Methods and Tools for Designing and Developing Usable Multi-Platform Interactive Applications

Cristina Chesta[♠1], Fabio Paternò[2], Carmen Santoro[2]

(1) Motorola Electronics S.p.A. – Global Software Group
Turin, Italy

(2) I.S.T.I. - C.N.R.
Pisa Italy

ABSTRACT

The increasing availability of new types of interaction devices raises the need for new methods and tools to support the design and development of highly usable context-sensitive nomadic applications accessible through multiple platforms.

This paper provides an overview and discusses a solution based on the use of multiple levels of abstractions, which has been studied within the framework of the European project CAMELEON. Moreover it addresses the problem of evaluating the usability of these tools by discussing the specific issues, the criteria and methodologies applied as well as some results obtained in an experimental activity on the subject.

1. 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. In fact, on the one hand the decreasing cost at which the devices are now offered has enabled an increasing variety of people to become potential users of features and services of novel generations of communication technology as never before.

On the other hand, rarely such a high number of flourishing range of opportunities offered have become effective, due to the low quality of the user interfaces provided to the   users.

♠ Corresponding Author:
Cristina Chesta
E-mail: Cristina.Chesta@motorola.com

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. Each device is characterized by different interaction resources, including computational capability, accessible display area, interaction channels and network bandwidth. Moreover the interaction resources are subject to variations according to the physical and environmental conditions. For software developers, this introduces the difficult task of constructing multiple versions of single applications and endowing these versions with the ability to dynamically respond to changes in context. Currently, developers often create different versions of applications for different devices. This requires extra development, and maintenance costs and complicates the configuration management. A proliferation of versions reduces the resources available for usability engineering, and requires expensive maintenance of cross-platform consistency of the user interface.

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 able to adapt to different targets while preserving usability.

The evaluation of the tools for nomadic applications development requires specific criteria and methodologies enabling the assessment of usability and effectiveness from the double point of view, of the developer using the tool itself and of the final user dealing with the application implemented by exploiting the tool.

The paper presents some innovative techniques to provide software engineering support for the development of applications accessible through multiple heterogeneous platforms, which have been studied within the framework of the European project CAMELEON.

The paper is organized as follows. We present first a discussion on related work, a comprehensive vision of the project objectives and the approach adopted.

The next sections are dedicated to the current activities, with special focus to the ones carried-out at ISTI-CNR and Motorola GSG Italy. We introduce the TERESA tool for forward engineering design and development of multi-platform applications. Then we illustrate the case study proposed in order to provide a real application example and the experimental evaluation performed. Finally we report some preliminary evaluation results, followed by the concluding remarks.

2. The Approach

In a recent paper discussing the future of user interface tools Myers, Hudson, and Pausch (Myers et al., 2000) 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. This is also called user interface plasticity (Thevenin et al., 1999). Methods for modelling work context (Beyer & Holtzblatt, 1998) can provide useful information for this type of approach.

The basic idea of how to cope with the current situation of heterogeneity of currently available devices and the need for usable User Interfaces (UIs) is that, instead of having separate applications for each device, which exchange only basic data, there is some abstract description and an environment able to suggest a design suitable for a specific device.

This is the main goal of model-based design and development of interactive applications, which have been considered though not extensively adopted during last decade. Nomadic applications raise new challenges that can be better addressed using a model-based approach. There is a need for a unitary view of nomadic applications, even if their parts require different instantiation for different platforms. This allows designers to understand and control the dependencies among such instances. Secondly, new design criteria suitable for mobile devices should be introduced. The potentialities of these approaches have only begun to be addressed. In the GUITARE Esprit project (http://giove.cnuce.cnr.it/guitare.html) a user interface generator was developed: it takes ConcurTaskTrees (CTT) task models (Paternò, 1999) and produces user interfaces for ERP applications according to company guidelines. However, automatic generation is not a general solution because of many, varying factors that have to be taken into account within the design process. Semi-automatic support is more general and flexible: Mobi-D (Puerta, 1997) is an example of a semi-automatic approach, but it only supports design of traditional graphical desktop applications.

UIML (Abrams, 1999) is an appliance-independent XML user interface language. While this language is ostensibly independent of the specific device and medium used

for the presentation, it does not take into account the research work carried out over 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 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 of Web forms, based on the separation between the purpose and the presentation of a form. If it shows the importance of separating conceptual design from concrete presentation, it also highlights the need for meaningful models to support such approaches.

XIML (Puerta & Eisenstein, 2002) (eXtensible Interface Markup Language, http://www.ximl.org) is an XML-based language, whose initial development took place at the research laboratories of RedWhale Software. It is intended to be a universal user interface specification language, since it provides a way to completely describe a user interface and represent attributes and relations of the important elements of a user interface without worrying about how they will be implemented. In other words, it enables a framework for the definition and interrelation of interaction data items, thereby providing a standard mechanism for applications and tools to interchange interaction data and interoperate within integrated user-interface engineering processes, from design, to operation, to evaluation. Today XIML is probably the most advanced UI specification language, as it can serve for context sensitivity and many other objectives. However, it is worth noting that XIML mainly focuses on syntactic, rather than semantic aspects. In addition, tool support is not publicly available. Collagen (Rich & Sidner, 1998) uses an explicit embedded task model to support the creation of task-aware collaborative agents. The agent interprets and guesses the user's current intentions, and can determine efficient plans to achieve them. The issue related to platforms is not considered.

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 (Calvary et al., 2001), (Einsenstein et al., 2001) but there are still many issues that need to be solved to identify systematic, general solutions that can be supported by automatic tools.

The CAMELEON approach aims to support design and development of nomadic applications providing general solutions that can be tailored to specific cases, whereas current practice is still to develop ad hoc solutions with few concepts that can be reused in different contexts.

The actors in charge of adaptation depend on the phase of the development process:

- At the design stage, multi-targeting can be performed explicitly by humans such as system designers and implementers, and/or it can rely on dedicated tools.

- At run time, the adaptation may be performed by the user and/or the system. A UI is adaptable when it adapts at the user's request (typically, by providing preferences menus). It is adaptive when the user interface adapts on its own initiative.

A distinction can also be made between methods for forward engineering, allowing automatic generation of the interface for various targets starting from a common abstract description of the scenario to address, and methods for reverse engineering, which automatically transform web pages to pages at a certain level of abstraction, and these result pages can later on be transferred to other computing platforms.

Currently the forward engineering approach seems to be more promising, even if combining the two approaches by automatically reconstructing a task model from a web page and then automatically converting it for others platforms would open great opportunities from the application point of view.

A set of methods and tools supporting a number of transformations useful when designing multi-platform applications have been proposed by the CAMELEON consortium (Berti et al., 2003). At ISTI the TERESA tool has been developed, currently supporting transformations from task models to desktop, phone user interfaces and vocal interfaces (XHTML, XHTML Mobile Profile, and VoiceXML). Another tool called Web Revenge and supporting automatic reconstruction of task models from HTML code has been developed as well. A different approach to reverse engineering of web sites has been investigated at University of Louvain where the VAQUITA and RUTABAGA tools have been implemented supporting reconstruction of presentation models from HTML code. Unlike the previous tools, ArtStudio, developed at University of Grenoble, allows development of Java interfaces for multi-platform applications. Research activities are also ongoing about run-time mechanisms and infrastructure (Coutaz et al., 2003).

The set of tools demonstrate how the concepts and methods developed can be incorporated in real tools that can support the work of designers and developers in many types of software companies.

As far as the integration between the CAMELEON tools is concerned, effort has been put within the consortium particularly on the communication between two tools: VAQUITA and TERESA. Since the first one mainly covers an abstraction step while the second one covers reification, an example of interest for the consortium was judged to analyse the result of a two-step process in which e.g. the output of reverse-engineering a web page in VAQUITA (first step: abstraction) becomes the input for TERESA tool to the aim of performing in turn a reification step on it and possibly re-design the user interface for another computing platform.

Such an integration has been achieved through the introduction of a common XML-based language, CameleonXML (Limbourg et al., 2004), used to describe abstract user interface and developed by the consortium having in mind the general goal of modelling and represent the different requirements about the design of multi-platform user interfaces that have been raised up to now by discussions within the project.

In order to validate the CAMELEON approach and to elicit requirements for the tools design, the industrial partners provided examples of application to real case studies. In particular Motorola GSG Italy proposed an *e-Desk* service allowing people to access from any place with different devices office productivity applications, including an *e-Agenda* offering calendar, appointment schedule and automatic reminder (Chesta & Fliri, 2003).

The multi-context interface of both e-Desk service and e-Agenda application has been realized through the support of TERESA tool, serving as basis for the experimental evaluation (Chesta et al, 2003).

3. The TERESA Tool

TERESA is a transformation-based environment supporting a number of transformations useful for designers to build and analyse their design at different abstraction levels and consequently generate the concrete user interface for a specific type of platform (Paternò, 1999), (Paternò & Santoro, 2003), (Mori et al., 2003).

The abstraction levels considered (see Figure 1 at the end of this section) are:

- *High level task modelling*: the output of this phase consists of the description of the logical activities that need to be performed in order to reach the users' goals. This description initially considers an integrated task model where all the activities that have to be supported have been specified. Next, the task model is refined and structured so as to identify the activities that have been supported for

each platform considered. For example, a nomadic task model could analyse the activities supported by a system for reserving a hotel room through a cell phone and a desktop system: such specifications might share portions of the task model (the activities are performed in the same way), while being different for other tasks (for example some details about the room might be neglected with the cell phone platform);

- *Abstract user interface (AUI):* in this phase the focus shifts to the interaction objects supporting task performance. After having obtained the task model for a specific platform, an abstract user interface is derived from it. It is defined in terms of presentations (the set of user interface elements perceivable at the same time), and each presentation is composed of a number of interactors (Paternò & Leonardi, 1994), which are abstract interaction objects identified in terms of their main semantics effects. For instance, going on with the hotel reservation example, at this level we will just consider that for selecting a specific hotel, we do need some widget supporting a *single selection* task: the implementation details of such an object are irrelevant at this moment, and for this reason we identify it as an *abstract* interaction object supporting a selection. An XML-based language has been specified in order to describe the organisation of the various interactors within the presentations. The structure of the presentation is defined in terms of elementary interactors characterised in terms of the task they support, and their composition operators. Such operators are classified according to the communication goals to achieve: a) Grouping: indicates a set of interface elements logically connected to each other; b) Relation: highlights a one-to-many relation among some elements, one element has some effects on a set of elements; c) Ordering: some kind of ordering among a set of elements can be highlighted; d) Hierarchy: different levels of importance can be defined among a set of elements.

- *Concrete user interface (CUI*): at this point each abstract interactor is replaced with a concrete interaction object depending on the type of platform and media available and with a number of attributes that define more concretely its appearance and behaviour. For example, the abstract interaction object we mentioned in the previous phase (an interactor supporting the selection of a specific hotel) could be rendered through a scrollable list-box on a desktop

platform and through a pull-down menu on a cell phone platform. It is worth pointing out that, at this level, there is still no mention about the specific language used for implementing such concrete objects.

- *User interface generation*: this phase is completely platform-dependent and has to consider the specific properties of the target device. The interactors are mapped into interaction techniques supported by the particular device configuration considered (operating system, toolkit, etc.), and also the operators defined in the language for abstract user interface are implemented with appropriate presentation techniques. At this level we should specify e.g. if the pull-down menu on the cell phone platform will be rendered through WML, or through XHTML Mobile Profile, etc..

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

- *Mixed initiative*: we want a tool able to support different level 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, then the call for effective models able to capture the relevant information that should be considered.

- *XML-based* languages have been proposed for every type of domain. In the field of interactive systems there have been a few proposals that partially capture the key aspects to be addressed.

- *Top-down*: this approach is an example of forward engineering. Various abstraction levels are considered, and we support cases when designers have to start from scratch. So, they first have to create more logical descriptions and then move on to more concrete representations until the final system.

- *Different entry-points*: our approach aims to be comprehensive and to support the entire task/platform taxonomy. However, there can be cases where only a part of it needs to be supported.

- *Web-oriented*: the Web is everywhere; therefore, for generality purposes, 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, etc.) because only the last transformation needs to be modified for this purpose.

The TERESA tool offers a number of transformations between different levels of abstractions and provides designers with an easy-to-use integrated environment for generating both XHTML and VoiceXML user interfaces (Berti & Paternò, 2003). 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 thanks to a number of automatic features that have been implemented (e.g. the possibility of links between abstract interaction objects and the corresponding tasks in the task model so that designers can immediately identify their relations). This result is 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.
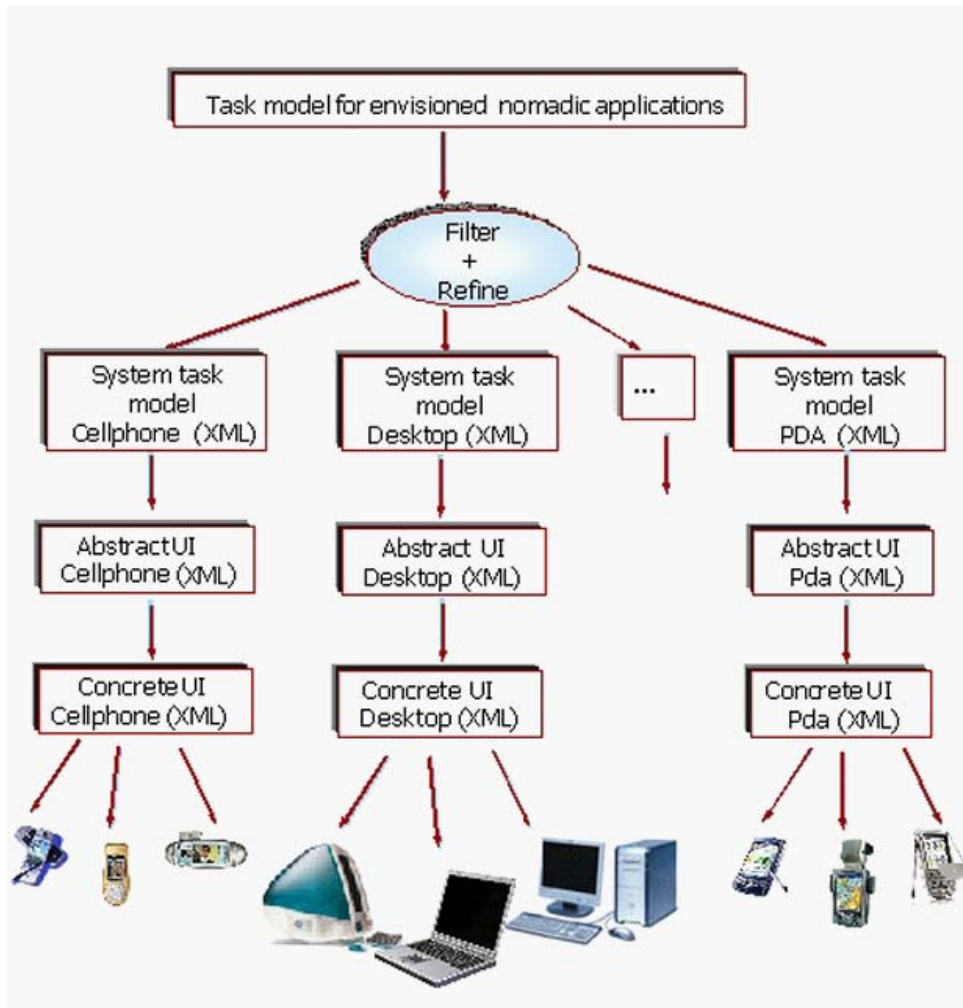
Fig.1: The Main Abstraction Levels of TERESA

4. Experimental Evaluation

The experimental evaluation has been conducted in parallel to the tool development in order to provide a formative rather than a summative evaluation.

Starting with the ISO 9241-11 standard definition (ISO9241-11, 1991) and Shneiderman's (Schneidermann, 1998) and Nielsen's (Nielsen, 1994) metrics, but considering the double perspective of the tool itself versus the product realized through the tool, we identified four aspects to be evaluated and eight related requirements as listed in Table 1.

Two experiments have been designed in order to cover different aspects according to the criteria framework formerly exposed. Both of them refer to the common application scenario related to Business to Employee environment.

Aspect	Requirement
Tool Interface	Intuitiveness
	Learnability
Tool Functionalities	Completeness
	Developer satisfaction
Final Product Obtained by employing the Tool	User Satisfaction
	Maintainability and Portability
Approach Cost/Effectiveness	Development Efficiency
	Integrability

Table 1: Evaluation criteria

Five subjects, selected within Motorola GSG Italy staff, were involved in the evaluation. All of them, within a range of different background and specialization, have technical knowledge and experience in software design and development, and are experienced computer users. They have been asked to participate in a 30 minutes preparation session and to dedicate 10 minutes reading the TERESA help prior to start the exercises.

The first experiment focused on tool usability and functional coverage, with the objective to highlight potential weaknesses and to provide design recommendations useful while implementing subsequent versions of the TERESA tool.

The experiment consisted in starting with a given task model created with CTTE 1.5.7 and obtaining the concrete user interface for both desktop and mobile phone using the version 1.1 of TERESA tool. The exercise goal was to realize a simple version of an e-desk application allowing three main actions: the registration to the service by inserting a username and a password, the selection of a location (workplace, home, travel or vacation), and the selection of an application from a menu. The applications offered are different in the desktop and in the mobile versions of the service.

The actions to be performed, such as Generate Enabled Task Sets, Generate Abstract User Interface, etc. were predefined in order to require the access to every tool menu. For each step evaluators were asked to record any difficulties they may have encountered in achieving the goal and their suggestions to improve the user interface. In addition, they were invited to provide comments about: approach, functionalities and result produced, reporting advantages/disadvantages with respect to traditional methods and providing indications on additional functionalities they would like to introduce.

The first evaluation resulted in an amount of data about the aspects considered. The analysis has been conducted in two steps. Firstly the raw comments have been abstracted to recurrent issues aggregated with a functional criterion, counting the occurrences of each issue; this step has been conducted iteratively, in order to progressively obtain a clean taxonomy. In the second step the taxonomy has been presented to the evaluators, who were requested to express for each issue a relevance assessment (high, medium, or low) a sort of quantitative measurement of the 'severity' of the problem; from such new data a relevance index has been synthesized for each item.

Table 2 reports as an example the analysis of the results related to the Final UI generation functionality. Seven main aspects requiring attention were identified.

Issue	*Occurrences*	*Relevance*
Final UI generation	*5/5*	
Messages language	1/5	low
Inserted data not reported	3/5	high
Destination folder	2/5	high
Windows unresizable and overlapping	3/5	medium
Not intuitive fields and controls	3/5	high
Not intuitive presentations structure and content	1/5	high
Relation operator for mobile UI	2/5	high
No browse button for URL	1/5	medium
Window consistence	1/5	low
Confirmation panel	2/5	high

Table 2: Example of analysis results

The results of analysis have been reported to the development group, which integrated them in the new version of TERESA used for the second experiment.

The next version of TERESA was substantially improved with respect to the first prototype taking into account the results of the experiment. For example the effect of the heuristics used for combining two or more PTS has been made more predictable, the AUI generation window has been redesigned in order to be intuitive and usable, the Final User Interface Generation has been improved by the introduction of a preview windows, the task corresponding to an object can be automatically identified, and some model editing options have been introduced.

A second experiment has then been conducted in order to collect more information about developer satisfaction and cost/effectiveness of the approach.

The subjects involved in the second experiment were the same people that performed the first experiment, so they could better appreciate the modifications introduced in the tool and provide specific feedback.

The experiment consisted in developing a prototype version of an e-Agenda application running on both desktop and mobile phone and including the following functionalities: visualization of the appointments of a single day; visualization of the details of each appointment; possibility of inserting/modifying/deleting an appointment. This had to be realized in two ways:

- At first using traditional techniques such as a template for the design phase and Microsoft Front Page or Netscape Composer for the implementation phase.

- Then using tool-supported techniques: CTTE 1.5.7 for task tree realization and version 1.5 of TERESA tool (updated taking into account the results of the first experiment) for XHTML and XHTML Mobile Profile pages generation.

Every evaluator had been asked to perform the same task using the two approaches, in the order specified above.

The evaluators have been required to collect quantitative metrics related to development efficiency, such as the total effort needed to complete the exercise expressed as creation or rework time and categorized by process phase, as well as the number of errors introduced. Moreover, they have been required to express their judgment on specific TERESA characteristics such as support offered to identify the most suitable interaction techniques, support offered to compose interactors in the interface, and others aspects related to developer satisfaction and product maintainability/portability by a rating from 1 (poor) to 5 (very good). In case of negative evaluation they were invited to provide an explanation note and suggestions for improvement.

The results of the second experiment show how developers' productivity is affected by the use of the tool. Data about time performance have been collected in each phase of the experiment and summarized through average values.

Results graphically illustrated in Figure 2, show similar total times for the traditional and TERESA approaches, with different distributions over the development phases and between first version and rework time.

The tool-supported methodology offers a very good support to fast prototyping, producing a first version of the interface in a significantly shorter time.

This difference is significant and interesting, considering that developers are often required to implement in a short time different interface prototypes to present and discuss with their customers, and then to refine later the selected version.

On the other side rework time results increased. In particular the design phase results negatively affected while the development phase is positively impacted by the use of the semi-automatic environment.

This is mainly due to the greater familiarity of the subjects with traditional techniques than with model-based techniques and notations. Future refinements to TERESA and a continuous use of the tool in the software production process are then expected to consistently reduce rework time needed and to confirm the advantages of the proposed tool supported methodology.
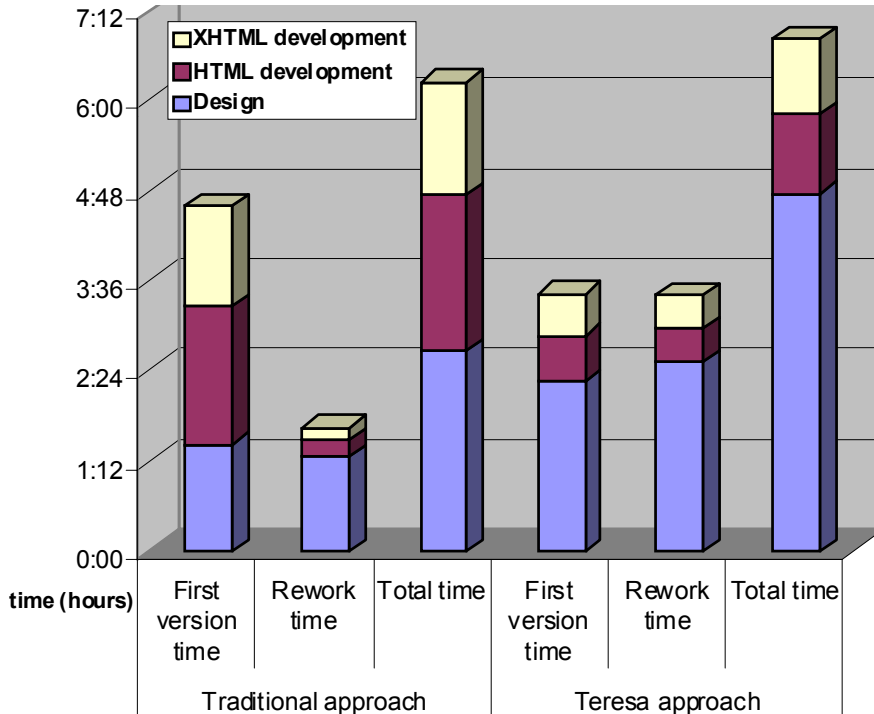

Fig. 2: Comparative results on time performance.

Even more interesting than the time performance itself have been the comments of the evaluators, who remarked an increased design overall quality and appreciated the benefits of a formal process supporting the individuation of the most suitable interaction techniques. For example, the subjects reported satisfaction about how the tool supported the realization of a coherent page layout and identification of links between pages; they noticed and appreciated the improved structure of the presentations and more consistent look of the pages resulting from the model-based approach, as well as the reduced risk to forget the formal specifications; they pointed out an increased consistence between desktop and mobile version.

5. Conclusions and Acknowledgements

In this paper a model-based approach for designing and developing multi-platform applications has been presented and discussed through an experimental evaluation.

In summary, TERESA emerged from the evaluation as an appealing and promising solution for designing and developing UIs on multiple and heterogeneous devices.

At the same time the evaluation methodology and criteria we introduced appears to be general and applicable to different systems.

Further activities will include additional experiments focusing on the final product and involving end users.

The TERESA tool is publicly available at http://giove.cnuce.cnr.it/teresa.html.

This work has been supported by the IST V Framework CAMELEON (Context Aware Modelling for Enabling and Leveraging Effective interaction) project. More information is available at http://giove.cnuce.cnr.it/cameleon.html.

We also would like to thank the colleagues Cristina Barbero, Simone Martini, Bianca Russillo and Massimiliano Fliri for participating to the experimental evaluation and for the useful discussions.

6. References

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

Berti, S., Mori, G., Paganelli, L., Paternò, F., Santoro, C., Calvary, G., Coutaz, J., Thevenin, D., Vanderdonckt, J. & Bouillon, L., (2003) *Tools for Model-Based Design of Multi-Context Applications*. Deliverable D2.1, CAMELEON project.

Berti, S. & Paternò, F., (2003) *Model-based Design of Speech Interfaces*. Proceedings of DSV-IS 2003, Funchal, Madeira, Springer Verlag.

Beyer, H. & Holtzblatt, K. (1998) *Contextual Design: Defining Customer Centred Systems*. Morgan Kaufmann, San Francisco.

Calvary, G., Coutaz, J. & Thevenin, D. (2001) *A Unifying Reference Framework for the Development of Plastic User Interfaces*. Proceedings of EHCI 2001, Toronto, Canada, Springer Verlag.

Chesta, C. & Fliri, M., (2003) *Early e-Desk Prototype Description*. Deliverable D3.3, CAMELEON project.

Chesta, C., Fliri, M., Martini, S., Russillo, B., Barbero, C. & Raymond, S., (2003) *First Evaluation of Tools and Methods*. Deliverable D3.4, CAMELEON project.

Coutaz, J., Balme, L., Barralon, N., Calvary, G., Demeure, A., Lachenal, C., Rey, G., Bandelloni, R. & Paternò, F., (2003) *Initial Version of the CAMELEON Run Time Infrastructure for User Interface Adaptation*. Deliverable D2.2, CAMELEON project.

Einsenstein, J., Vanderdonckt, J. & Puerta, A. (2001) *Applying Model-Based Techniques to the Development of UIs for Mobile Computers*. Proceedings of IUI'01, Santa Fe, New Mexico, ACM Press.

ISO9241-11 (1991) *Ergonomic requirement for office works with VDT's – guidance on usability*. Technical report, International Standard Organisation.

Limbourg, Q., Bouillon, L., Vanderdonckt, J., Michotte, B., Santoro, C. & Paternò, F., (2004) *CameleonXML V1.0 Description Document*. Deliverable 1.3, CAMELEON project.

Mori, G., Paternò, F. & Santoro, C. (2003) *Tool Support for Designing Nomadic Applications*. Proceedings of IUI'03, Miami, Florida.

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

Nielsen, J. (1994) *Usability Engineering.* Morgan Kaufmann, San Francisco.

Paternò, F. & Leonardi, A., (1994) *A Semantics-based Approach to the Design and Implementation of Interaction Objects*. Computer Graphics Forum, Blackwell Publisher, 13(3), 195-204.

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

Paternò, F. & Santoro, C., (2003) *A Unified Method for Designing Interactive Systems Adaptable to Mobile and Stationary Platforms.* Interacting with Computers, Elsevier, 15(3), 347-364.

Puerta, A.R. (1997) *A Model-Based Interface Development Environment*. IEEE Software, 14(4), 40-47.

Puerta, A.R. & Eisenstein, J. (2002) *XIML: A Common Representation for Interaction Data*. Proceedings of IUI'02, San Francisco, California, 214-215.

Rich, C. & Sidner, C. (1998) *COLLAGEN: A collaboration manager for software interface agents*. User Modelling and User-Adapted Interaction, 8(3/4), 315-350.

Schneiderman, S. (1998) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison Wesley, Reading, MA.

Thevenin, D. & Coutaz, J. (1999) *Plasticity of User Interfaces: Framework and Research Agenda.* Proceedings of Interact'99, Edinburgh, Scotland, 110-117.