User Interfaces: A Proposal for Automatic Adaptation

Montserrat Sendín[1], Jesús Lorés[1], Francisco Montero[2], Víctor López[2], and Pascual González[2]

[1] Computer Science and Industrial Engineering Department, University of Lleida, Spain
{msendin, jesus}@eup.udl.es
[2] Higher Polytechnic School of Albacete, University of Castilla-La Mancha, Spain
{fmontero, victor}@info-ab.uclm.es

Abstract. In previous works we have been developing a tourism support prototype that offers a proven solution for aspects of multi-platform, personalisation and spatial-awareness. The aim of this paper is to analyse its drawbacks and limitations, and to propose an architecture that solves these problems relying on the principle of abstraction. It consists of a reflexive architecture based on models, and component-oriented, that allows specifying the user interface independently of the rest of the implementation. Thus, developers only have to focus on modelling the functionality of the application –which resides at a base level-, leaving the interface to a meta level, constructing thereby *applications without apparent interface*. The generation of the UI is not carried out until run-time, applying an automatic translation from abstract interaction components to concrete ones according to the device and the user's features and current state.

1 Introduction

Mobile computing offers the possibility of dramatically expanding the versatility of computers, by bringing them off the desktop and into new and unique contexts. As it grows, it offers a wider variety of devices suitable for multiple and varied *contexts of use*[1]. However, to take advantage of all these new possibilities, UI designers are forced to accommodate a growing variety of devices and contexts of use, as well as to solve any contextual or environmental changes, while preserving usability. Only that way it will be possible to create UIs that get together all of the demanded features: adaptivity, context-awareness, device independence and usability.

In this line, what we intend is to exploit the possibilities of mobile computing in the tourism sector. Our specific aim is to develop a generic tourism support tool that gets together all of the previously mentioned features, applicable to any tourist enclave. Nevertheless, our experimentation laboratory has been until now the mountainous zone of Montsec. It is an area of outstanding natural and cultural wealth, situated in the western sector of the Catalan Pre-Pyrenees (province of Lleida).

The previous prototype we developed –tested and published in [1] -, already solved the adaptation to the device and to the user's profile, by providing a specific template

[1] It includes a set of environment parameters that describe a particular context in which a determined user is realizing a concrete task.

J.M. Cueva Lovelle et al. (Eds.): ICWE 2003, LNCS 2722, pp. 263–266, 2003.

in XSL (eXtensible Style Language), adapted to each context of use. However, that solution had strong limitations. On one hand, it is a method notably inflexible and difficult to maintain. On the other hand, it is necessary to diversify and extend the offer of pre-designed templates for each new demand, turning this task into a repetitive, complex and hard process that makes it difficult to preserve the consistency between the different interfaces. This approach could only be appropriate if the set of devices or of adaptation possibilities to different profiles taken into consideration was quite restricted. However, if we do not want it to fall quickly into disuse, we cannot ignore the increasing availability of diverse devices. It is accordingly necessary to make the accommodation to different platforms easier and more flexible, and thereby the generation of suitable UIs.

We think that many of the portability problems in UI development are caused by the low abstraction level at which design decisions are made, resulting in lack of overview in the design process. The abstraction level is the key, and also the base of programming paradigms, such as the *component-oriented paradigm* [2]. This paradigm proposes implementing systems by means of the incorporation of components adaptable to different contexts and formalised through *models*, as in a *model-based technique* -see [3] to look up a complete overview of some of the best-known model-based techniques. This is the design and implementation base that is proposed in this paper: the provision of as many concrete components of interface as necessary, which will be automatically selected according to the user's technology (device) and preferences. The rest of the application will work making use of the abstract interface components, obtaining therefore *generic UIs*[2], and accordingly reusable in other applications. The translation of abstract to concrete components, and also the relation between functionality and interface are situated at a meta level corresponding to a typical *reflexive architecture.*

2 Our Proposal

2.1 Environment Recognition

As a tourism support tool, our system also has to solve all kind of spatial consultations, thus providing the *spatial-awareness* property. In order to offer environment recognition, a detailed spatial model from the real world is required, which has to be built and integrated into the architecture.

The modelling is being solved using an object-oriented approach - this makes the adaptation to each particular case easier, following the concept of ontology-, in an XML-based language. Henceforth, we will name this model Augmented World model (AW-m). It provides an integrated and homogeneous view of the data.

The automatic extraction of the objects in the model that may interest the user – according to his profile, state and current circumstances- with regard to position, is going to be delegated to a spatial data managing module: the *Nexus platform* [4]. We can define this as a global and open infrastructure, sufficiently versatile to contact to any spatial-awareness application –independently of its functionality-, provided that

[2] UI whose aspects can vary in different devices, while its functionality prevails in all of them.

the application works with XML-based languages to define the AW-m, and also with XML-based query languages to formulate the queries.

The essential components of the AW-m are the static geographical objects (in the Montsec case: castles, towers, monasteries, rural areas, churches, etc), the location information of mobile objects (the zone visitors), and also virtual objects which provide additional information or services to Nexus users. The answer to a query consists of a list of matching objects in a particular zone. For example, a list of the nearest restaurants with their respective attributes –address and menu-, if it is lunchtime, or a list of the castles that can be visited in the zone, according to attributes such as historical period and visiting timetable. This would be suitable if our user had a certain preference for History. Other parameters that also take part in the adaptation are, for example, the information level required by the user, if he already knows a term related to the information -such as the features of the Romanesque style of a church-, the number of similar visits already realised, etc.

The Nexus platform is currently being developed at the University of Stuttgart [4]. For more details about it or how to join our prototype, consult [1].

2.2 The Reflexive Architecture

Under an object-oriented reflexive architecture view [5], [6], a system is considered to be integrated by two parts: the application part and the reflexive part, which reside at two different levels: the base level and the meta level, respectively. The application is represented at the base level and is manipulated by the meta level. Both levels are connected causally, so that changes in the base level are reflected in the meta level.

Our objective consists of developing a system where the part destined to contemplate the interface and the part destined to the functionality are independent. At the base level, following the abstraction principle, there is no conception of interface. The interface will be taken on in the meta level, at run-time, fixing which concrete interface components will represent the functionality described by the abstract ones, depending on the device and the user's profile. Other responsibilities of the meta level will be the interaction validation, the aid to the user in his actions, etc, in a transparent way for the user. Following the *component-oriented programming approach* [2], the base level can be formed, independently of its functionality, by a control structure similar to a *states machine*, where each state is associated with a dialogue, identified in the design phase. Fig. 1 represents the whole idea.

With respect to the required models, the ones that we consider relevant for applying the proposed architecture to tourism, are the next ones: user model, task model, dialogue model, presentation model and platform model. The AW-m described previously corresponds to the domain model.

In device-based UIs the connections between the platform model and the presentation model acquire crucial importance, because they tell us how the constraints posed by the various platforms will influence the UI visual appearance. In short, they determine the transition from abstract components to concrete ones, decision that depends in a great deal on the final device to interact with.

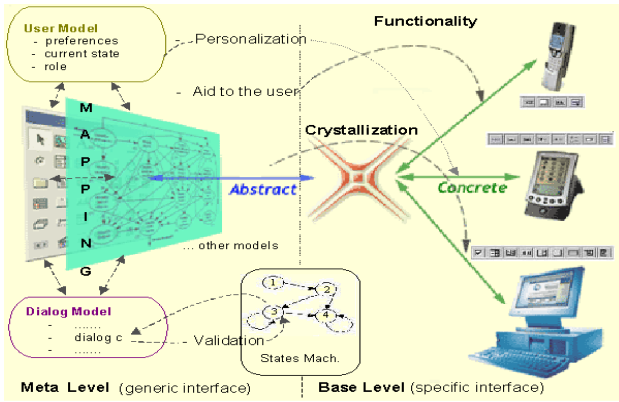

Fig. 1. Reflexive architecture and distribution of responsibilities

3 Conclusions

This paper proposes the construction of an *application without apparent interface*. As happens in the hardware elaboration, we can dispose of concrete interface components available for selection, leaving the interface generation at run-time. At this moment the suitable crystallization from abstract to concrete components occurs, making use of a reflexive architecture. One of the advantages of this mechanism is that the developer only has to focus on modelling the functionality of the application, leaving the interface to a meta level. This will be, for definition, reusable.

The use of abstract models to describe the UI makes the development of usable UIs to mobile devices substantially easier, providing an automated support that allows designers to overcome the challenges posed by mobile computing.

References

1. Sendín, M., Lorés, J., Solà, J.: Making our Multi-device Architecture Applied to he Montsec Area Heritage Adaptive and Anticipating. Proc. Workshop on HCI in Mobile Tourism Support (Mobile HCI 2002) Pisa, Italy (2002) 51-56
2. Szyperski, C.: Component Software-Beyond Object-Oriented Programming. Addison-Wesley. ACM Press. ISBN: 0-201-17888-5 (1998)
3. Pinheiro, P.: The Unified Modelling Language for Interactive Applications. (2002) http://www.cs.man.ac.uk/img/umli/links.html
4. Grossmann, M., Leonhardi, A., Mitschang, B., Rothermel, K.: A World Model for Location-Aware Systems. Published on behalf of CEPIS by Novática and Informatik/Informatique http://www.upgrade-cepis.org., Vol. II, Issue, 5. Ubiquitous Computing (2001) 32-35
5. Maes, P.: Concepts and Experiments in Computational Reflection. Proc. of the 2nd OOPSLA'87 (1987) 147-156
6. Zimmerman, C.: Advances in Object-Oriented Metalevel Architectures and Reflection. CRC Press, Inc., Boca Raton, Florida 33431 (1996)