks not performed according to the task model corresponding to the Web site design. These elements are useful to identify the pages that create problems to the user. 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 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. 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 *clic*k, *chang*e), 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 *(clic*k, *chang*e, …) 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.

Analysis of task performance of single users

During the test phase all the user actions are automatically recorded, including those associated to the goals achievement. The evaluation performed by WebRemUsine mainly consists in analysing such sequences of actions to determine whether the user has correctly performed the tasks complying the temporal relationships defined in the task model or some errors occurred. In addition, the tool evaluates whether the user is able to reach the goals and if the actions performed are actually useful to reach the predefined goals. In order to determine whether the sequence of tasks performed is complying with the temporal relations defined in the task model we used 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 performance of that task was logically enabled. If not then a precondition error is identified. If yes, then the list of the enabled tasks after its performance is provided. In addition, also the list of accomplished high-level tasks after its performance is provided and it is used to check whether the target task has been completed.
In the report analysing the user session, for each action there is an indication whether it was correctly performed or a precondition error occurred. The analysis of the user actions allows the detection of problems generated from the execution task order. The precondition errors highlight what task performance did not respect the temporal relations defined in the model describing the system design and consequently mismatch between the user and the system task model occurred.

Precondition errors may reveal some underlying usability problems. For example, if people want to access a remote service (such as Web access to emails), usually they have to provide username and password and then activate the request through a button. If the user interface elements are not located in such a way that the user can easily realise that both fields have to be filled in before connecting to the mail box, then 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.

Figure 2: Example of problematic interface.

The presence of events not associated to any task can indicate parts of the interface that create problems to the user. For example, if in the log there are events associated to images that are not associated to any link then evaluators can understand that the image confuse the user. In this case designers can change the page in such a way that it is clear that has no associated function or decide to associate a link to 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 are 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 never tried or because of precondition errors.
- The patterns (sequences of repeated tasks) occurred during the session ad their frequency.

Such information allows the evaluator to identify what tasks are easily performed and what tasks create problems to the user. Moreover, the identification of tasks never performed can be useful to identify 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 trying to diminish the number of activities to perform and make the task performance required of the user simpler and easier.

The main user goal is associated to a high-level task (called target task), thus allowing the automatic identification of the basic tasks to perform in order to reach it.
The target task can be in various states:

- *Enabled*, the user can start performance of the associated basic tasks.
- *Disabled*, the user cannot perform any of the associated basic tasks according to the temporal relations defined in the task model. Thus, the user has to perform some other tasks in order to enable it.
- *Active*, the user has started the performance of the associated basic tasks.
- *Completed*, 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. Thus, after each user action this state is reported. This also allows the identification of useless tasks.

The user can change the target task at any time. During the test session 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 analysed to determine whether is useful to accomplish the target task.

During the log analysis various types of results can be generated:

- **Success**: the user has been able to perform all the basic tasks associated with the target task and thus achieve the goal.
- **Failure**: the users starts the performance of the target task but is not able to complete it;
- **Useless uncritical task**: the user performs a task that is not strictly useful to accomplish the target task but does not prevent its completion.
- **Deviation from the target task**: in a situation where the target task is enabled and the user performs a basic task whose effect is to disable it. This shows a problematic situation since the user is getting farther away from the main goal in addition to performing useless actions.
- **Inaccessible task**: when the user is never able to enable a certain target task.

Figure 3 shows a part of the analysis of a log of a user session. At the beginning of the test the user selects the target task *AccessCoursesInfo*. Then, the user performs a number of actions that are useless in order to reach the current goal. During this navigation the user does not perform any precondition error 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.

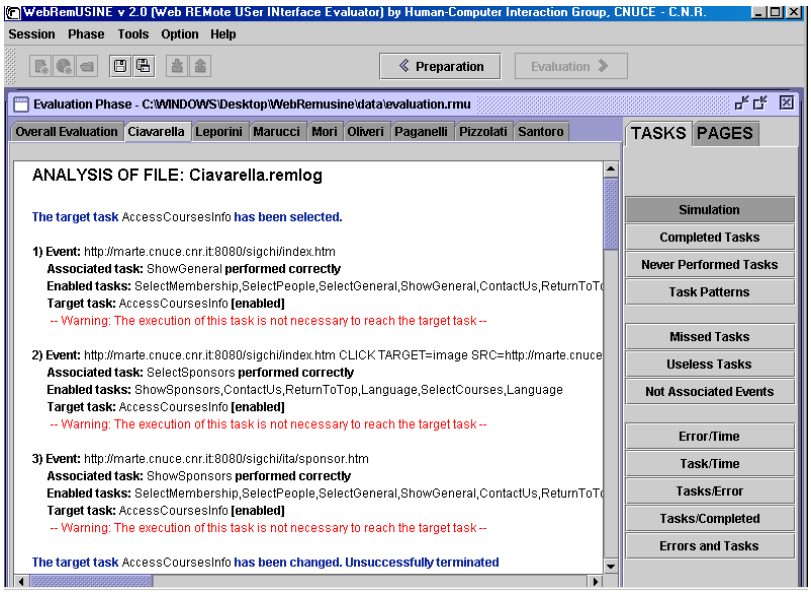

Figure 3: Example analysis of a session log.

A further 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 to have an overall view useful to identify what the complex activities that require longer activities are.
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 same session. In case of multiple performance of a task during a session, the tool identifies the minimum, maximum and average duration. The execution time is represented through bar charts where each task is associated with a bar and different colours in the bar indicate minimum (blu), average (green) and maximum (red) time.

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 mouse. Thus, it is possible to know the number of times it has been performed and the duration of each of them. This can be done for both basic and high-level tasks.

The set of results provided regarding the execution time can provide information useful to understand what the most complicated tasks are or what tasks require, in any event, longer time to be performed. Longer execution time does not always imply complicated tasks, for example in some cases downloading time can be particularly high. WebRemusine provides also detailed information regarding downloading time so that evaluators can know its influence on the performance time.

A further type of evaluation concerns the time associated with actions that generate errors. By analysing when errors occur it is possible to determine if the user performance improves over the session. If the errors concentrate during the initial part of the test, it is possible to understand that the user interface is easy to understand and the number of errors decreases over time.

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 they made multiple access to the same page.
- The navigation patterns and their frequency;
- The download and visit time 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, an excessive use of the scrollbar can indicate that the page is not structured linearly and the user has to scroll it often in order to find the desired information or can indicate pages too long. In the last case it would have been better to split the information on multiple pages. Too 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 to solve 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 highlight that the site design can be improved. For example, if we consider a site that in the home page contains the index to access various groups of information. If in order to access a type of information it is always required to 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 this case, it is possible to propose an horizontal structure, including also in the other pages of the site the possibility of accessing directly 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 (left-side) there is an index that allows access to the various types of information available. Such a 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 in black in the right side figure). This allows users to

immediately understand where they are in the site avoiding to go through the home page to perform a new access.

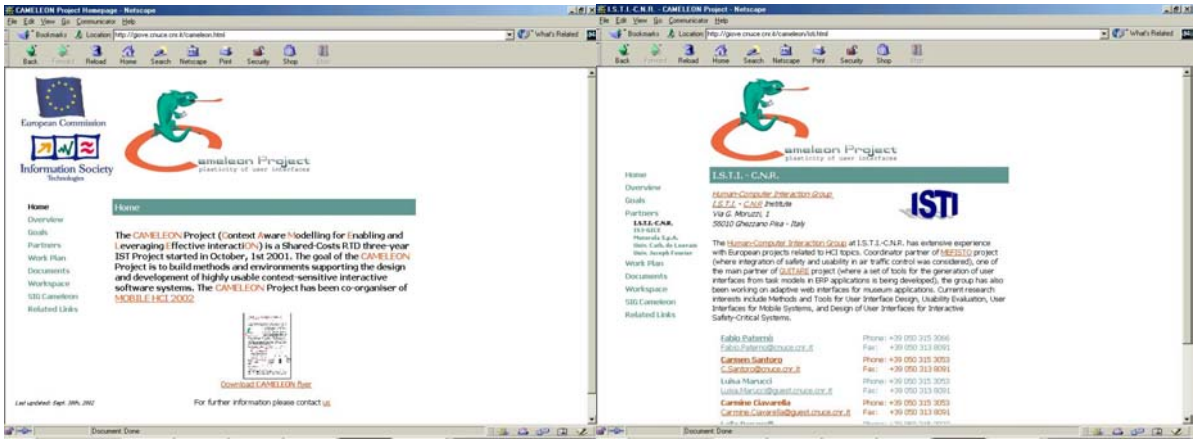

Figure 4: Example of Web Application with Horizontal Structure.

Still concerning the accessed pages, the identification of the pages never accessed can highlight parts of the site that are either not interesting or difficult to reach. If this occurs for several users, then it may be advisable to redesign the site in order 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 frequency. 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 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 to have an overall view on the paths preferred by the user.

The response time of both the site and the user provide useful hints on various usability aspects. If the transfer time is too long it means that files are too large. For example, if the image loading is often interrupted (causing the 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 to reduce the dimensions, 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.

The page visit time indicates the time spent from when the page is completely loaded in the browser until when 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 from 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 analyse them before deciding how to proceed in the navigation. Also the 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, in addition to the actual time, some measures through an analysis of the HTML code in order to determine the complexity of the page. Thus, in addition to the visit time the tool calculates the number of words container, the number of forms and links defined. It is reasonable to think that pages with a higher number of words and link can have longer visit time.

Figure 5 shows information regarding visit time through a bar-chart where each bar is associated with a page. As discussed before, it is possible to have multiple access to the same

page, thus also in this case we use different colours to indicate minimum, average and maximum visit time. Each bar is also annotated with the number of words, links and forms contained in the associated page.

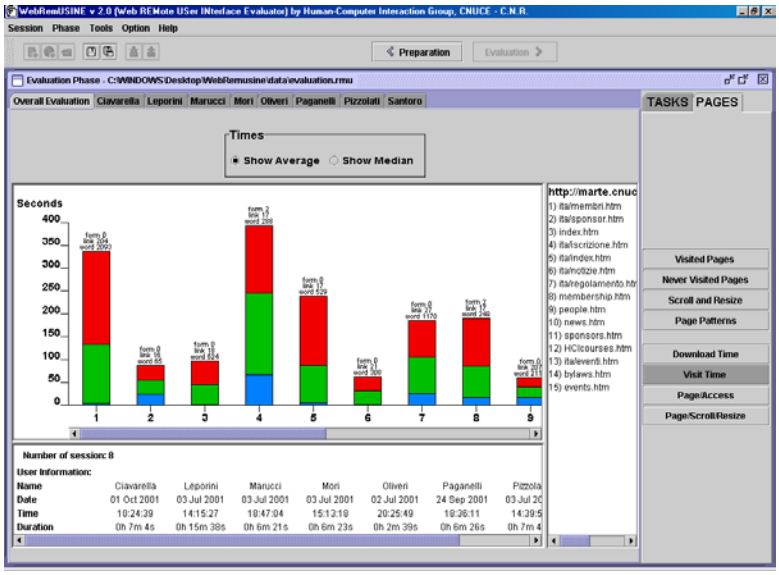

Figure 5: Example of Analysis of Web Pages.

The downloading 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 downloading times affect the 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 downloading 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).

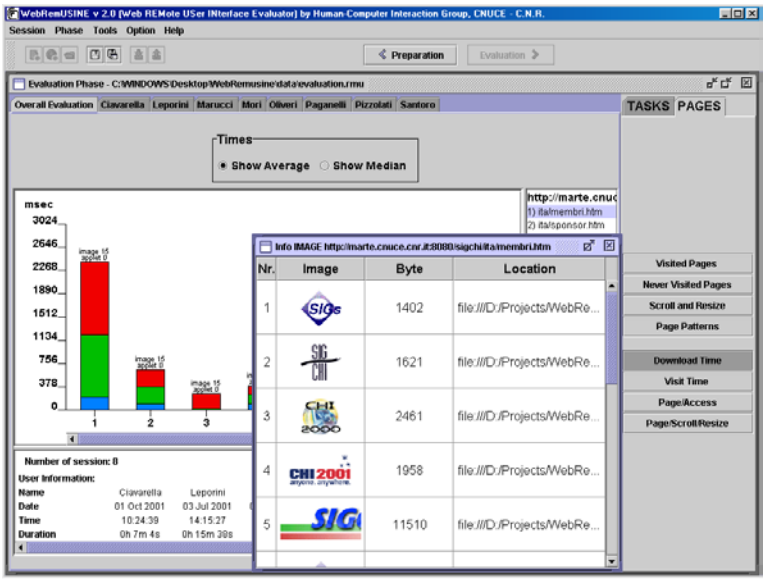

Figure 6: Example of access to details regarding page images through WebRemUsine.

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 summarising the set of sessions.

A first evaluation of the overall test results is provided by the average value and the standard deviation of (see Figure 7):

- Session duration.
- Number of complete tasks;
- Number of errors, subdivided in precondition errors and errors derived from occurrence of events (in figure "Other Error") not associated to any task.
- Number of scrollbar movements and browser resizing.

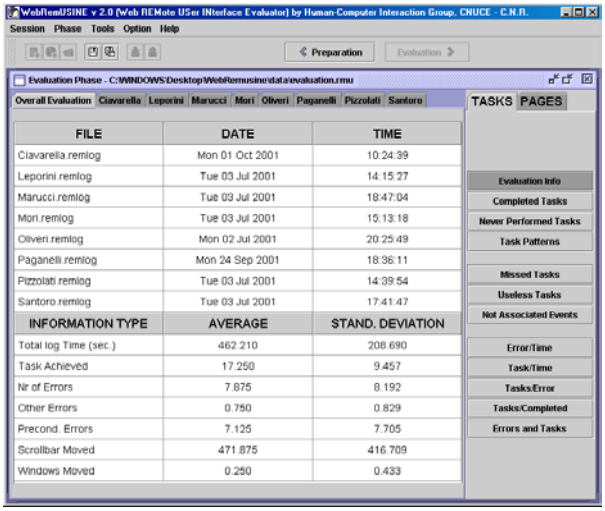

Figure 7: Example of Summary Information of a User Group.

Considering the tasks performed at least once during the user sessions, the following average values are calculated with respect to the total number of users:

- The average number of times that a basic task has been performed correctly and that target tasks have been accomplished successfully;
- The average number of times that the performance of a basic task has generated a precondition error and that target tasks have not been accomplished successfully;
- The average number of basic tasks performed that are useless to reaching the current goal.
- The average frequency of a task pattern.

For example, Figure 7 shows two tables that summarise the results related to the tasks associated with the goals successfully achieved (a), and the basic tasks correctly completed in the sessions considered, with the average number of times they have been realised (b). The results in each table are ordered according to the average value associated with each task. In addition, as Figure 7 shows, 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 what tasks create no problem because they are accomplished by all users.

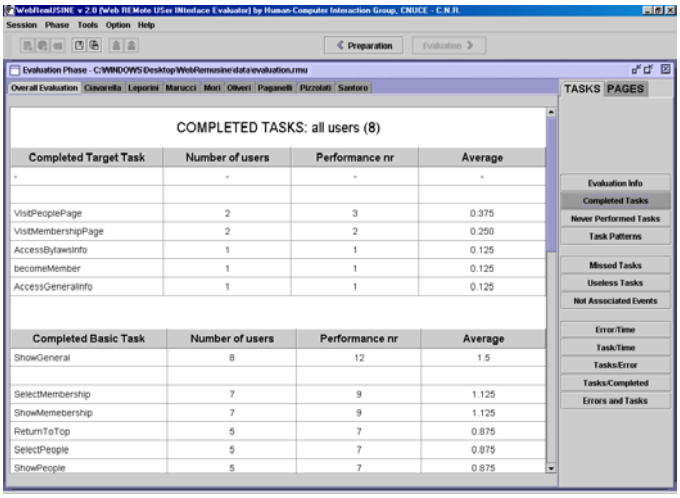

Figure 8: Table providing information regarding task performance in a group of sessions.

Even with regard to the times required to perform a task, the average values of execution time are calculated with respect to the set of session evaluated.
As we noticed before, 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 time for each single session, divided by the number of users that have 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 access to a page;
- the average frequency of a task pattern;
- the average downloading time.

We noticed 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 the mean value. This allows the tool to determine the time spent on a page by users, without having singles values associated to 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 analyse the information contained in Web browser logs and task models. Such information regards task performance 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 to basic tasks). However, once the preparation phase has been completed, then it is possible to easily analyse even high numbers of user sessions.

References

[1] S.Card, P.Pirolli, M. Van der Wege, J.Morrison, R.Reeder, P.Schraedley, J.Boshart, *Information Scent as a Driver of Web Behavior Graphs: Results of a Protocol Analysis Method for Web Usability*, Proceedings ACM CHI 2001, pp.498-504.

[2] M. Y. Ivory and M. A. Hearst (2001). *The state of the art in automating usability evaluation of user interfaces*. ACM Computing Surveys, 33(4), pp. 470-516, December 2001.

[3] B.John, D.Kieras, *The GOMS family of user interface analysis techniques: comparison and contrast*, ACM Transactions on Computer-Human Interaction, 3, 1996, pp.320-351.

[4] J. I. Hong, J. A. Landay, *WebQuilt: a framework for capturing and visualizing the web experience*. WWW 2001 conference: pp.717-724

[5] L.Paganelli, F. Paternò, *Intelligent Analysis of User Interactions with Web Applications*. Proceedings ACM IUI 2002, pp.111-118, ACM Press.

[6] F. Paternò, *Model-based design and evaluation of interactive applications*, Springer Verlag, 1999. ISBN 1-85233-155-0.

[7] 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