Designing Multi-Device Interactive Services through Multiple Abstraction Levels

Fabio Paternò, Carmen Santoro

{fabio.paterno, carmen.santoro}@isti.cnr.it
ISTI-CNR
Via G.Moruzzi 1
Pisa, Italy

ABSTRACT
The purpose of this paper is to provide an introduction to the work done in our group on methods and tools for supporting design and development of multi-device interactive services. One important result of this work is the TERESA tool. This is a tool for model-based design of multi-device interfaces. It considers three levels of abstractions (task model, abstract user interface and concrete user interface). For each of them a specific language has been defined and used. In addition, since the lowest abstract level (the concrete interface) is platform-dependent, there are different variants for each platform considered.

Keywords
Multi-device services, XML user interface languages, tools.

INTRODUCTION
With the advent of the wireless Internet and the rapidly expanding market of smart devices, designing interactive applications supporting multiple platforms has become a difficult issue. The main problem is that many assumptions that have been held up to now about classical stationary desktop systems are being challenged when moving towards nomadic applications, which are applications that can be accessed through multiple devices from different locations. Consequently, one fundamental issue is how to support software designers and developers in building such applications. In particular, there is a need for novel methods and tools able to support development of interactive software systems that adapt to different targets while implementing usability design criteria.
Model-based approaches [5] could represent a feasible solution for addressing such issues: the basic idea is to identify useful abstractions highlighting the main aspects that should be considered when designing effective interactive applications. Our approach extends previous work in the model-based design area in order to support development of nomadic applications through logical descriptions and associated transformations.

By platform we mean a class of systems that share the same characteristics in terms of interaction resources. Examples of platforms are the graphical desktop, PDAs, mobile phones and vocal systems. Their range varies from small devices such as interactive watches to very large flat displays. While we think that designers should be aware of the potential platforms (not devices) early on in the design process, so they can identify the tasks suitable for each, our method allows developers to avoid dealing with a plethora of low-level details, because the last transformation (from concrete to implementation) is automatic. In addition, the same languages are used to describe tasks and abstract interfaces for all platforms; only the language for describing concrete user interfaces is to some extent platform-dependent.

To support this approach, the TERESA (Transformation Environment for inteRactivE Systems representAtions) tool [3] has been designed and developed providing general solutions that can be tailored to specific cases. This tool supports transformations in a top-down manner, providing the possibility of obtaining interfaces for different types of devices from logical descriptions. It differs from other approaches such as UIML [1], which mainly consider low-level models. XIML [7] has similar goals but there is no publicly available tool supporting it.

Some usability criteria are incorporated into the tool transformations from task to user interface. This means that the tool is able to provide suggestions for selecting the most appropriate interaction techniques and ways to compose them. Such transformations guarantee a consistent design because the same design criteria are applied in similar situations. In addition, most of the functionality of the CTTE task modelling tool has now been integrated into TERESA, so that designers can use just one tool and not lose time switching between two different tools.

THE METHOD
Our method for model-based design is composed of a number of steps that allow designers to start with an overall envisioned task model of a nomadic application and then derive concrete and effective user interfaces for multiple devices:

- *High-level task modelling of a multi-context application*. In this phase designers develop a single

model that addresses the possible contexts of use and the various roles involved and also identify all the objects that have to be manipulated to perform tasks and the relationships among such objects. Such models are specified using the ConcurTaskTrees (CTT) notation [5], which also allows designers to indicate the platforms suitable to support each task.

- *Developing the system task model for the different platforms considered*. Here designers have to filter the task model according to the target platform and, if necessary, further refine the task model, depending on the specific device considered, thus, obtaining the system task model for the platform considered.
- *From system task model to abstract user interface*. Here the goal is to obtain an abstract description of the user interface composed of a set of presentations that are identified through an analysis of the task relationships. Each presentation is structured by means of interactors composed of various operators.
- *User interface generation*. In this phase we have the generation of the user interface. This phase is completely platform-dependent and has to consider the specific properties of the target device.

The advantage of this approach is that it is able to address all the possible relations between tasks and platforms. In general, when a multi-platform application is considered, it is important to understand what type of tasks can actually be performed in each available platform. We have identified a number of possibilities:

- *The same task can be performed on multiple platforms in the same manner*. There may be only some changes in attributes of the user interface objects from platform to platform. For example, a login is often performed in almost the same manner through different platforms.
- *Same task on multiple platforms but with different user interface objects.* An example of this case can be a selection task. One platform can support a graphical selection whereas another one can be able to support only a selection among textual links.
- *Same task on multiple platforms but with different domain objects*. This means that during the performance of the same task different sets of domain objects are presented.
- *Same task on multiple platforms but with different task decomposition.* This means that the task is sub-divided differently, with different sets of sub-tasks, depending on the platform.
- *Same task on multiple platforms but with different temporal constraints.* In this case the difference is in the temporal relationships among the subtasks. For example, vocal interfaces tend to serialize interactions that can be performed concurrently on graphical interfaces.
- *Dependencies among tasks performed on different platforms.* An example of this can be found in applications where the users can reserve their flight reservation through a desktop system, and this enables the possibility of getting real-time information regarding the flight through a mobile phone.

TOOL SUPPORT

TERESA is intended to provide a complete semi-automatic environment supporting a number of transformations useful for designers to build and analyse their design at different abstraction levels and consequently generate the user interface for various types of platforms.

A number of main requirements have driven the design and development of TERESA:

- *Mixed initiative*; we want a tool able to support 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.
- *Model-based*, the variety of platforms increasingly available can be better handled through some abstractions that allow designers to have a logical view of the activities to support.
- *XML-based*, each abstraction level considered can be described through an XML-based language.
- *Top-down*, this approach is an example of forward engineering. So, designers first have to create more logical descriptions, and then move on to more concrete representations until the final interface is obtained.
- *Different entry-points*, our approach aims to be comprehensive and to support various possibilities, including also when different set of tasks can be performed on different platforms. However, there can be cases where only a part of it needs to be supported and, for example, designers want to start with a logical interface description and not with a task model.
- *Web-oriented*, we decided that Web applications should be our first target. However, the approach can be easily extended to other environments (such as Java applications, Microsoft environments, …) by just modifying only the last transformation (from concrete interface to final interface).

The TERESA tool offers a number of transformations and provide designers with an integrated environment for generating XHTML interfaces for desktop, mobile phones and VoiceXML user interfaces. With the TERESA tool, at each abstraction level the designer is in the position of modifying the representations while the tool keeps maintaining forward and backward the relationships with

the other levels. For example, it maintains links between abstract interaction objects and the corresponding tasks in the task model so that designers can immediately identify their relations. This results in a great advantage for designers in maintaining a unique overall picture of the system, with an increased consistence among the user interfaces generated for the different devices and consequent improved usability for end-users.

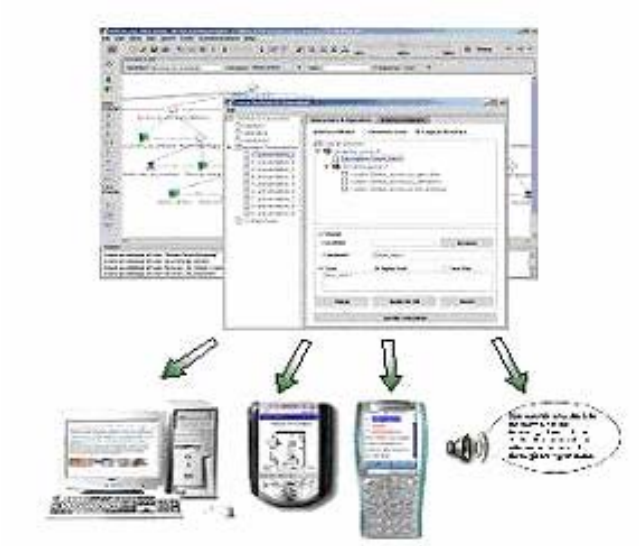

Figure 1: The TERESA tool.

The first transformation supported allows designers to obtain the abstract user interface corresponding to the initial task model. The abstract user interface is defined in terms of presentations. In each presentation there are interactors [6] and composition operators indicating how to put together such interactors. Once the elements of the abstract user interface have been identified, each interactor has to be mapped into interaction techniques supported by the particular device configuration considered (characterised by the modalities supported, the screen size, …), and also the abstract operators have to be appropriately implemented by highlighting their logical meaning: a typical example is the set of techniques for conveying grouping relationships in visual interfaces by using presentation patterns like proximity, similarity and continuity. However, different techniques for grouping elements are used in case of vocal interfaces, such as using a specific sound to delimit a set of elements.

The logical descriptions and the transformations defined in the method presented can also be used at run-time to support migratory interfaces: interfaces able to dynamically move from one device to another while preserving interaction continuity and adapting to the features of the new device [2].

The tool has provided a good opportunity to clarify various issues associated with the linkage between different models and the associated transformations, which must be fully understood in order to achieve real solutions and for which previous work in the area provided rather vague solutions. While the current TERESA version supports the design and development of graphical and vocal interfaces for various platforms (currently through the generation of XHTML, XHTML Mobile Profile and VoiceXML, though other languages are planned), further work will be dedicated to supporting a broader set of modalities and their combinations.

CONCLUSIONS

The TERESA environment supports design and development of multi-platform user interfaces through a number of transformations that can be performed either automatically or through interactions with the designer.

To this end, a number of XML languages that capture the relevant information at different abstraction levels are used. Such languages are introduced in this paper along with a discussion of how they are used in the environment. The tool can be freely downloaded at http://giove.cnuce.cnr.it/teresa.html.

Future work will be dedicated to supporting a broader set of modalities and their combinations

ACKNOWLEDGMENTS

We gratefully acknowledge support from the EU IST CAMELEON project (http://giove.cnuce.cnr.it/cameleon.html) and the EU SIMILAR NoE (www.similar.cc).

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. R. Bandelloni, F. Paternò, Flexible Interface Migration, Proceedings ACM IUI 2004, pp.148-157, Funchal, ACM Press, 2004.
3. G. Mori, F. Paternò, C. Santoro, Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions, accepted for publication on IEEE Transactions on Software Engineering, 2004.
4. Mullet, K., Sano, D., Designing Visual Interfaces. Prentice Hall, 1995.
5. Paternò, F., Model-Based Design and Evaluation of Interactive Application. Springer Verlag, ISBN 1-85233-155-0, 1999.
6. 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.
7. Puerta, A., Eisenstein, XIML: A Common Representation for Interaction Data, Proceedings ACM IUI'01, pp.214-215.