Models, Tools and Transformations for Design and Evaluation of Interactive Applications

Fabio Paternò, Laila Paganelli, Carmen Santoro

CNUCE-C.N.R.
Via G.Moruzzi, 1
Pisa, Italy
fabio.paterno@cnuce.cnr.it

SUMMARY

This paper discusses a framework to address issues related to the design and evaluation of interactive systems in current technological settings characterised by wide variability of context of use. The basic idea is that effective solutions can be provided only if a meaningful set of models are identified along with transformations that allow moving from one model either to another model or to an implementation system and vice versa. In such a framework, the environment calls for the support of a set of tools able to ease model development and analysis and possible transformations.

KEYWORDS : Models, Transformations, Tools.

INTRODUCTION

The quickly increasing availability of a large spectrum of interaction devices that can be accessed from many types of environments risks creating chaos because designers of interactive applications have difficulties in managing all the relevant factors. Whenever there is some complexity to manage, humans tend to create models to highlight the important aspects to consider. In human-computer interaction a number of models have been used to support user interface designers and developers by highlighting relevant aspects that they should take into account and providing logical descriptions of such features. For example, task, user, domain, context, presentation, and dialogue models, as well as scenarios have widely been used. Such models can also be useful to address the new challenges, such as those raised by the increasing platform variability. However, we need at least two key elements to take advantage of them: automatic tools and transformations for moving from one model to another or to the system implementation and vice versa. The basic idea is that the design and evaluation process can be effective if it is supported by a flexible set of tool-supported transformations [5]. There are many potential transformations but we do not need all of them in all situations. The choice of which transformation to perform depends on the current goals, the material and time available, the characteristics of the application domain, and so on. For example, in some cases designers have to start from scratch and so they need to envision the main features of the new system, whereas in other cases there is an existing system for which the underlying abstractions must be identified in order to evaluate it and determine suggestions for improvements. The goal of this paper is to give an introduction to these problems and their possible solutions, and to provide a number of examples of transformations and tools supporting them, most of them developed at the HCI group of CNUCE-C.N.R.. In particular, the discussion is divided into three parts: an introduction to the logical framework proposed based on the use of models, transformations and tools and a discussion of two instances of this framework, one dedicated to supporting the design of nomadic application, the other to the remote usability evaluation of web sites.

Regarding models for interactive applications, we pay particular attention to task models, how to represent them (including representations for cooperative applications), analyse their content, and support the development of such models using information taken from informal descriptions, such as scenarios. We will also discuss how to use information provided by task models to derive user interfaces for heterogeneous interaction platforms.

Next, we will move on to discuss how inverse transformations can be useful for usability evaluation: starting with logs of user interactions with the actual system, it is possible to identify how users perform tasks and compare this with the system task model in order to check potential mismatches that can generate usability problems.

MODEL-BASED DESIGN OF NOMADIC APPLICATIONS

Particular attention is to be paid to the wide variety of devices currently available, which is bound to increase in the coming years, because it poses a number of issues for the design cycle of interactive software applications. Model-based approaches for the design of nomadic applications aim to enable each interaction device to support appropriate tasks that users expect to perform and help designers to develop the various device specific application modules in a consistent manner.

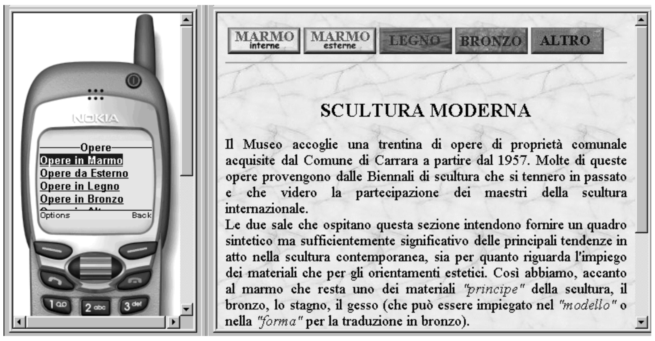
In a recent paper, discussing the future of user interface tools, Myers, Hudson, and Pausch [1] indicate that the wide platform variability *encourages a return to the study of some techniques for device-independent user interface specification, so that developers can describe the input and output needs of their applications, so that vendors can describe the input and output capabilities of their devices, and so that users can specify their preferences. Then, the system might choose appropriate interaction techniques taking all of these into account*. The basic idea is that instead of having separate applications for each device that exchange only basic data, there is some abstract description and then an environment that is able to suggest a design for a specific device that adapts to its features and possible contexts of use.

This problem is a novel challenge for model-based design and development of interactive applications. The potentialities of these approaches have been addressed in a limited manner. In the GUITARE Esprit project a user interface generator has been developed that takes ConcurTaskTrees (CTT) task models and produces user interfaces for ERP applications according to company guidelines. However, completely automatic generation is not a general solution because design is a complex process; many factors must be taken into account, and the weight of each factor can vary depending on many parameters (type of application, users, devices, …). Semi-automatic processes can be more general. It means that there are tools that help designers move from models to concrete user interfaces by choosing from several available criteria.

UIML is an appliance-independent XML user interface language. While the language proposed is ostensibly independent of the specific device and medium used for the presentation, it does not seem to take into account the research work carried out in the last decade on model-based approaches for user interfaces. For example, the language provides no notion of task, it mainly aims to define an abstract structure. The approach takes an interesting path, but the declarative language needs improvement. This confirms the need for a universal XML Human-Computer Interaction language able to represent all the aspects that should be considered when supporting the rendering of user interfaces in heterogeneous environments. The W3C consortium has recently delivered the first version of a new standard (XForms) that presents a description of the architecture, concepts, processing model, and terminology underlying the next generation Web forms. It is based on the separation of the purpose from the presentation of a form. Once again this shows the importance of separating conceptual design from concrete presentation, but it also highlights the need for meaningful models to support such approaches.

More generally, the issue of applying model-based techniques to the development of UIs for mobile computers has been addressed at a conceptual and research level, but there are still many issues that should be solved to identify systematic, general solutions that can be supported by automatic tools. Our approach provides designers with support for the design and development of nomadic applications: these applications manage users' access both to their own personal information as well as to the public domain. Such access should be ubiquitous and independent of specific devices. The aim is to find general solutions that can be tailored to specific cases, whereas current practise is to develop *ad hoc* solutions with few concepts that can be reused in different contexts.

The design of multi-platform applications has to take into account various aspects. It is possible to support the same type of tasks with different devices. In this case, what has to be changed is the set of interaction and presentation techniques to support information access while taking into account the resources available in the device considered.



SCULTURA MODERNA

Il Museo accoglie una trentina di opere di proprietà comunale acquisite dal Comune di Carrara a partire dal 1957. Molte di queste opere provengono dalle Biennali di scultura che si tennero in passato e che videro la partecipazione dei maestri della scultura internazionale.
Le due sale che ospitano questa sezione intendono fornire un quadro sintetico ma sufficientemente significativo delle principali tendenze in atto nella scultura contemporanea, sia per quanto riguarda l'impiego dei materiali che per gli orientamenti estetici. Così abbiamo, accanto al marmo che resta uno dei materiali *"principe"* della scultura, il bronzo, lo stagno, il gesso (che può essere impiegato nel *"modello"* o nella *"forma"* per la traduzione in bronzo).

Figure 1: Example of same task, different interface.

However, often interaction devices are suitable to support different sets of tasks. For example, phones are more likely to be used for quick access to limited information, whereas desktop systems better support browsing through large amounts of information. To complicate matters, it must be borne in mind that even within the same class of devices there are different presentation models that need to be handled. For example, more and more, cellular phones are being used to access remote applications, and currently access is provided by WAP phones. There are many usability issues that are limiting their spread. While in desktop systems we have mainly two well-known browsers with some compatibility issues (even though such issues often create some problems), in WAP-enabled phones a number of microbrowsers tend to accept slightly different versions of

WML, assume to interact with slightly different phones (for examples, phones with a different number of softkeys) and interpret the softkeys interactions differently.

More precisely our method [3] is composed of a number of steps (see Figure 2) that allow designers to start with an overall envisioned task model of a nomadic application and then to derive concrete and effective user interfaces for multiple devices:

High-level task modelling of a multi-context application. In this phase designers need to think about what logical activities have to be supported and identify the logical relationships among them. Here they develop a single model that addresses the various possible contexts of use and the various roles involved. This model should describe all the activities users may want to perform in the possible contexts. In the task model we consider also how changes in the users' environment can either trigger or stop some activities. Various techniques (such as scenarios or use cases) can be used to gather information relevant for the development of this model. Along with the task model, we also develop a domain model aiming to identify all the objects that have to be manipulated to perform tasks and the relationships among such objects.

Developing the system task model for the different platforms considered. Here designers have to filter the task model according to the target platform. It is possible to consider any type of target platform including wearable computers, augmented reality and so on. This involves creating task models in which the tasks that cannot be supported meaningfully in a given platform are removed. It also implies adding the navigational tasks deemed necessary to interact with the considered platform. For example, selecting one element from a list of options is something that should be supported differently if we consider a desktop system, a PDA or a Wap phone. Thus, we obtain the system task model for the platform considered. These task models are specified using the ConcurTaskTrees notation (CTT) [4] because it allows description of flexible behaviours and is tool supported (http://giove.cnuce.cnr.it/ctte.html). In this notation it is possible to graphically represent the hierarchical logical structure of the task model and 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.

From system task model to abstract user interface. Here the goal is to obtain an abstract description of the user interface that provides greater detail about the structure of the user interface. The abstract description is composed of a set of abstract presentations that are identified with the support of the enabled task sets. The main purpose of the enabled tasks sets is to help in identifying when the interaction techniques supporting the tasks should be enabled. Then, still with the help of the task model, we identify the possible transitions among the presentations considering the temporal relationships that the task model indicates. We indicate the structure of each presentation using operators such as grouping (a set of interaction techniques should be grouped somehow). Analysing task relationships can be useful for structuring the presentation. For example, the hierarchical structure of the task model can be considered to identify interaction techniques to be grouped, for example, those that have the same parent task and are thus logically more related to each other. Likewise, concurrent tasks that exchange information can be better supported by highly integrated interaction techniques (to some extent, merged), as happens when using adjacent techniques, so that users can better follow their mutual dependencies. At this point we also obtain a refined description of the basic interaction tasks that have to be supported, such as selection, control, editing, as well as their attributes, such as frequency, and related objects (including their cardinality). The classification of the basic task should be as refined as possible. So, for example, it can be better to indicate, not only that there is a selection task, but whether it should be a single or a multiple selection with low or high cardinality of choices.

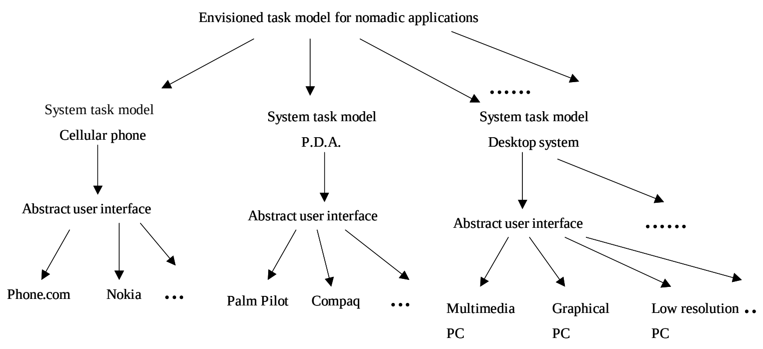


Figure 2: Approach proposed for design of nomadic applications.

User interface generation. In this phase we have the generation of the user interface. This phase is completely platform-dependent and must consider the specific properties of the target device. It is not sufficient to say, for example, that we want to consider a cellular phone; rather, we need to consider the type of microbrowser supported, the number and the types of softkeys available. If we consider PDAs, we need to know whether they support colour and audio output or not. Likewise, if we consider desktop systems, we need to know if they have good multimedia features and, in the case of a graphics-only desktop system, the display's resolution and size.

We have started development of tool support for this method. To date, we have defined XML versions of the language for task modelling (ConcurTaskTrees), enabled task sets (sets of tasks enabled over the same time) and the language for modelling abstract interfaces. We aim to tightly integrate this new tool for interface prototyping for multiple platforms with a previously developed tool for task modelling and analysis.

REMOTE USABILITY EVALUATION OF WEB SITES

In the previous section we have seen a chain of possible transformations able to derive user interfaces from task models of nomadic applications. Now, we consider how we can support usability evaluation of web sites. In particular, we will discuss what can be analysed with tool support by starting with logs of user interactions. To this end, we will need to transform the log events into task-related information.

Creating a Web site allows millions of potential users with various goals and knowledge levels to access the information that it contains. For this reason, interest in usability evaluation of Web sites is rapidly increasing. In general terms, many methods for usability evaluation have been proposed: they range from inspection-based methods based on the ability of the evaluators (such as heuristic evaluation or cognitive walkthrough) to methods based on users involvement (such as usability testing or cooperative evaluation) or to others that use modelling techniques for example to predict time performance (such as GOMS-based methods).
There are many motivations for automatic tools able to support evaluation process. The total or partial automation of usability evaluation can reduce the time required and costs and release evaluators from repetitive and tedious tasks. A number of tools for usability evaluation of traditional graphical applications have been proposed, however, the different nature of Web interfaces requires specific tools. Here we discuss a method and a relative tool (WebRemUSINE [2]) to detect usability problems in Web interfaces through a remote evaluation. Our approach combines two techniques that usually are applied separately: empirical testing and model-based evaluation. The reason for this integration is that models can be useful to detect usability problems but their use can be much more effective if they can be related to the actual use of a system.
In empirical testing the actual user behaviour is analysed during a work session. This type of evaluation requires the evaluator to observe and record user actions in order to perform usability evaluation. Manual recording of user interactions requires a lot of effort thus automatic tools have been considered for this purpose. Some tools support video registration but also video analysis requires time and effort (usually it takes five times the duration of the session recorded) and some aspects in the user interaction can still be missed by the evaluator. In model-based evaluation, evaluators apply user or task models to predict interaction performance and identify possible critical aspects. For example GOMS (Goals, Operators, Methods and selection rules) has been used to describe an ideal error-free behaviour. Model-based approaches have proven to be useful but the lack of consideration for actual user behaviour can generate results that can be contradicted by the real user behaviour. It becomes important to identify a method that allows evaluators to apply models in evaluation still considering information empirically derived. To this end the main goals of this environment 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.

Our tool is able to analyse the possible inconsistency between the actual user interactions and the task model of the Web site that describes how its concrete design assumes that activities should be performed. To support remote evaluation, we have developed a technique that allows recording user actions during a site visit. The analysis of the logged data is based on the comparison of the traces of actions performed with the temporal constraints described in the task model. This analysis provides evaluators with a number of results that are related to the tasks that users intend to perform and the Web pages and their mutual relationships.
The starting point was RemUSINE, an automatic tool based on the use of task models to support evaluations of graphical applications. This tool was not suitable for web applications whose specific aims are to support tasks related to retrieving and accessing information, and navigation is based on links to remote pages. In RemUSINE to identify errors (useless actions for the current task), the possible enabling and disabling of user interface actions was considered. Then, if users try to perform an action, this means that they want to perform the associated task, and if the action is disabled, then an error is performed. For example, suppose the user has to perform some actions and then save the data. If the user tries to

save the data before terminating the sequence of actions planned, then this action would be disabled, and the error can be automatically detected. During Web site evaluation it is not possible to apply this concept because usually links are always enabled. Thus, in this context it is difficult to automatically identify user intentions. The solution that we have adopted to capture this information is to display the high-levels tasks that are supported by the Web site asking the user to indicate explicitly what task they want to perform (See Figure 3). During the testing, since we perform remote evaluation without direct observation of the user interactions, it is important to obtain logs with detailed information. We have designed and implemented a logging tool able to record a set of actions wider than those contained in server logs. WebRemUSINE compares the logs with the task model and provides results regarding both the tasks and the Web pages supporting an analysis from both viewpoints.

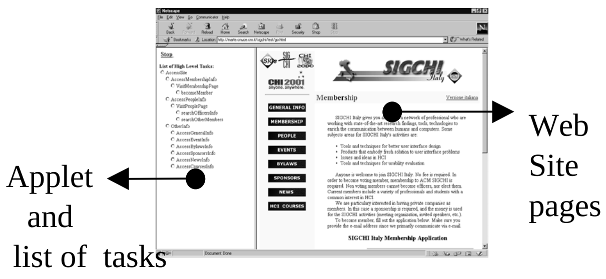


Figure 3: The interface during the test session.

The method is composed of three phases:

- *Preparation*, it consists in creating the task model of the Web site, collecting the logged data and defining the association between logged actions and basic tasks;

- *Automatic analysis*, where WebRemUSINEs examines the logged data with the support of the task model and provides a number of results concerning the performed tasks, errors, loading time, …

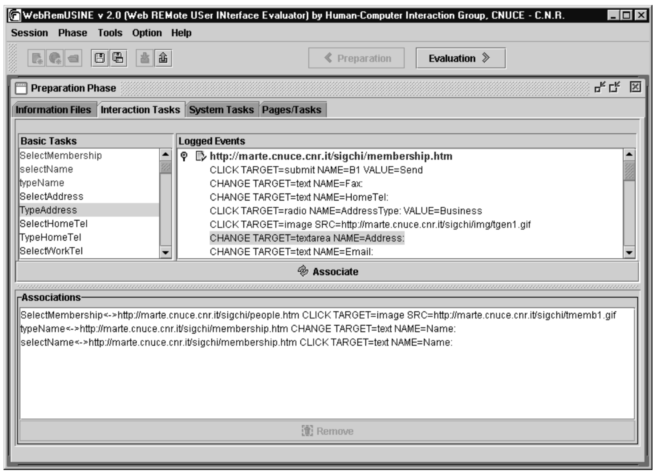
- *Evaluation*, the information generated is analysed by the evaluators to identify usability problems and possible improvements in the interface design.

The environment is mainly composed of three modules: the ConcurTaskTrees editor 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.

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

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

The preparation module, this module filters the information recorded during the testing, then the evaluator has to associate each basic task with the corresponding event (see Figure 4). All the event-basic task associations are recorded in a file.



Basic Tasks

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

Logged Events

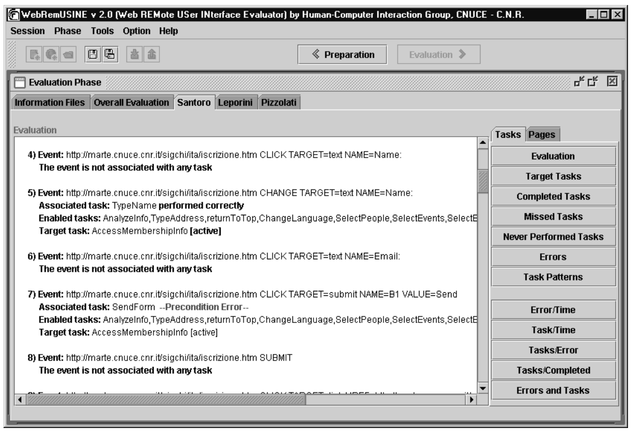
```
http://marte.cnuce.cnr.it/sigchi/membership.htm
    CLICK TARGET=submit NAME=B1 VALUE=Send
    CHANGE TARGET=text NAME=Fax:
    CHANGE TARGET=text NAME=HomeTel:
    CLICK TARGET=radio NAME=AddressType: VALUE=Business
    CLICK TARGET=image SRC=http://marte.cnuce.cnr.it/sigchi/img/tgen1.gif
    CHANGE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

Associations

```
SelectMembership<->http://marte.cnuce.cnr.it/sigchi/people.htm CLICK TARGET=image SRC=http://marte.cnuce.cnr.it/sigchi/tmemb1.gif
typeName<->http://marte.cnuce.cnr.it/sigchi/membership.htm CHANGE TARGET=text NAME=Name:
selectName<->http://marte.cnuce.cnr.it/sigchi/membership.htm CLICK TARGET=text NAME=Name:
```

Figure 4: The part of the tool supporting the preparation phase.

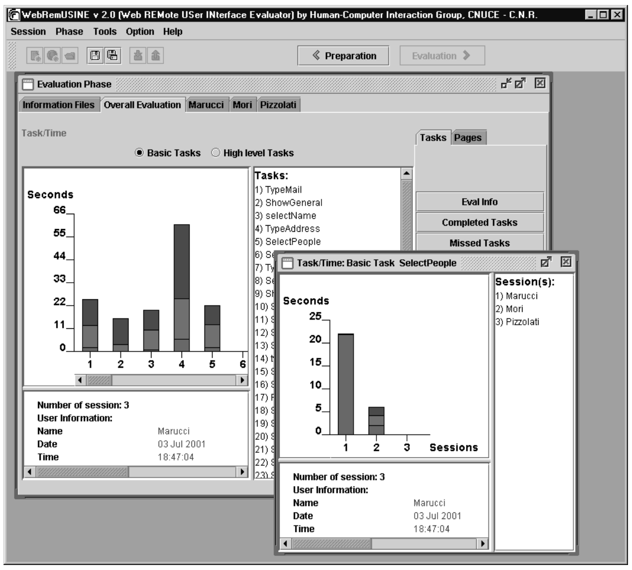
The evaluation module, it has three inputs: the task model, the log files and the event-basic tasks associations. This information is useful to analyse the logs with the support of the task model and identify errors performed by the user during the navigation. By following the sequence of events stored in the log it is possible to identify the corresponding tasks (through the event-basic tasks association) and comparing the sequence with the temporal relationships among the tasks it is possible to identify the tasks performed correctly and those that generate errors (see Figure 5). It is also possible to calculate the completion time for the relative tasks (see Figure 6). All results are displayed by WebRemUSINE in various formats both textual and graphical.



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

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

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

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

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

Figure 5: Automatic analysis of a user session.

The WebRemUSINE analysis can point out usability problems such as tasks with long performance or tasks not performed according the task model corresponding to the Web site design. These elements are useful to identify the pages that create problems to the user. As previously explained, log files store both user interactions (mouse movements, keyboard input, link selection) and browser behaviour (start and end of page loading). The events corresponding to user interactions are associated with interaction tasks whereas the internal browser events are associated with system tasks.



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

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

Number of session: 3
User Information:

Name	Marucci
Date	03 Jul 2001
Time	18:47:04

Figure 6: Example of display of task performance analysis.

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.

CONCLUSIONS

There are many possible transformations useful to support design and evaluation of interactive applications. Here we have introduced the current potentials of such an approach and discuss examples of tools and methods addressing some of the open issues.

There is a need for widely accepted modelling languages to exploit the potentialities of this approach. In this way, we could obtain a wide range of tools enabling easy development of models, providing meaningful metrics for the analysis of such models and supporting transformations in other environments. Then, designers could choose the set of tools most suitable to their specific goals.

ACKNOWLEDGEMENTS

Support from the European IST Project CAMELEON (http://giove.cnuce.cnr.it/cameleon.html) is gratefully acknowledged.

BIBLIOGRAPHY

1. B. Myers, S. Hudson, R. Pausch. *Past, Present, Future of User Interface Tools*. Transactions on Computer-Human Interaction, ACM, 7(1), March 2000, pp. 3-28.
2. Paganelli L. and Paternò F., *Remote Analysis of User Sessions for Usability Evaluation of web Sites*, CNUCE-C.N.R. Internal Report 2001-013, September 2001.
3. Paternò F. and Santoro C. *One Model, Many Interfaces.* CNUCE-C.N.R. Internal Report, July 2001.
4. Paternò F., *Model-Based design and Evaluation of Interactive applications*, Springer Verlag, ISBN 1-85233-155-0, 1999.
5. Weicha C., Szekely P., *Transforming the UI for anyone, anywhere*, Proceedings CHI 2001 Extended Abstracts, pp.483-484.