Automatic Reconstruction of the Underlying Interaction Design of Web Applications

L.Paganelli, F.Paternò
C.N.R., Pisa
Via G.Moruzzi 1
{laila.paganelli, fabio.paterno}@cnuce.cnr.it

ABSTRACT
In this paper we present a method, and the related tool, for analysing Web site code in order to automatically reconstruct the underlying logical interaction design. Such a design is represented through task models that describe how activities should be performed to reach user's goals. We also discuss how the result of this reverse engineering process can be provided as input to a tool for usability evaluation.

Keywords
Human-Computer Interaction, Reverse Engineering, Web Applications, Task Models.

INTRODUCTION
Nowadays, creating a Web site takes little effort. There are many tools that support automatic generation of Web pages from many possible formats. However, when we navigate the Web we often encounter problems finding the sought for information or accomplishing the desired tasks. This reveals how difficult it still really is to obtain usable Web sites.
Task models describe how activities should be performed in order to reach users' goals. They are the link between user interface designers and developers as they provide an abstract description able to highlight important information for both that can be used to analyse and discuss design solutions. They can also be used to support usability evaluation via various techniques (for example, to predict task performance, to compare predicted use with actual use, represented by logged sessions).
However, developing task models, as many modelling activities, can require some effort especially when large applications are considered. In addition, designers often have to analyse and evaluate applications developed by others without any logical representation of the design choices.
Often, when considering design of user interfaces researchers have paid attention to top-down approaches, whereby the designer first thinks about an abstract design and then moves on to the concrete design and lastly to the implementation. As already mentioned, there is often the need, in particular with Web applications, also for an inverse process: starting with an existing application developed by somebody else, designers have to reconstruct the underlying design decisions in order to better analyse them and propose an improved design.
In this paper we present the results of research that aims to support the possibilities of bottom-up approaches. This can be a useful complement to methods and tools able to support user interface generation from task models.
We will also discuss how the output of the tool developed can be used as input for Web site usability evaluation tools (such as WebRemUSINE [6]), thus, facilitating the usability evaluation of Web applications developed by other groups with fast development solutions.

RELATED WORKS
So far, little attention has been paid to support the development of models representing the design of interactive systems. Some approaches have addressed the issues related to how to derive such models from informal material used in the design phase, such as textual descriptions of scenarios of use. An example in this area is U-Tel [3], a tool supporting automatic identification of nouns and verbs that are then used as input for the domain and the task model, respectively, on the assumption that nouns indicate the objects that compose the domain model and verbs the activities considered in the task model. In our case we want to address a different issue: how to identify the task model starting with the Web pages composing the site considered.
A tool with more similar goals is Critique [5], which aims to support the development of KLM models [2] from user session logs. KLM models have a hierarchical description of sequential activities where the basic elements are the actions. Such models are also used to predict task performance on the basis of some cognitive studies that indicate estimated time to perform the types of actions considered. The rules for identifying the types of

actions from elements of user interaction logs are straightforward. Those for chunking the actions in order to identify the corresponding higher-level task are more elaborate. To this end, the rules proposed create a new chunk when users begin working with a new interactive objects or start to provide a different form of input to the current object (e.g. switch from clicking to typing in a text box). While this approach can provide useful information, it seems rather limited because the resulting model will reflect the actual use made by the user and moreover has no rule able to identify general temporal relationships among tasks (apart from sequentiality) and, in any event, does not address the specific aspects of Web applications.

Vaquita [1] addresses Web applications but is limited to identifying presentation models for the component pages, which mainly means the abstract description of the interaction techniques used in the implementation. This tool is therefore unable to reconstruct the task model associated with the Web application considered.

This review of some of the previous approaches to supporting reconstruction of relevant models for interactive systems design highlights the current lack that our work aims to fill: supporting reconstruction of task models of existing Web applications.

In the paper we first introduce the method that we have developed, next we provide a description of the rules that we have identified and implemented in the associated tool. Then, we move on to illustrate an example of applications of the method in order to clarify the approach. Lastly, we discuss how the resulting models can be used to support usability evaluation through another tool that we have developed and provide some concluding remarks.

THE APPROACH

The purpose of this work is to automatically reconstruct the task model of the Web application considered. In particular, we consider task models represented in ConcurTaskTrees. It is not the purpose of this paper to describe this notation, which is presented elsewhere [7]. Thus, we will briefly illustrate its features in order to allow readers unfamiliar with it to follow what is presented here. In this notation activities are represented in a hierarchical manner (as is the case for several task model notations). It uses different icons to represent task allocation (whether a task should be performed by the system or the user or their interaction) and has a rich set of temporal relationships that can be used to describe flexible and dynamic behaviours. These operators will be introduced during the explanation of the rules for the reverse engineering transformation. In addition, the notation has some features such as the possibility of describing the objects manipulated during task performance or task attributes (such as frequency, pre-condition and so on).

The tool (see Figure 1) receives as input the Web pages of the site. The site can be either in the local system or remote. The tool is able to automatically identify the pages composing it. Using the classes provided by Tidy [8] it first check that the HTML code is well-formed and if not it corrects it and then create the corresponding DOM [4] which describes the structure of the page (all the elements contained in it). Then, following the rules that we have developed and are illustrated in the next section it creates the task model associated with each page. The final part of the underlying algorithm is dedicated to describe higher-level tasks that involve multiple pages. The resulting task model can be saved either in XML format or in a format that can still subject to modifications or adjustments by the designer using the freely available tool (http://giove.cnuce.cnr.it/ctte.html) for this purpose that can help also to analyse the model with features such as dynamic interactive simulations.

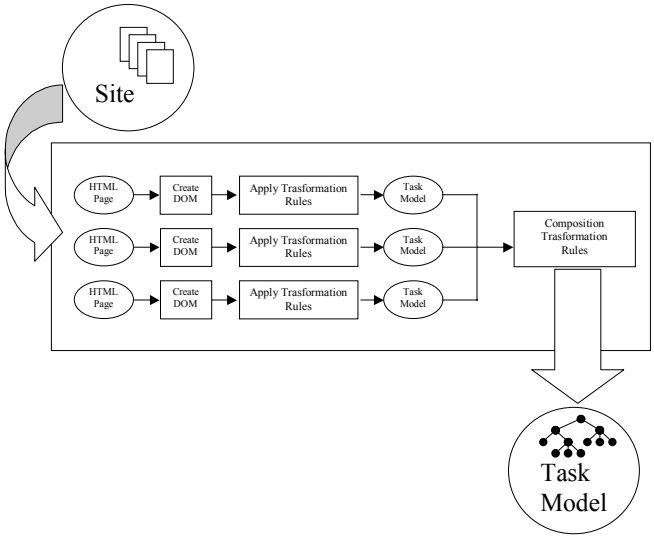


Figure 1: The structure of the tool.

THE RULES

In this section we describe the rules developed to reconstruct the underlying task model. We also aimed at identifying meaningful names for the tasks identified.

In HTML documents we can find many interaction elements (Links, Buttons, CheckBoxes, Radio Buttons, …) and elements for their logical grouping (Forms, Fieldsets..). Our rules analyse all the main elements that can be used in HTML Web pages and aim to identify the corresponding tasks, their mutual relationships in order to build the task model corresponding to the Web site considered.

R1 - Creation of initial task structure associated with the web page

When a new page is considered the tool starts to create a structure such as that in Figure 2. The task name of the root is "Access" + Title of the page. The first subtask is a basic application task (application tasks are represented by the computer icon), associated with the activity of loading the page and the name

"Load" + Title of the page. It is followed by an enabling operator (>> operator) and an abstract task named "Handle" + Title of the page (abstract tasks are represented by the cloud icon and indicate tasks that can be decomposed into more basic elements).
In particular, this task will be decomposed later on with the description of the activities that can be performed once the page has been loaded.

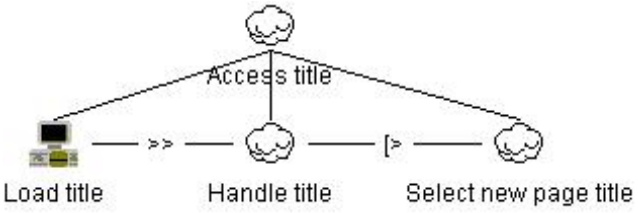


Figure 2: Initial structure of the single page task model.

The second subtask is followed by the disabling operator ([> symbol) indicating that it can be interrupted. In particular, the disabling activity will be the selection of a new page. The third subtask is an abstract task named "Select new page" + Title of the page.

Creation of task structure associated with link

Once we have created the initial structure of the task model, the next rules are dedicated to its refinement. In the analysis we considered the three possible types of links:

- links to anchors internal to the current page (R2);
- links to other pages still internal to the current Web site (R3);
- links to pages external to the current Web site (R3b).

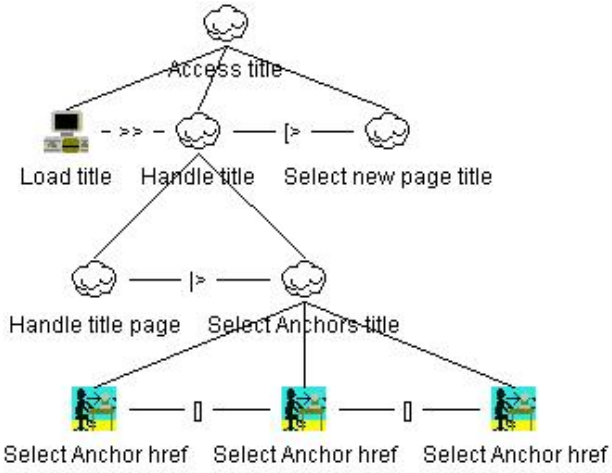


Figure 3: Decomposition for selection of links to internal anchors.

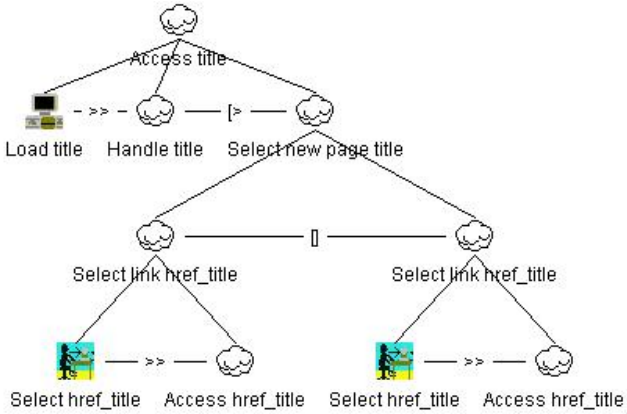
R2 - Creation of task structure associated with links to internal anchors

If there are links to internal anchors then the node "Handle" + Title of the page is expanded in the following manner. The first subtask is an abstract task followed by a suspend/resume operator (|> symbol) indicating that the left-hand activity can be interrupted by the right activity, but when the right activity is terminated then the first activity can be carried on from where it was left off. The second subtask is an abstract task named "Select Anchors" + Title of the page, which is decomposed in such a way that for each link to an anchor we have a basic interactive subtask called "Select Anchor " + name of internal anchor.

R3 - Creation of task structure associated with links internal to the current Web site

Similarly, there is a decomposition describing selection of internal links. For each internal link there is an abstract task with two subtasks that represent the user link selection and the set of tasks that can be performed on the page associated with the selected link.
The name of the abstract task is given by "Select" + Title or href. The first subtask is a basic interactive task named "Select" + Title or href followed by the enabling operator.
The second subtask is an abstract task called "Access" + Title of the page. All the generated subtrees are included as subtasks of the task named "Select new page" + Title and composed by the choice operator.

Figure 4: Decomposition for selection of links internal to the current Web site.



In the case of the presence of links to the same page, no subtree is built. Only a basic interactive task is added and the root becomes iterative to indicate that once the link is selected then the page is loaded and can be accessed again (see Figure 5).

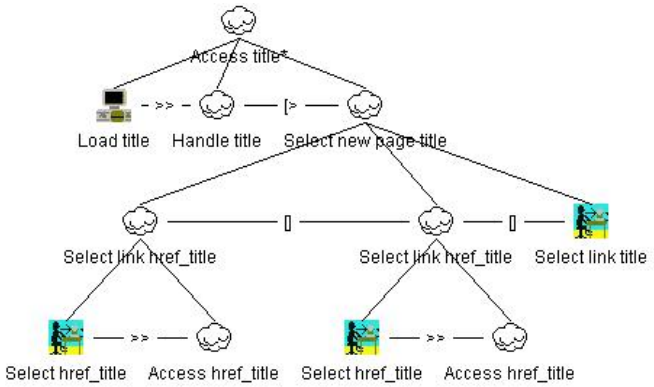


Figure 5: Decomposition with link to the same page.

R3-b Creation of task structure associated with links external to the current Web site

When a page contains at least one link to a page external to the current site, in the task model an interactive task, called Select External Link, is added as subtask of the node grouping all the subtasks defined through rule R3. This indicates that at this point the user leaves the site.

Interaction tasks

When navigating in a web application, link selection is not the only basic interactive task. A number of other interaction techniques may be available. These rules indicate how they are considered in the reconstruction process.

R4 - Each button is associated with the creation of a selection task in the model;

R5 - For each checkbox an iterative interactive basic task called "Select Checkbox " + name is created. It is iterative because users can select multiple alternatives.

R6 - For each radio element, a basic interactive task named "Select Radio" + name is created.

R7 - Creation of task structures associated with text areas

For each **Text** or **Text area** we build a subtree, with interactive root, composed of two basic interactive subtasks linked by an enabling operator. The second subtask is iterative to indicate that the user can type multiple characters. The name attribute of the HTML element is used to define the task names.

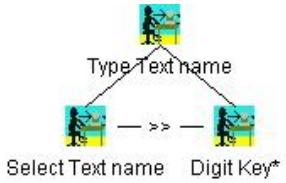


Figure 6: Decomposition for Text area elements.

R8 - Creation of a new task level associated with FieldSets

HTML provides some techniques to logically group related user interface elements, for example, the element FieldSets. To make this logical grouping explicit in the task model, all the tasks related to elements included in a tag **Fieldset** are grouped as subtasks of a new higher-level task. The name of the new task is derived from the tag **Legend**.

R9 - Creation of a new task structure associated with a Form

In HTML Forms contain the interactive part of a Web page. For each form we have a task structure in which all the tasks related to each of the form's input elements or fieldsets are inserted as subtasks of the *Compile Form name* task. Non-mandatory fields are associated with optional tasks. Input elements are composed by the order independence operator (|=| symbol) to indicate that they have to be performed but the order of performance is not predetermined. If the Form contains both Submit and Reset buttons then the resulting model has the structure indicated in Figure 7. If the Reset button is missing then the subtree *Reset Form name* will not be included.

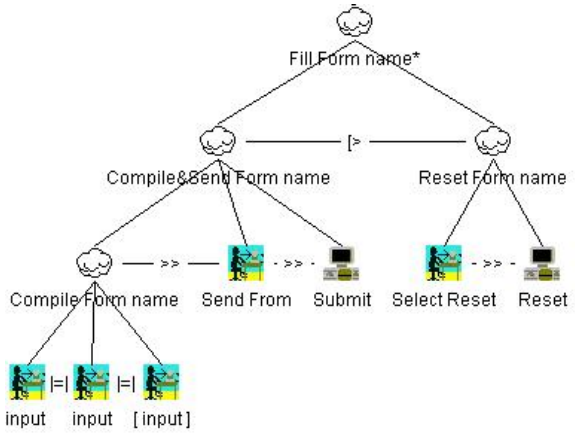


Figure 7: Decomposition for Form elements.

Additional rules

The nesting among **Form** and **Fieldset** elements is reflected in the hierarchical structure of the task model. The root of these elements is a subtask of the Handle element created through Rule 1.

All the tasks associated with input elements not included in a form or fieldset are included as subtasks of the Handle element created through Rule 1.

Creation of the task structure associated with the navigation bar

Once we have created the task model associated with the single pages, then there is the issue of creating the model's higher levels describing how the various parts are composed.

To this end, we consider the presence of structures such as a navigation bar common to all the pages. In this case the subtrees associated with each page are composed with a choice operator ([] symbol) and by a higher level task associated with accessing the site. Choosing from among the various pages can be interrupted by the high-level task whose recursion indicates that once a branch has been selected it is then still possible to select any other branch at any time.

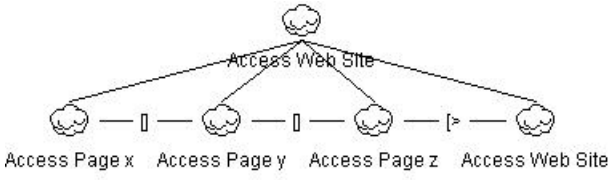


Figure 8: Overall structure for the entire site.

AN EXAMPLE

In order to illustrate the foregoing rules and their application let us consider the example of the Web site at SIGCHI http://giove.cnuce.cnr.it/sigchi/index.html

All pages of the site display a navigation bar that allows users to reach any part of the site at any time.

Figure 9: The example considered

The first phase of the reverse engineering process aims to create the task model associated with each page. For example, we can consider the page for subscribing as a member (http://giove.cnuce.cnr.it/sigchi/membership.htm). This page contains a form with some input fields (Text and textarea), internal anchors, external links and internal links in the navigation bar common to all the pages of the site.

Figure 10 shows the sequence of rules applied during creation of the corresponding task model.

RULE 1 - Creation of initial task structure associated with the web page
RULE 2 - Creation of task structure associated with internal anchor
RULE 3 - Creation of task structure associated with internal link
RULE 3b - Creation of task structure associated with external link
RULE 9 - Creation of task structure associated with form
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area
RULE 7 - Creation of task structure associated with text area

Figure 10: Rules applied to create the task model of the subscription page.

At the beginning the initial structure is created according to Rule 1. Next, all the links in the page are analysed by applying rules 2 and 3. Figure 11 shows the task model created up to this point.

The application of the Rule 2 has determined the creation of two interactive tasks related to an internal link (#page-content) and a mailto element. The task *Select new page Membership-Sigchi* contains all the tasks related to external and internal links (in the figure it has not been expanded for reason of space).

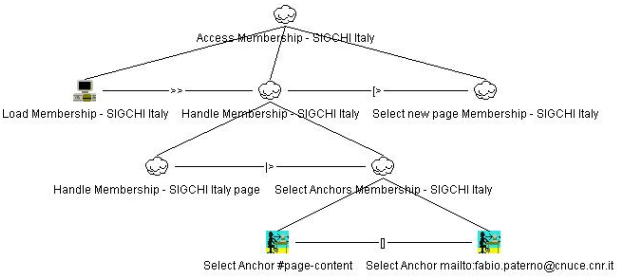


Figure 11: The initial task model for the example

In the next phase a depth-first visit of the DOM of the page is performed and the rules for the elements internal to the page are applied.

Thus, the various subtrees are created as subtasks of the task *Handle Membership – SIGCHI Italy*. This visit reveals the presence of a form that requires application of rule 9. While the analysis considers the part of the DOM associated with the form,

all the tasks obtained through the application of rule 7, associated with the input elements, are inserted as sub tasks of *Compile Form sigchi*. At the end of the DOM analysis the task model associated with the Form is that shown in Figure 12.

Once the process of creating the task model associated with each page is completed, we move on to consider the overall navigation in the site. Since the site has a navigation bar that allows the user to access any section from any page, we can apply the related rule.

The resulting model has a recursive structure that describes the possibility of accessing any page at any time. It contains 371 tasks structured into six levels with 256 basic elements. There are 193 abstract tasks, 168 interaction task, and 10 application tasks.

The automatic analysis can take place after a preparation phase during which the designer indicates the mappings between user interactions and the basic tasks in the model (the leaves of the task hierarchy). This association allows the tool to use the semantic information contained in the task model to analyse the traces of actions of the actual use.

More generally, WebRemUSINE provides three types of information regarding tasks, pages and session simulation.

Information regarding task performance concerns the performance time, which is indicated for both high-level and basic tasks through diagrams that also highlight the minimum, median, mean and maximum values (see for example Figure 14). In addition, page download times are highlighted in order to distinguish

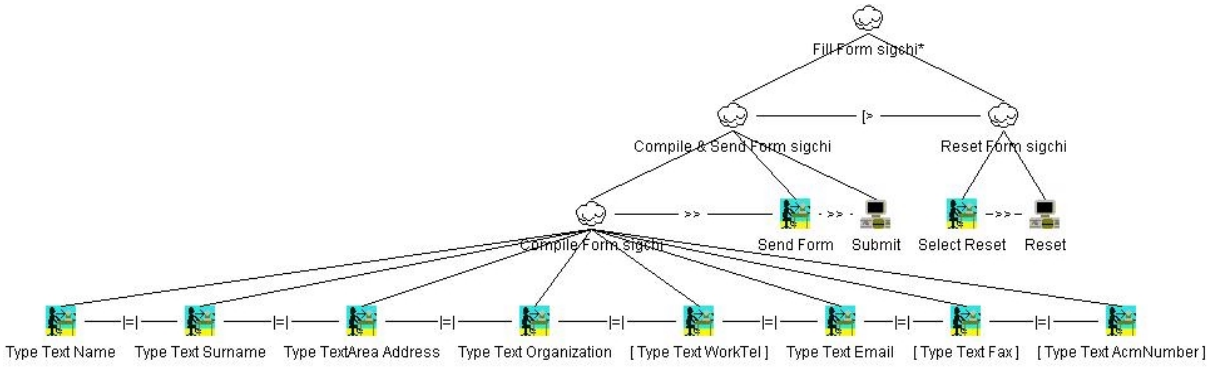


Figure 12: Evolution of the example.

USABILITY EVALUATION

Once we obtain the task model, it can be used in different ways: as a logical description of the design to analyse and discuss with all the stakeholders, or as an input element for usability evaluation. For the latter case, we have developed another tool, WebRemUSINE. Using as input the task model describing how the current design assumes that activities should be performed and logs of user sessions automatically created at the browser side, the tool provides various information that can be useful for usability evaluators (see Figure 13).

Thus, during a session all the user interactions are recorded through a Java script that is automatically included in all the pages composing the Web site. In addition, the list of tasks supported by the application is presented to the users so that they can explicitly indicate what the current target task is. This information is then used to analyse the user interactions, for example to detect errors, which are actions useless for accomplishing the current task.

This approach combines empirical testing with model-based evaluation. In addition, it supports remote evaluation because it does not require users and evaluators to be at the same place, at the same time, thus, opening up the possibility of analysing large number of sessions with users spread in many sites.

between the time spent actually visiting and downloading.

The information provided regarding Web page use is once again represented by the minimum, median, mean and maximum time values.

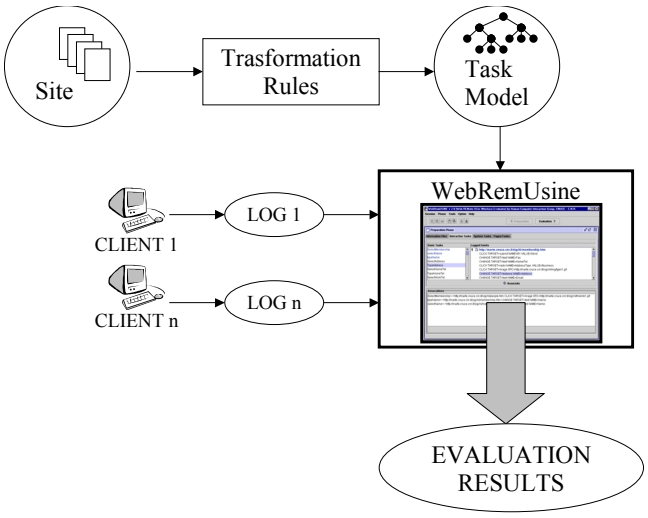


Figure 13: The overall environment developed

Regarding the user session, the tool is also able to take a user log session and simulate it with respect to the task model. This means that for each action in the log it identifies the corresponding basic task and looks at what happens in the model when that task is performed. If some precondition is violated, it highlights the error. It is also able to point out whether the task is useful to accomplish the target task and indicate what the enabled basic tasks are according to the temporal relationships indicated in the model. This type of analysis is useful to understand whether there is a mismatch between the actual use and the predicted use represented by the task model.

With this type of analysis it is possible to reveal whether users are able to achieve their goals, if there are parts that require longer than expected for users to understand, what types of errors they perform, frequently performed task sequences (or frequently accessed pages), never performed tasks or never accessed pages.

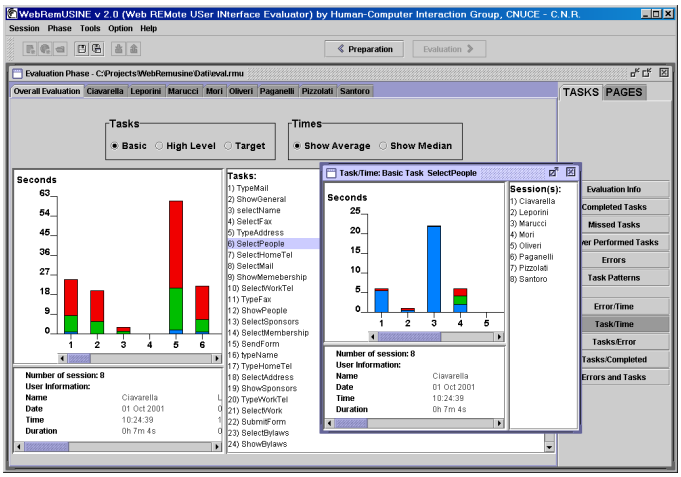


Figure 14: Example of results provided by WebRemUSINE.

CONCLUSIONS

We have presented a method and the related tool supporting the reconstruction of the task model of existing Web sites. This allows designers to better understand how current implementation assumes that tasks should be performed in order to reach user's goals.

We have also discussed how the resulting task model can be provided as input to WebRemUSINE a tool supporting automatic evaluation of Web sites based on the use of such models and automatically generated browser logs of user sessions.

The resulting overall environment aims to support remote usability evaluation of Web applications.

REFERENCES

[1] Bouillon, L., Vanderdonckt, J., and Souchon, N. Recovering Alternative Presentation Models of a Web Page with VAQUITA, Proceedings of CADUI'02, Valenciennes, pp.311-322, Kluwer, 2002.

[2] Card, S., Moran, T., Newell, A., *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum, Hillsdale, 1983.

[3] Chung-Man Tam R., Maulsby D., Puerta A., "U-TEL: A Tool for Eliciting User Task Models from Domain Expert", Proceedings ACM IUI'98, pp.77-80, ACM Press, 1998.

[4] Document Object Model (DOM) Level 1 Specification (W3C Recommendation) http://www.w3.org/TR/REC-DOM-Level-1/

[5] Hudson, S., John, B., Knudsen, K., Byrne, M., "A Tool for Creating Predictive Performance Models from User Interface Demonstrations", *Proceedings UIST'99*, pp.93-102, ACM Press, 1999.

[6] Paganelli, L., and Paternò, F. Intelligent Analysis of User Interactions with Web Applications, Proceedings ACM IUI'02, pp.111-118, ACM Press, 2002.

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

[8] Tidy http://www.w3. /People/Raggett/tidy/