MODEL-BASED DESIGN AND NEW USER INTERFACES: CURRENT PRACTICES AND OPPORTUNITIES

José Pascual Molina Massó and Pascual González López
LoUISE Research Group
Instituto de Investigación en Informática de Albacete (I^3A)
Campus Universitario s/n
Albacete, 02071 Spain
+34 967 59 92 00
{jpmolina | pgonzalez}@info-ab.uclm.es

ABSTRACT
In conventional applications it is easy to find detailed and structured practices that make use of models in order to describe almost every aspect of the user interface. On the other hand, new user interfaces, such as those used in virtual and augmented applications, are usually designed and developed in an ad hoc fashion or following a rather simple systematic approach. There are, however, a few notable efforts to develop model-based design methods for these new interfaces. Current practices, research efforts and holes that are still to be addressed in the development of new user interfaces are reviewed in this paper.

Keywords
Model-based design, new user interfaces

INTRODUCTION
The new user interfaces that are usually known as non- or post-WIMP [6][13] are those that aim to free us from the tyranny of the PC desktop and its windows-based interface, offering a style of interaction much closer to the human being that is not only based on the conventional pointing device, buttons and menus. Examples of this kind of interface can be found in virtual reality, augmented reality and ubiquitous computing applications. These user interfaces are designed to offer a more natural interaction, making learning a much easier task, and making use a more satisfactory experience. In order to achieve that goal, many designs aim to replicate our interaction in the real world as long as technology allows us to do that, as designers are sometimes forced to substitute some interaction processes by more feasible ones. Fortunately, current technology also allows us to offer new styles of interaction that empowers the user within the information space, through interactions that are not possible in the real world.

These new user interfaces have then some unique characteristics that make them radically different from conventional WIMP interfaces. In opposition to the "ping-pong" dialogue style of the latter, where the user consciously sends a command to the computer and waits for its answer, non- or post-WIMP user interfaces offer a new kind of dialogue based on several parallel channels, through which the user interacts not only in an explicit way, but also implicitly, as when he or she changes his or her point of view with an unconscious movement of the head within an immersive virtual reality application.

Virtual and augmented reality
Focusing our attention to virtual and augmented reality, one of the main features of these interfaces is that they are explorative in nature, which makes them more similar to hypertext interfaces, sharing some of the design and use problems, such as the user orientation within the information space. Another kind of interface, which can be found in may information points, have also a significant relationship, sharing the requirement of allowing any user to quickly learn how to use the system.

Development and related problems
All in all, the design of these new user interfaces is still a challenge for any developer because, as they have not reached the state of popularity and maturity that characterizes WIMP interfaces [1], it is quite difficult to form a working group that has enough experience to successfully deal with this task, following a proven systematic and structured process, and supported by a well-known software and hardware, in the form of virtual and augmented reality toolkits and devices. Besides, projects are usually sporadic and they do not last too long, and for that reason the gained experience is not usually reused, nor is it transferred to other working groups.

A NAÏVE APPROACH TO DEVELOPMENT
As it has been explained, the development of a virtual or augmented reality application is not an easy task. However, most projects are not usually so demanding, such as virtual walkthroughs or virtual prototype visualizations, which makes development a great deal easier. Under these conditions, the main task can be broken down into four main steps: object geometry and appearance definition,

import models to the virtual reality toolkit, object behavior definition, and virtual environment visualization in a virtual reality facility. The main advantage of this approach is that it allows existing working groups, commonly formed by programmers and 3D designers, to get access to this kind of technology in a easy way. This naïve approach is usually offered by several companies, such as EON Reality, allowing them to broad their market.

In Kaur's PhD thesis [7], it can be found an interesting study that describes the methods used in the development process by people involved in virtual reality projects. This study revealed a work pattern that can be summarize in five steps:

a. Requirements specification;
b. Gathering of reference material from real world models;
c. Structuring of the graphical model and, sometimes, dividing it between designers;
d. Building objects and positioning them in the VE;
e. Enhancing the environment with texture, lighting, sound and interaction, and optimizing the environment.

Development approaches as the one that Kaur describes can be useful for less complex projects and, in fact, we are currently using this approach –with subtle enhancements– in a real project of a virtual walkthrough of a technology park. However, as Kaur found, even the application of such methodologies on simple VEs does not lead to efficient interfaces, as there are other factors that have to be taken into account, such as usability. Besides, this kind of methodology are difficult to apply to more complex projects that aim to make use of all the unique characteristics that virtual and augmented reality offers.

THE CURRENT ROLE OF SOFTWARE ENGINEERING

Although many projects are tackled using an informal working strategy, usually due to the incunabula status of the technology, there is no doubt that a more structure process is desirable, guiding the development team and leading it to project success. In this sense, software engineering practices can be presented as the key to that success, bearing in mind the important role they take in conventional user interface development.

The application of software engineering practices in the development of new interfaces is not a well-documented issue, as most publications focus their attention on the technological side of the interfaces, and only a few describe the development process. Despite that, there are papers that present different approaches to the development of this kind of interfaces from a software engineering point of view, making use of notations, models and techniques that are characteristics of software development processes. After performing a broad survey on this topic, the results show that these software engineering practices are usually applied at two different levels:

- The first level make use of specification practices such as task hierarchical analysis (HTA) [9], annotated scene graph [11] or UML use cases [3]. The level of abstraction at this point is, therefore, high, and pursues the description of the user's tasks and the interaction processes that relates the user and the application objects.
- A second level is based on formalisms such as augmented transition nets (ATNs) [4][9], Petri nets [10], data-flow diagrams [6][10] and state charts [6]. This formalisms are used to describe the fine-grained detail of the system, such as the interaction techniques that the user will use in the application or the different object behaviors, modeling the flow of data that takes place within the system. At this point, it should be remarked the research effort that has been done on new formalisms that describe both the continuous and discrete flows of information that characterizes these new interfaces, a kind of formalisms that are known as hybrid notations [6][10].

The connection between the two previous levels is usually established through the refinement of the hierarchy of tasks, until the low-level subtasks are identified, which correspond to the basic interaction processes whose description must be done using a much more appropriate formalism. In opposition to WIMP user interfaces, there is not a set of interaction objects and techniques (usually called widgets 3D [4]) that can be claimed to be shared by most of the different applications, although there is an ongoing research effort in this topic [5].

A similar discussion could be made on the current role of usability engineering on the development of these new user interfaces, although this matter is outside the scope of this paper. Anyhow, it is worth pointing out that this is also an important topic of research within the field, and that much research is aimed to understand the unique characteristics of these interfaces with regard to their evaluation, which is usually done by applying methods that have been shown to be effective for the evaluation of WIMP user interfaces [2].

MODEL-BASED DESIGN FOR NEW USER INTERFACES

Even though the models and formalisms that have been cited in the previous section can be thought of being simple and easy to apply, they can give us the wrong impression that the development of these new interfaces is simpler than it was formerly said and, for that reason, those models and formalisms are enough to carry out the development. Unfortunately, there are still many holes in the design process, which are usually managed thanks to the know-how and experience of a skilled working group. Thus, it is a common practice that many aspects of the application are described by informal descriptions using natural language.

On the other hand, in the field of conventional user interfaces –mainly those that grasp the most attention, such

as Web interfaces– it is easier to find detailed structured processes that make use of models at different stages of the development cycle in order to describe every single aspect of the final product, allowing the designers to use the most appropriate notation at each step. This model-based approach has many advantages, and one of them is that model compilers can be developed to generate code from those models, so that the final coding phase is automated to a great extent.

In a similar way, the application of a model-based methodology should have a great impact on the development of new user interfaces:

- New models: they can be proposed in order to help developers in the design of the information space that is presented to the user, as well as the interaction resources that will have at his or her disposal in that space. Modeling languages such as UML can be thought of an example to follow or even as a starting point towards a modeling language of new user interfaces, which obviously requires extensions that meet the unique features of these paradigms [8].
- Communication: models must facilitate the information exchange within the working team, which is usually a heterogeneous one, formed by experts of different skills, such as interaction designers, 2D/3D designers, programmers and evaluators. Bearing in mind that is not an easy task to recruit such an expert team, models should act as the common ground that allows that information exchange.
- Documentation: one of the major problems regarding the development of new interfaces is their reuse in new projects. Models can then be used to document the application, presenting all the information in a comprehensible way that can be transmitted to other working groups.
- Reverse engineering: taking into account that many projects are carried out following an informal development process and that they are usually poorly documented, another major topic of interest are tools that allow us to generate models from a previously done project, decreasing the effort needed to understand what was done in that project. These tools could work in an automatic or semi-automatic fashion.

As it can be seen after reading the last point, it is not only important to have models but also to have software tools that allow us to work with them.

One example that shows how a model-based software tool can be of valuable help is the abstract modeling of the interface. In this case, the aim is to create a prototype of the virtual space using simple objects, similarly to what 3D designers usually do, but this time the prototype could be created by domain experts or interaction designers, who are not used to 3D design environments. For this reason, the tool should be much easier to use. In fact, this tool could also be based on new user interfaces, similar to the ones that are to be designed, such as for instance an immersive design tool.

CONCLUSIONS

A fundamental lesson learned in software engineering is that improvements in design and quality assurance processes require systematic work practices that involve well-founded methods [12]. However, new user interfaces such as those used in virtual and augmented reality applications are very much in the incunabula stage of development, where the technology exists but its effective application is only partially understood. Despite most of those interfaces are designed and developed in an ad hoc fashion or following a naïve systematic approach, there are remarkable attempts in applying models as a part of the development process. In this short paper, we have shown some of these approaches, and we have identified some holes in development lifecycle where MB-UID can also be applied to, two topics that can be worth discussing at the workshop.

REFERENCES

1. Bowman, D.A., Kruijff, E., LaViola, J.J., and Poupyrev, I. An Introduction to 3-D Interface Design, Presence 10, 1, 2001, 96-10.
2. Bowman, D.A., Gabbard, J.L., and Hix, D. A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods. Presence: Teleoperators and Virtual Environments 11, 4, 2002, 435-455.
3. Boyd, D., and Sastry, L. Development of the INQUISITIVE Interaction Toolkit-Concept and Realisation. User Centered Design and Implementation of Virtual Environments (UDCIVE) Workshop, University of York, September 1999. Available at: http://www.cs.york.ac.uk/hci/kings_manor_workshops/UCDIVE/
4. Conner, D.B., Snibbe, S.S., Herndon, K.P., Robbins, D.C., Zeleznik, R.C., and van Dam, A. Three-Dimensional Widgets. Special Issue of Computer Graphics: Proceedings of the 1992 Symposium on Interactive 3D Graphics, ACM Press, 1992, 183-188.
5. Dachselt, R., Hinz, M., and Meissner, K. CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications. Proceedings of the ACM Web3D Symposium, Tempe, USA, February 2002, 24-28.
6. Jacob, R.J.K., Deligiannidis, L., and Morrison, S. A Software Model and Specification Language for Non-WIMP User Interfaces. ACM Transactions on Computer-Human Interaction, Vol. 6, No. 1, March 1999, 1-46.
7. Kaur, K. Designing virtual environments for usability, PhD thesis, Centre for HCI Design, City University, London, 1998.

8. Molina-Massó, J.P., González-López, P., Lozano-Pérez, M.D., Montero-Simarro, F., and López-Jaquero, V. Bridging the gap: developing 2D and 3D user interfaces with the IDEAS methodology. Preliminary proceedings of the 10th Workshop on the Design, Specification and Verification of Interactive Systems, DSV-IS, Madeira, Portugal, June 2003, 297-309.
9. Murray, M., and Fernando, T. A Task Manager for Virtual Environments. User Centered Design and Implementation of Virtual Environments (UCDIVE) Workshop, University of York, September 1999.
10. Smith, S., Duke, D., and Massink, M. The Hybrid World of Virtual Environments. Computer Graphics Forum 18, 3, September 1999, 287-307.
11. Smith, S., and Duke, D. Design Support for Virtual Environments. Visualization and Virtual Environments Community Club (VVECC) Workshop: Design of Virtual Environments, January 2001. Available at: http://www-ais.itd.clrc.ac.uk/VVECC/proceed/vedesign/
12. Sommerville, I. Software Engineering, 5th edition, Addison-Wesley, 1999.
13. van Dam, A. Post-WIMP User Interfaces, Communications of the ACM 40, 2, 1997, 63-67.