Deriving Multiple Interfaces from Task Models of Nomadic Applications

Fabio Paternò
CNUCE-C.N.R.
Via V.Alfieri 1
56010 Pisa, Italy
+39 050 3153066
fabio.paterno,@cnuce.cnr.it

ABSTRACT

The wide availability of many types of devices has become a fundamental challenge for designers of interactive software systems. Here we discuss a model-based method for the design of nomadic applications and the types of transformations that it requires to support the design of such applications. The aim is to enable each interaction device to support appropriate tasks that users expect to perform and designers to develop the various device specific application modules in a consistent manner.

Keywords
Task models, Multiple interaction devices, Nomadic Applications.

BACKGROUND

In a recent paper, discussing the future of user interface tools, Myers, Hudson, and Pausch [3] indicate that the wide platform variability *encourages a return to the study of some techniques for device-independent user interface specification, so that …*. 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 is also called user interface plasticity [1].

This problem is a novel challenge for model-based design and development of interactive applications [4]. 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. Mobi-D is an example of this approach developed at Stanford University, but it only supports design of traditional graphical desktop applications.

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) which 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 [1, 2], but there are still many issues that should be solved to identify systematic, general solutions with the support of automatic tools.

Our approach provides designers with environments supporting the design and development of nomadic applications: in these applications users have access to both their own personal and public information from wherever they are, independent from specific devices (mobile, stationary desktop or kiosk system), all of which have access to information spaces relevant for the user. We aim 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 [5].

OUR APPROACH

The design of multi-platform applications can follow different approaches. 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. Another, more promising option is to consider different devices also with regard to the choice of the tasks to support. 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 is composed of a number of phases:

- *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.
- *Developing the system task model for the different platforms considered*. Here designers have to filter the task model according to the target platform. 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. Thus, we obtain the system task model for the platform considered. These task models are specified using the ConcurTaskTrees notation because it allows description of flexible behaviours and is tool supported (http://giove.cnuce.cnr.it/ctte.html).
- *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 we identify the possible transitions among the presentations still with the help of the task model (considering the temporal relationships that it indicates) and within each presentation we indicate the structure of the presentation.
- *User interface generation*. In this phase we have the generation of the user interface. This phase is completely platform-dependent and it has to consider the specific properties of the device considered. It is not sufficient to say, for example, that we want consider a cellular phone but we need to consider the type of microbrowser is supported, the number and the types of softkeys available. If we consider PDAs we need to know whether they support colours and audio output or not. Likewise if we consider desktop system we need to know if they have good multimedia possibilities and, if they are only graphical desktop system the resolution and the size of the screen available.

CONCLUSIONS

In this paper we have introduced and discussed a method that is able to support design of applications supported by multiple-devices starting from the task model of a nomadic application. Through the use of a number of intermediate abstractions it is possible to achieve a design that is able to support effectively the activities that can be used through the various possible devices in different contexts.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the GUITARE project (http://giove.cnuce.cnr.it/guitare.html).

REFERENCES

1. G. Calvary, J. Coutaz, D. Thevenin A Unifying Reference Framework for the Development of Plastic User Interfaces, Proceedings EHCI 2001, to appear
2. J.Einsenstein, J.Vanderdonckt, A.Puerta, Applying Model-Based Techniques to the Development of UIs for Mobile Computers, Proceedings IUI'01: International Conference on Intelligent User Interfaces. 2001. ACM Press.
3. 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.
4. Paternò, F., Model-Based Design and Evaluation of Interactive Application. Springer Verlag, ISBN 1-85233-155-0, 1999.
5. R.Oppermann, M.Specht, A Context-Sensitive Nomadic Exhibition Guide, Proceedings Symposium on Handheld and Ubiquitous Computing, pp.127-142, LNCS 1927, 2000.