Models for Universal Usability

Fabio Paternò
ISTI-C.N.R.
Via G.Moruzzi 1
56124, Pisa, Italy
fabio.paterno@isti.cnr.it

ABSTRACT
This paper discusses how model-based approaches can support designers and developers to address a number of challenges raised by universal usability, such as the possibility of obtaining user interfaces able to adapt to any device and usability evaluation tools able to analyse any users who can be located anywhere. The paper provides an overview concerning results that can be obtained in this area with links to activities carried out in current projects. Lastly, a research agenda for the field is proposed and discussed.

KEYWORDS : Heterogeneous clients. Multi-platform User Interfaces. Authoring environments. Abstract User Interfaces. Remote evaluation. Task Models.

INTRODUCTION

Recent years have seen the ever-increasing introduction of new types of interactive devices. Everyday life is becoming a multi-platform environment where people are surrounded by devices through which they can access interactive applications in different ways. They range from small devices such as interactive watches and smart phones to very large flat displays. A platform is a class of devices that share similar characteristics, especially in terms of interaction resources. Examples of platforms are the desktop, the PDA, the mobile phone. In addition, new improvements in communication technology have opened up many possibilities. WLAN, Bluetooth, Infrareds, RFIDs support different ways to communicate information, identify users position, and enable new functionality.
All this changes the nature of many interactive systems, converting them to nomadic applications, which are applications that support user access in various contexts through different interactive devices. Their nature changes because in order to be usable, they have to adapt their user interfaces to the different contexts of use, in particular to the different devices used to access their functionality. Consequently, one fundamental issue is how to assist software designers and developers in managing such complex aspects when developing nomadic applications.

Models can help in the design and evaluation of interactive applications even if often model-based approaches have sometimes been criticised because some researchers have considered them too theoretical, a kind of useless complication. However, if we think of what we do in practice we can discover that we often use models. As soon as there is some complex entity to manage, people try to identify the main aspects that should be taken into account and their relations. So, we build models to understand reality and to guide our interactions with it. In this case the complexity is given by the variability in the types of interaction platforms, users and environments. Even researchers in user interface tools who have often been sceptical regarding model-based approaches admit [8] that they can be useful for developers, vendors and users in order to reason about user interface design solutions.

BASIC CONCEPTS

As often happens, the solution to a complex problem can be based on a small set of clear basic concepts. In order to address such issues it is important to consider the various viewpoints that it is possible to have on an interactive system. Such viewpoints differ for the abstraction levels (to what extent the details are considered) and the focus (whether the task or the user interface is considered). The model-based community has long discussed such possible viewpoints and in the CAMELEON project there is a study to better structure them and understand their relations through a reference framework [5] [6]. Such abstraction levels are:

- *Task and object model*, at this level, the logical activities that need to be performed in order to reach the users' goals are considered. Often they are represented hierarchically along with indications of the temporal relations among them and their associated attributes. The objects that have to be manipulated in order to perform tasks can be identified as well.
- *Abstract user interface*, in this case the focus shifts to the interaction objects supporting task performance. Only the main aspects are considered, thereby avoiding low-level details. An abstract user interface is defined in terms of presentations, identifying the set of user interface elements perceivable at the same time, and each presentation is composed of a number of interac-

tors [16], which are abstract interaction objects identified in terms of their semantics effects.

- *Concrete user interface*, at this point each abstract interactor is replaced with a concrete interaction object that depends on the type of platform and media available and has a number of attributes that define more concretely its appearance and behaviour.
- *Final User interface*, at this level the concrete interface is translated into an interface defined by a specific software environment (e.g. XHTML, Java, …).

To better understand such abstraction levels we can consider an example of a task: making a flight reservation. This task can be decomposed into selecting departure and arrival towns and other subtasks. At the abstract user interface level we need to identify the interaction objects needed to support such tasks. For example, for specifying departure and arrival towns we need selection interaction objects. When we move on to the concrete user interface, we need to consider the specific interaction objects supported. So, in a desktop interface, selection can be supported by a list object. This choice is more effective than others because the list supports a single selection from a potentially long list of elements. The final user interface is the result of these choices and others involving attributes such as the type and size of the font, the colours, and decoration images.

Many transformations are possible among these four levels for each interaction platform considered: from higher level descriptions to more concrete ones or vice versa or between the same level of abstraction but for different type of platforms or even any combination of them. Consequently, a wide variety of situations can be addressed. More generally, the possibility of linking aspects related to user interface elements to more semantic aspects opens up the possibility of intelligent tools that can help in the design, evaluation and run-time execution.

XML-based languages can be useful for representing the relevant concepts and ease their automatic manipulation. Examples of projects that have addressed these issues are: *The User Interface Markup Language (UIML)* (http://www.uiml.org/) [1] is an XML-compliant language that allows a declarative description of a user interface in a device-independent manner. This has been developed mainly by Harmonia and Virginia Tech. However, their tools do not support the task level. *The eXtensible Interface Markup Language (XIML)* (http://www.ximl.org/) [18] is a XML specification language for multiple models. This has been developed by a forum headed by RedWhale software. Tool support is not currently publicly available.

UNDERSTANDING TASK-DEPENDENT TASKS

Depending on the type of the platform different tasks can be meaningful. For example, in a flight application it can be possible to use a desktop system for comparing prices of flights and making reservation whereas a mobile device can be useful to check status of a particular flight. Likewise, in a cinema application the resources of a desktop system can be used to read a movie review and watch the associated trailer whereas a mobile system can be used to buy the cinema ticket and avoid the line while moving in the town.

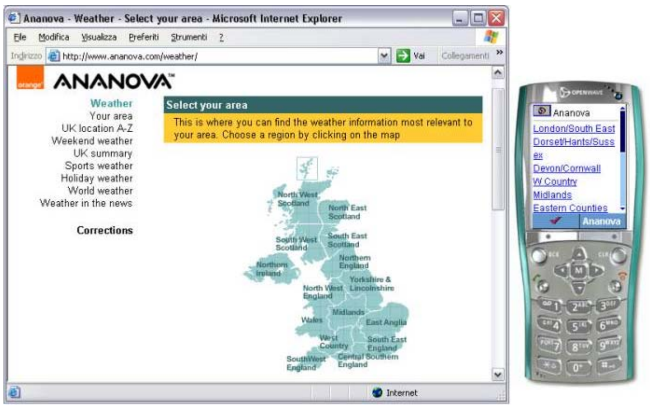
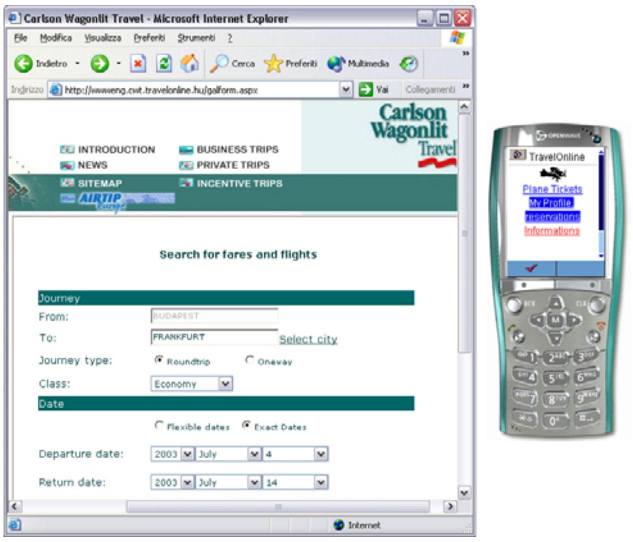


Figure 1: Example of same task/different performance.

More generally if we analyse tasks and platforms we can find that various relations can exist [14]:

- There are tasks that are meaningful only when certain platforms are considered as we mentioned above.
- There are tasks that can be performed in various platforms but in different ways: using different interaction objects, considering different domain objects, or having different associated subtasks.
- There are tasks performed on one platform that enable or disable tasks that can be performed through another platform. For example, making a flight reservation through the desktop system can enable the possibility of accessing updated information on that flight through the mobile phone (see Figure 2).

These three main task/platforms relationships (same tasks/different performance, platform-dependent tasks, dependencies among platform-dependent tasks) along with the various possible views on an interactive systems can be used to analyse current approaches to design of multi-device interfaces, in particular their coverage and potential limitations.


Carlson Wagonlit Travel - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo http://wwweng.cwt.travelonline.hu/galform.aspx Vai Collegamenti
Carlson Wagonlit Travel
INTRODUCTION
NEWS
SITEMAP
AIRTIP
BUSINESS TRIPS
PRIVATE TRIPS
INCENTIVE TRIPS
Internet
TravelOnline
Plane Tickets
My Profile
reservations
Informations

Search for fares and flights

Journey
From: BUDAPEST
To: FRANKFURT Select city
Journey type: [✓ Roundtrip] Oneway
Class: Economy

Date
Flexible dates [✓ Exact Dates]
Departure date: 2003 July 4
Return date: 2003 July 14

Figure 2: Example of dependency among task-dependent platforms.

POTENTIAL APPROACHES TO THE DESIGN OF MULTI-DEVICE INTERFACES

In current practise the design of multi-platform applications is still often obtained through the development of several versions of the same applications, one for each platform considered. Then, such versions can at most exchange data. This solution with no tool support to address multi-platform issues is rather limited because it implies high implementation and maintenance costs. The opposite solution, completely automatic, is to use transcoding where an application written in a language for a platform is automatically transformed into an application in a language for another platform. An example is HTML to WML transcoding. However, this type of solution often provides bad results in terms of usability because it assumes that the same tasks are supported by each platform and tends to support them in the same manner without taking into account the specific features of the platform at hand. This solution has also been applied to transcoding from one modality to another one, such as from HTML to VoiceXML [17]. However, such transcoding between different modalities raises even more issues. For example, how to translate a graphical interface in a vocal interface when various images with informative content are provided. It is said that "a picture is worth a thousand words" but people do not like to hear so many words. For example, this problem occurs when a map is provided in order to indicate how to reach a point. This cannot be easily supported by a vocal interface especially if it is generated automatically. Another solution proposed is the use of style sheets. Each platform is associated with a different set of stylesheets. Thus, the same elements are presented differently according to the type of platform available. This can be useful to cover most of the cases belonging to the same task/different performance class. Indeed, style sheets can help only try to better support the same tasks through different platforms, but unfortunately this is not always adequate because users often want to carry out different tasks according to the various types of platform, not to mention the fact that even mutual relations among tasks performed through several platforms may exist.

A more comprehensive solution can be obtained through the use of model-based approaches: different levels of abstractions can capture relevant information without having to deal with a plethora of details related to each device and associating semantic aspects to low-level interface elements. This also enables automatic generation of the specific version adapted for each device, thus representing a viable alternative to overcome the limitations of other approaches.

I have sometimes heard the criticism that model-based approaches are too academic and cannot be actually applied in practise. The best demonstration that this sceptical view is wrong is in the recent W3C standards, such as XForms [20], which have introduced the use of abstractions to address new heterogeneous environments. In particular, XForms aims to separate presentation from content through the definition of a set of platform-independent controls suitable for general-purpose use. So, to some extent it applies concepts that have long been discussed in the research area concerning model-based design of human-computer interaction. The types of abstractions supported by XForms are at the level of an abstract user interface. The task level is not explicitly addressed. Once XForms is actually supported by the major browsers it will be interesting to support transformations from other HCI models to those supported by this mark-up language.

AN EXAMPLE OF APPROACH TO SUPPORT PLATFORM-DEPENDENT TASKS: TERESA

The TERESA tool [9] (publicly available at http:giove.cnuce.cnr.it/teresa.html) is able to support top-down development of a wide range of user interfaces for various platforms (desktop, PDA, mobile phones and vocal interfaces) and maintain links among the various abstraction levels considered in order to allow designers to better analyse them.

TERESA is a *mixed initiative* tool because supports different levels of automation ranging from completely automatic solutions to highly interactive solutions where designers can tailor or even radically change the solutions proposed by the tool. The tool aims to satisfy a variety of needs: situations when the time available is short, the application domain is rather narrow, or the designer has no expertise call for completely automatic solutions. When designers are expert or the application domain is either broad or has specific aspects, then more interactive environments are useful because they

allow the designer to directly make important design decisions.
It supports *a top-down, model-based* approach. So, designers first have to create more logical descriptions, and then move on to more concrete representations until they reach the final system. We are aware that in other cases reverse engineering approaches may complement the approach pursued through TERESA.

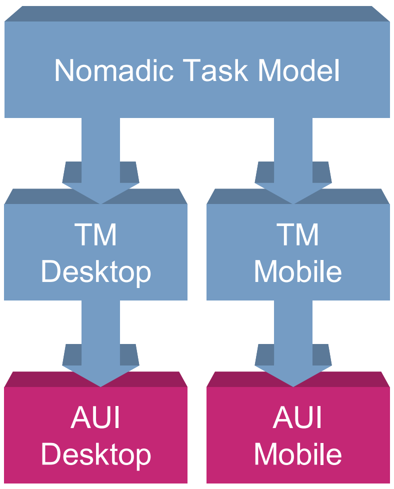


Figure 3: Example of design process for a nomadic application.

In order to support platform-dependent tasks, we start with *high-level task modelling of a multi-platform application*. The purpose of this model is to provide an overview of the tasks supported by the nomadic application. For each task it is possible to indicate what platforms are able to support it and it is also possible to show dependencies among tasks that can be performed through different platforms. The next step derives the system task model for the different platforms considered. Figure 3 shows an example of the resulting process in the case of a nomadic application that can be accessed through the desktop and the mobile platform.
Our approach aims to be comprehensive and to support the various possibilities indicated by our task/platform taxonomy, although it may happen that only a part of it needs to be actually supported (for example, when only different brands of mobile phone are considered). In this case, there is no need for a nomadic task model, given that only one type of platform is involved and designers can start with either the corresponding system task model or the corresponding abstract user interface.
Then designers have to move *from the system task model to the abstract user interface*. The goal of this phase is to obtain an abstract description of the user interface composed of a set of abstract presentations that are identified through an analysis of the task relations. The presentation part is specified by means of abstract interaction objects composed through various operators (grouping, ordering, hierarchy, relation), which stand for different composition techniques (for example, the grouping operator will highlight the fact that there are objects which should be grouped together because they are closely related to each other). The next step is *from abstract to concrete interfaces*. This phase is completely platform-dependent and has to consider the specific properties of the target platform. Then, every interactor is mapped into interaction techniques supported by the particular target platform, and the abstract operators also have to be appropriately implemented by highlighting their logical meaning

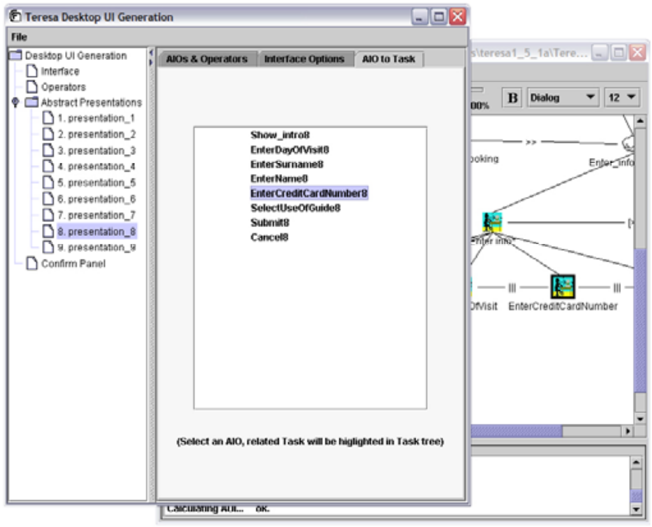


Figure 4: How the tool shows associations between interaction objects and tasks.

The last phase is *Code generation,* this can be done completely automatically because all the design choices have already been made.

One important support provided by TERESA is that the relationships among the various abstraction levels are stored and can be highlighted for the designers on request (see Figure 4). This is very useful, for example when people are moving from abstract to concrete interfaces in order to understand more quickly what tasks and contexts the interaction objects under consideration refer to. This information is also useful in order to support the choices in terms of concrete user interface design.

REVERSE ENGINEERING FOR RECONSTRUCTING DESIGN DECISIONS

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.
We have seen in the previous section how it is possible to design user interfaces through 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, in some cases there is also a need 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.

A tool which can be considered a kind of reverse engineering approach is Critique [7], which aims to support the development of KLM models 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 relations among tasks (apart from sequentiality).

Vaquita [3] addresses Web applications and aims at identifying presentation models for single pages, which mainly means the abstract description of the interaction techniques used in the implementation. Recent developments of this tool [4] add the possibility of retargeting, which means translating the concrete interaction objects in the source user interface into abstract interaction objects for another target platform. In particular, in this way it supports transformation from HTML to mobile phone user interfaces.

Another approach, WebRevenge [12], supports reconstruction of task models of existing Web applications (implemented using HTML) and is able to describe the temporal and semantic relations among tasks.

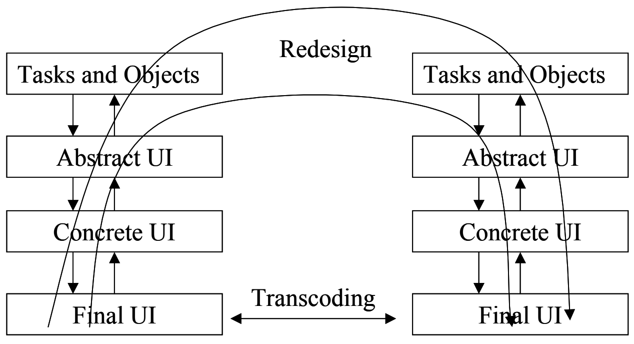


Figure 5: Approaches to changing the design for a different platform

As Figure 5 shows various approaches can be considered when designers want to start from an existing system and derive the user interface for another platform. Transcoding mainly operates on the syntactical level and it is unable to change the design choices. A real redesign can be obtained only through the use of reverse engineering techniques. The more the reverse engineering techniques are able to go up in terms of corresponding abstraction levels, the more substantial the design choices that can be performed taking into account the characteristics of the target.

RUN-TIME TRANSFORMATION

Improvements in digital connectivity are providing an ever increasing number of technologies that need to be exploited for various purposes: wireless connections between devices, mobile presentations, universal remote commanders, extensible mobile phones, … In Xweb [11] Olsen and others proposed a model for nomadic interaction based on the concepts of Join (mechanism to acquire access to a service's interface) and Capture (means for users to assemble interactive resources for a particular problem). They support the dynamic composition of devices (including Java rings and computer vision-based interaction techniques) also through an extension of the HTTP protocol.

There are some works that aim to support some kind of universal controller able to receive the descriptions of any nearby devices and generate the corresponding user interface. An example is given in [10] where the specification of each appliance includes a high-level description of each function, a hierarchical grouping of the functions, and dependency information, which relates the availability of each function to the appliance's state.

One important result that can be achieved through run-time transformations is user interface migration where the user interface is transferred from one device to another one at run-time, still maintaining the current state. This can be very useful for users who can freely move and dynamically exploit the resources of new devices that they encounter during their movements. The migration takes into account the runtime state of the interactive application and the different features of the devices involved. Different types of runtime migration can be identified, along with different levels of complexity for each one of them:

- *Total Migration*: the client application migrates totally from a device to the other.
- *Control Migration*: the client application is divided into two parts, one for user interaction (control part) and one for information presentation (presentation part). The control part remains on one device, while the presentation one migrates to the other device, or vice versa (an example is discussed in [10]).

- *Mixed Migration*: the client application is split into several parts, concerning both control and presentation and different parts are distributed over two or more devices.

An example of possible solutions to migration is introduced in [2] where we consider applications developed through a model-based approach. The runtime migration engine exploits information regarding the application's runtime state and higher-level information on the platform-dependent application versions. Runtime application data are used to achieve interaction continuity, while semantic information is considered to adapt the application's appearance and behaviour to the specific device. Since the number of presentations and the tasks supported by the various platforms may be different, it is not possible to create a one-to-one correspondence between presentations for different platforms. One important issue is how to identify the presentation for the target platform corresponding to the one active on the platform requesting migration, while maintaining the state of its interaction objects. The page to be visualized on the target device is identified using the following process: from the set of tasks to support, the tool identifies the most similar abstract presentation and then the corresponding page in the application version for the target platform.

Similarity is calculated in terms of supported tasks, the more similar the tasks associated to the two presentations (source and target) are, the more similar the presentations will be. Presentation similarity is the basic criterion to be considered, but under particular conditions it may not be enough. When the migrating presentation supports a task set that is associated with multiple presentations in the target version, each of which supports the same number of tasks, then the similarity will be the same for each potential target presentation. Thus, a further criterion is used to decide which target presentation to activate. To this end, we identify the target presentation supporting the task associated with the interaction object last modified by the user, since the user is most likely to continue interaction from that point. Once the target presentation has been identified, it is necessary to calculate the state of the objects contained in the corresponding page, which will be communicated to the target device along with its URL. For this purpose, data referring to the runtime state of the application will be associated to the corresponding tasks and adapted to the object implementation for the target device. This allows obtaining, for example, that if an element in a radio-button was selected and the same choice is supported through a list in a desktop system then the corresponding element in the list will be automatically selected.

REMOTE EVALUATION

Many methods for usability evaluation have been proposed so far. Automatic tools can be helpful in evaluation for many reasons: increase consistency in the identification of the problematic parts, reduce the cost of usability evaluation, widen the coverage of the evaluated features, … In particular, remote evaluation (evaluation which occurs when users and evaluators are distant in time and/or space) is a promising approach because it allows reduction of the evaluation costs and analysis of the users in their daily environment where they behave more realistically.

WebRemUsine [13] 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 and some aspects of 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. Model-based approaches can be useful but the lack of consideration for actual user behaviour can generate results that do not agree with the real user behaviour.

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 6). 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 analysi*s, where WebRemUSINE examines the logged data with the support of the task model and provides a number of results concerning the performed tasks, errors, loading time. WebRemUSINE displays all results in various formats both textual and graphical.
- *Evaluatio*n, 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); the browser logging tool to record user interactions; WebRemUSINE, a tool able to perform an analysis of the files generated by the logging tool using the task model created with the CTTE tool.

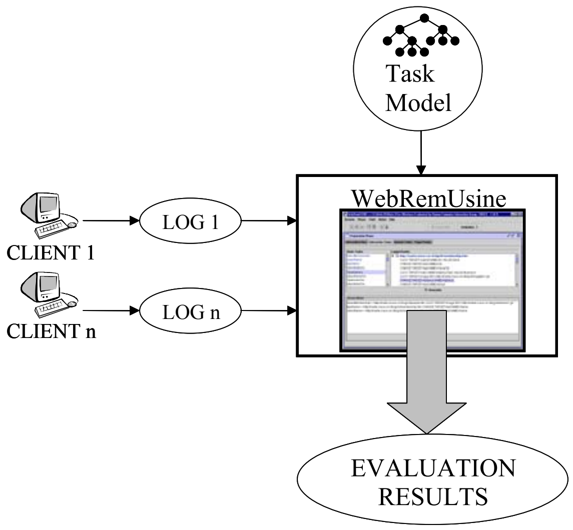


Figure 6: The Overall Architecture of WebRemUSINE.

This approach allows the evaluators 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 and to obtain a set of quantitative measures (such as execution task time or page downloading time), for individuals and group of users, useful for identifying usability problems.

RESEARCH AGENDA

This discussion has shown that we need tools able to incorporate a lot of design knowledge in order to support designers and evaluators. The goal is not to provide tools for a completely automatic design or evaluation. Rather, a more meaningful approach is to provide designers with a number of pieces of information that can help them in their work, support their creativity and find more consistent solutions. There is a need for a trade-off between interactivity and automation and this trade-off depends on many factors (time, application domain, designer's experience).

While models have been used extensively at design time, it is probably at run-time that they will prove to be particularly useful in supporting changes in user-interface presentation, dialogue, and content according to the context, while still supporting users effectively. To this end, integration at run-time of forward and reverse engineering techniques can provide interesting results. Run-time environments generate a number of interesting issues because many changes in the context of use can occur and it is difficult to foresee all of them at design time. So, it is important to provide solutions able to handle these situations.

However, a number of issues are still open also for design environments based on model-based approaches. One issue is how to make them suitable for a large number of people, including those with no experience in programming. Model-based approaches can be useful for end-user development because they allow people to focus on the main concepts (the abstractions) without being confused by many low-level details. Thus, they can be useful to achieve natural development, which implies that people should be able to work through familiar and immediately understandable representations that allow them to easily express and manipulate relevant concepts, and thereby create or modify interactive software artefacts. To this end, it becomes important to improve techniques for editing and analysing the relevant models, for example using vocal interaction with natural language-to-model specification translation, skecth-based input, and tangible interfaces.

In addition, their resulting user interfaces should not be only mono-modality interfaces: usually they just generate graphical or vocal interfaces, whereas the generated interfaces should be able to support the combined use of different modalities at the same time. This will soon become important also because the Web in the near future will also be able to support multi-modality.

Another important goal is to obtain general solutions to the development of user interfaces able to adapt to the change in context. Sensing context can generate a lot of raw data and in order to manage such data it is important to identify meaningful abstractions, but it is also important to define and implement toolkits able to support adaptive rules that can be reused across multiple context-dependent applications.

Regarding evaluation, we can say that it will become context-dependent as well. So, it will be important to identify methods and tools able to analyse user behaviour and task performance in the different contexts of use and to identify relevant metrics.

CONCLUSIONS

Despite some scepticism in the HCI research community, model-based approaches have shown their utility to address some challenges raised by universal usability. After recalling some basic concepts, I have discussed some examples of how they can be exploited at design, run and evaluation time.

However, a lot of work still needs to be done to completely exploit their potential, thus a research agenda for discussion on future work in the area is proposed.

ACKNOWLEDGMENT

This work has been partially supported by the EU IST CAMELEON Project (http://giove.cnuce.cnr.it/ cameleon.html). I wish to thank the project's colleagues and the members of the HCI group at ISTI-CNR for useful discussions on the topics addressed in this paper.

REFERENCES

1. Abrams, M., Phanouriou, C., Batongbacal, A., Williams, S., Shuster, J. UIML: An Appliance-Independent XML User Interface Language, Proceedings of the 8th WWW conference, 1999.

2. Bandelloni R. , Paternò F., Platform Awareness in Dynamic Web User Interfaces Migration, Proceedings Mobile HCI 2003, LNCS N.2795, pp.440-445, Springer Verlag, 2003.

3. 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.

4. Bouillon, L. Vanderdonckt, J., “Retargeting Web Pages to other Computing Platforms”, Proceedings of IEEE Conference on Reverse Engineering WCRE'2002, IEEE Computer Society Press, Los Alamitos, 2002, pp. 339-348.

5. Calvary, G., Coutaz, J., Thevenin, D., “A Unifying Reference Framework for the Development of Plastic User Interfaces”, Proceedings Engineering Human-Computer Interaction, pp.173-192, 2001.

6. Calvary, G. Coutaz, J. Thevenin, D. Limbourg, Q. Bouillon, L. Vanderdonckt, J., “A unifying reference framework for multi-target user interfaces”, Interacting with Computers Vol. 15/3, Pages 289-308, Elsevier.

7. 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.

8. Myers, B., Hudson, S., Pausch, R. *Past, Present, Future of User Interface Tools*. Transactions on Computer-Human Interaction, ACM, 7(1), March 2000, pp. 3-28.

9. Mori, G., Paternò, F., Santoro, C., “Tool Support for Designing Nomadic Applications”, Proceedings ACM IUI’03, Miami, pp.141-148, ACM Press. http://giove.cnuce.cnr.it/teresa/pdf/iui03.pdf

10. Nichols, J. Myers B. A., Higgins M., Hughes J., Harris T. K., Rosenfeld R., Pignol M.. “Generating remote control interfaces for complex appliances”. Proceedings ACM UIST’02, pp.161-170.

11. Olsen D., Nielsen S.T., Parslow D., "Join and Capture: A Model for Nomadic Interaction", Proceedings ACM UIST’01, 2001, pp. 131-140.

12. Paganelli, L., Paternò, F. “A Tool for Creating Design Models from Web Site Code”, International Journal of Software Engineering and Knowledge Engineering, World Scientific Publishing 13(2), pp. 169-189 (2003).

13. Paganelli, L., Paternò, F., “Tools for Remote Usability Evaluation of Web Applications through Browser Logs and Task Models”, Behavior Research Methods, Instruments, and Computers, 2003, 35 (3), 369-378, August 2003.

14. Paternò, F. Santoro, C. “A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms”, Interacting with Computers, Vol.15, N.3, pp 347-364, Elsevier, 2003.

15. Paternò, F., “Model-Based *Design and Evaluation of Interactive Application”*. Springer Verlag, ISBN 1-85233-155-0, 1999.

16. Paternò, F., Leonardi, A. “*A Semantics-based Approach to the Design and Implementation of Interaction Objects”*, Computer Graphics Forum, Blackwell Publisher, Vol.13, N.3, pp.195-204, 1994.

17. Perez-Quinones M., Capra R., Shao Z., ”The Ears Have It : A Task by Information Structure Taxonomy for Voice Access to Web Pages “, Proceedings INTERACT 2003, pp.856-859.

18. Puerta A., Eisenstein J., "XIML: A Common Representation for Interaction Data", Proceedings IUI2002: Sixth International Conference on Intelligent User Interfaces, ACM, Gennaio 2002.

19. W3C - XForms – The Next Generation of Web Forms, http://www.w3.org/MarkUp/Forms.