Open Issues for the development of 3D Multimodal User Interfaces from an MDE perspective

Joan De Boeck[1], Juan Manuel Gonzalez Calleros[2], Karin Coninx[1], Jean Vanderdonckt[2]

[1]Hasselt University, Expertise centre for Digital Media (EDM) and transnationale Universiteit Limburg
Wetenschapspark 2
B-3590 Diepenbeek, Belgium

{joan.deboeck, karin.coninx}@uhasselt.be

[2]Université catholique de Louvain, School of Management (IAG), Belgian Lab. Of Computer-Human Interaction (BCHI)
Place des Doyens 1
B-1348 Louvain-la-Neuve, Belgium

{gonzalez, vanderdonckt}@isys.ucl.ac.be

ABSTRACT

Given its current state of the art, Model-Based UI Development (MBDUI) is able to fulfill the major requirements of desktop and mobile applications, such as form-based user interfaces that adapt to the actual context of use. More recent research deals with the development of 3D interactive multimodal environments. Though user-centered design is more and more driving the design of these environments, less attention is devoted to the development processes than to interactive tools supporting isolated phases in the realization process. In this paper we describe our findings when considering model-based development of 3D multimodal applications in the context of model-driven engineering. We concentrate on the requirements of such a process, the models being used and the transformations that are able to guide or even automate part of the development process for the envisioned applications. We conclude with some open issues that have been discovered.

1. INTRODUCTION

Model-based development of user interfaces (MBDUI) is finding its way from academic research to practical applications in industrial projects. While the principles of MBDUI have largely been investigated for traditional form-based desktop UIs [9,10,14], the need for flexible development of contemporary interactive applications has raised the attention for a model-based approach. Mobile applications [10], (context-sensitive) multi-device user interfaces [2,3,10], distributed and migratable user interfaces [3,14] are already emerging, and will gain importance with the realization of pervasive computing.

Multimodality, including speech input, voice output and pen-based interaction, is a central topic in many research projects. However, most of the contemporary research activities in the area of model-based UI development concentrate on 2D applications, in which interaction is done in two dimensions with traditional or pen-based input, even when working with 3D scenes or data.

In order to interact with these 3D objects in a multimodal way, several methods have been introduced ([1,4,7,11,15,16]) but none of them is truly based on genuine models for the whole development life cycle. Most are focusing directly on programming issues rather than on the design and analysis of the final application. This is sometimes reinforced by the fact that available tools for 3D UI design are toolkits, interface builders, or rendering engines.

Based on our former experience with the realization of interactive virtual environments (IVEs) on the one hand, and with model-based development of multi-device applications on the other hand, our current research activities concern on bridging the gap between both techniques.

In order to solve the shortcomings of current model-based design approaches for IVEs, we investigate the possibilities of a tool-supported development process for virtual environment applications. To specify this development process we will first gather some requirements, based on existing tools and processes. Afterwards we will elaborate on two model-based approaches and compare them with respect to the identified requirements. We investigate which requirements are fulfilled and what the problems are in both processes. Finally, some open issues will be presented that have been discovered during their implementation and evaluation.

2. REQUIREMENTS

We expect model-based development of interactive 3D environments to be successful when it is conceptualized as a combination of two different development approaches, namely MBUID and the toolkit-based development of IVEs. Both approaches have been examined for their benefits in order to gather the requirements necessary to define our process.

An overview of model-based processes (e.g., [2,3,9,10]) shows that they have several common properties. Nearly all processes start with some kind of a task model and evolve towards the final user interface using an incremental approach. During each increment, a model is converted into the next by means of an automatic transformation (through mapping rules) or manual adaptation by the user. Although these model-based approaches have shown their value in dialog and web-based interface generation, none of them seems directly usable to design IVEs, since they all lack the possibility to describe direct manipulation techniques and metaphors. A good MBDUI should therefore consider both the UI widgets and the description of possible interaction techniques.

In contrast with the MBUID tools that have been studied, tools and toolkits to develop interactive virtual environments (e.g., [4,16]) do not immediately show common characteristics. This is mainly due to the wide variety of existing toolkits, each focusing on a specific part of the final application such as object design, scene composition and widget design. Combining these different toolkits is not easy since the output of one tool cannot, in most cases, be used as input for another tool. Therefore it is important that several design issues can be integrated within the same process, containing code generation algorithms that can be supported by some existing toolkits. An important issue for these code generation algorithms is that manual changes should be preserved after regeneration. Finally, graphical tool support should be offered in order to design the high-level description models, check their correctness and generate the final application.

3. PRACTICAL IMPLEMENTATIONS

In this section we will describe two model-based approaches for the design of IVEs, called CoGenIVE and Veggie, by means of the PIM-PSM pattern explained in [12] and depicted in Figure 1. This pattern starts with a Computing Independent Model (CIM) which is aimed at capturing general requirements of the future system independently of any implementation. From this model, a Platform Independent Model (PIM) is derived once a technological space has been selected. This model is in turn converted into a Platform Specific Model (PSM) by means of certain transformation rules once a particular target computing platform has been decided. This MDA pattern can be applied multiple times at these three levels, using the resulting PSM of the first pass as input PIM for the second pass. Usually, the initial CIM remains constant over time unless new requirements are introduced. In the remainder of this section CoGenIVE and Veggie will be compared to the requirements that were defined in section 2.

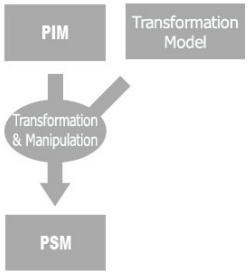

Figure 1: PIM-PSM pattern

3.1 The CoGenIVE approach

3.1.1 Process description

CoGenIVE (Code Generation for Interactive Virtual Environments) is a tool-supported process developed at the Expertise centre for Digital Media (EDM), a research lab at Hasselt University. The tool has been created in order to support and evaluate a model-based development process (depicted in Fig. 2a), to facilitate the creation of multimodal IVEs. See [6] for more details.

The first explicit artifact in the development process is a Task Model, expressed in ConcurTaskTrees (CTT) [10]. This widely used notation uses a graphical syntax and offers both a hierarchical structure and support to specify temporal relations between tasks. Four types of tasks are supported in the CTT notation: application tasks, user tasks, interaction tasks, abstract tasks. Sibling tasks on the same level in the hierarchy of decomposition can be connected by temporal operators.

Once the Task Model is created it will be converted to a Dialog Model, denoted as a State Transition Network (STN). The STN is based upon Enabled Task Sets (ETS), which can be automatically generated from the CTT by means of the algorithm described by Luyten *et al*. [9]. Each ETS consists in the tasks that can be executed in a specific application state. Since we are developing *Interactive* Virtual Environments, we will only elaborate on the interaction tasks within the application. A possible example of such a task is object selection. In a typical form-based desktop application, selecting an item is a straightforward task in which the user selects an entry from a list. In an IVE however, this task becomes more complex because of the wide variety of selection metaphors (e.g. touch selection, ray selection, go-go selection).

Figure 2: CoGenIVE (a) and Veggie (b) approaches

To handle this problem, an Interaction Description Model, called NiMMiT, has been created. The goal of NiMMiT is to describe an interaction task in more detail. The diagrams can be created by means of the CoGenIVE tool and are exported to an XML file which is loaded at runtime. The connection of the diagrams to the interaction tasks in the dialog model is currently done by hand. A more detailed description of NiMMiT, together with some examples, can be found in [13].

Within CoGenIVE the user can create user interface elements such as dialogs, menus, and toolbars, that are then expressed in a VRIXML presentation model. VRIXML is an XML-based user interface description language (UIDL), so that the resulting resources are loaded into the application at runtime as well. VRIXML examples and a motivation for the creation of this UIDL can be found in [5]. Like the interaction descriptions, the user interface elements should be connected to the different application states manually.

Once all models have been created and connected they are used to automatically generate a prototype of the IVE together with the external resource files in which the NiMMiT and VRIXML descriptions are stored. This approach offers some extra flexibility since the interaction techniques and the interface widgets) can be altered without regenerating the code of the virtual environment.

3.1.2 Process evaluation

CoGenIVE covers several of the requirements that have been found in section 2. The process starts from a task-model and incrementally evolves towards the final user interface. The first increment (towards the dialog model) can be done by an automatic transformation. Afterwards the designer has to manually connect the presentation and the interaction description model. Preservation of manual changes in conserved only in the second

transformation, resulting in possible inconsistencies between models that are manually adapted. However; once the code is generated from the designed models, manual adaptations are tracked and saved. This way, when regenerating the application prototype, the manually inserted code is preserved.
Preliminary evaluation of the CoGenIVE process in some IVE realizations has shown a considerable reduction of development time. Currently we are working on a new version of CoGenIVE with improved and more integrated tool-support.

3.2 The Veggie approach

3.2.1 Process description

Veggie (Virtual reality Evaluator providing Guidance based on Guidelines for Interacting with End-users) is a project developed at the Belgian Lab of Human Computer Interaction (BCHI), a research lab at University catholic of Louvain. A transformational method for developing 3D user interfaces of interactive information systems was presented (Figure 2b) [8].

The method relies on the Cameleon reference framework [2], which structures the development life cycle of multi-target UIs according to four layers: (i) the *Final UI* (FUI) is the operational UI, i.e. any UI running on a particular computing platform either by interpretation (e.g., through a Web browser) or by execution (e.g., after the compilation of code in an interactive development environment); (ii) the *Concrete UI* (CUI) expresses any FUI independently of any term related to a peculiar rendering engine, that is independently of any markup or programming language; (iii) the *Abstract UI* (UI) expresses any CUI independently of any interaction modality (e.g., graphical, vocal, tactile); and (iv) the Task & Concept level, which describes the various interactive tasks to be carried out by the end user and the domain objects that are manipulated by these tasks Models are uniformly expressed in the same UIDL, which is selected to be UsiXML (User Interface eXtensible Markup Language – www.usixml.org [14]). Any other UIDL could be used equally provided that the used concepts are also supported. For instance, other UIDLs in this area are VRIMXL [5], SSIML/AR [15], and DAC [1].

The method starts from a task model and a domain model to progressively derive a final user interface. This method consists of three steps (depicted in Fig. 2b): deriving one or many abstract user interfaces from a task model and a domain model, deriving one or many concrete user interfaces from each abstract interface, and producing the code of the final user interfaces corresponding to each concrete interface. To ensure the two first steps, transformations are encoded as graph transformations performed on the involved models expressed in their graph equivalent. In addition, a graph grammar gathers relevant graph transformations for accomplishing the sub-steps involved in each step.

Once a concrete user interface is resulting from these two first steps, it is converted in a development environment for 3D user interfaces where it can be edited for fine tuning and personalization. From this environment, the user interface code is automatically generated. By expressing the steps of the method through transformations between models, the method adheres to Model-Driven Engineering paradigm where models and transformations are explicitly defined and used.

3.2.2 Process evaluation

Veggie covered just some of the requirements identified in section 2. In particular there still is a lack to describe the dialog and interaction models. Also, the graphical support is only partly covered. Similarly the set of rules to go from Abstract to Concrete model is reduced. When modified manually there is no support to track changes, resulting in possible inconsistencies between models.

On the other hand the process covers the rest of the requirements automatically and manually. The automatic process is supported for the transformations from task/domain model until concrete model. Then manually, the concrete UI is edited in a high level editor which supports automatic code generation. The feasibility of the process has been tested through case studies [8].

4. OPEN ISSUES

The challenges to have a framework to support all the above requirements are considerable. From a technological point of view it involves an integration of technologies to support the complete process. A transformation engine to support the transformational approach, high-level editors to support the design of each model, a change tracking system (reverse engineering process) to identify changes in dependent models are also beneficial in a mature model-based approach.

From a methodological point of view on the other hand, there are quite some *open issues* for which the solution is not straightforward.

Traditional task models (such as the CTT) lack the ability to describe real 3D tasks such as selection or object manipulation. A first glance solution is to expand the task model so as to reflect 3D tasks [8] with a taxonomy of primitives. Another suggested solution is to create another starting model, such as an interaction description model (such as the NiMMiT notation [13]). An important question related to this issue is: 'when should a designer switch from the task model to the interaction description model?'

As more and more IVEs are multi-user environments, possibly supporting collaboration between users, task models and interaction descriptions should allow for specification of cooperative activities. Further research is needed to judge to what extent Cooperative ConcurTaskTrees and (an extended version of) NiMMiT can do the job, and when other task and interaction descriptions come into play. In addition, this poses the constraint of representing 2D vs 3D. tasks working on 2D vs. 3D objects, especially in augmented reality, where both could be combined on top of real world objects. [15] represents a first attempt towards this direction. [14] also refers to some effort in augmented reality for combining 2D traditional widgets with 3D objects.

Another question is related to the FUI. What is the appropriate representation of 3D UIs? Should the 2D desktop metaphor still be used or are there alternative visualizations or metaphors? Several attempts go towards defining a new toolkit of 3D objects [1,15] which are natively appropriate to 3D applications. Again, this represents an advantage to have a predefined collection of such 3D-widgets, but then the interaction is reduced by what they offer natively. Expanding already existing widgets or introducing custom widgets remains a very challenging issue.

A final issue we would like to address in this paper concerns the mapping rules. This is one of the most complex tasks in an MDE approach. A problem such as: 'how to define spatial positions to place 3D UI elements or objects', is not easy to automate due to the lack of semantic properties that define these spatial relations. Maybe the use of ontologies can be of any help to solve this issue.

5. CONCLUSION

This paper introduced a series of requirements for enabling model-driven engineering of 3D multimodal UIs, an area which is recognized for being challenging MDA.

In general, model transformation holds the promise that each step could be achieved by applying a limited set of transformations, which are declaratively stated. It is a fundamental research problem to assess that the declarative power of such transformations at least equal the procedural capabilities of algorithms traditionally used to produce a final UI. On the one hand, such algorithms could be very powerful, but do not preserve properties like observability, controllability, and traceability. On the other hand, algorithms could probably produce a final code which is hardly attainable by model transformation. Moreover, the multiplication of transformations in a same transformation set is complexifying the problems of transformation dependence, sequence, and organization. This is a potential reason why mixed-model-based approaches could be also attempted. In this case, the advantages of both paradigms could be combined without suffering from their drawbacks.

6. ACKNOWLEDGEMENTS

Part of the EDM research is funded by EFRO (European Fund for Regional Development), the Flemish government and the Flemish Interdisciplinary institute for Broadband Technology (IBBT). The VR-DeMo project (IWT 030248) is directly funded by the IWT, a Flemish subsidy organization. We gratefully thank the support from the SIMILAR network of excellence (The European research taskforce creating human-machine interfaces SIMILAR to human-human communication), supported by the 6th Framework Program of the European Commission, the Alban program supported by European Commission and the CONACYT program supported by the Mexican government.

7. REFERENCES

[1] Andujar, C., Fairén, M., and Argelaguet, F. A Cost Effective Approach for Developing Application-Control GUIs for Virtual Environments. In *Proc. of the 1st IEEE Symposium of 3D User Interfaces* 3DUI'2006 (Alexandria, March 25-26, 2006)..IEEE Comp. Society Press, 2006, pp. 45-52.

[2] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. *Interacting with Computers*, Vol. 15, No. 3, 2003, pp. 289-308.

[3] Clerckx, T., Luyten, K., and Coninx, K. Dynamo-AID: A Design Process and a Runtime Architecture for Dynamic Model-based User Interface Development. In *Proc. of 9th IFIP Working Conf. on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCI-DSVIS'2004* (Hamburg, July 11-13, 2004). LNCS, Vol. 3425, Springer-Verlag, Berlin, 2005, pp. 77-95.

[4] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *Proc. of Conf. on-Virtual Reality VR'2001* (Yokohama, 13-17 March 2001), IEEE Comp. Society Press, Los Alamitos, 2001, pp. 89-96.

[5] Cuppens, E., Raymaekers, C., and Coninx, K. VRIXML: A user interface description language for virtual environments. In *Proc. of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages"* (Gallipoli, May 25, 2004), Gallipoli, 2004, pp. 111–117.

[6] Cuppens, E., Raymaekers, C., and Coninx, K. A Model-Based Design Process for Interactive Virtual Environments. In *Proceedings of 12th Int. Workshop on Design, Specification and Verification of Interactive Systems DSVIS'05*, (Newcastle upon Tyne, July 13-15, 2005). LNCS, Vol. 3941, Springer-Verlag, Berlin, pp. 239-250.

[7] Fencott, C. Towards a Design Methodology for Virtual Environments. In *Proc. of Workshop on User Centered Design and Implementation of Virtual Environments* UCDIVE'99 (University of York, 30 September 1999).

[8] Gonzalez, J.M., Vanderdonckt, J., and Arteaga, J.M. A Method for Developing 3D User Interfaces of Information Systems. In *Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006* (Bucharest, 6-8 June 2006), Springer-Verlag, Berlin, 2006, pp. 85-100..

[9] Luyten, K., Clerckx, T., Coninx, K., Vanderdonckt, J. Derivation of a Dialog Model from a Task Model by Activity Chain Extraction. In *Proc. of 10th Int. Conf. on Design, Specification, and Verification of Interactive Systems DSV-IS'2003* (Madeira, June 4-6, 2003), LNCS, Vol. 2844, Springer-Verlag, Berlin, 2003, pp. 203-217.

[10] Mori, G., Paternò, F., and Santoro, C. Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. *IEEE Transactions on Software Engineering*, Vol. 30, No. 8, August 2004, pp. 1-14.

[11] Neale, H. and Nichols, S. Designing and developing Virtual Environments: Methods and Applications. In *Visualization and Virtual Environments Community Club VVECC'2001*, Designing of Virtual Environments. 2001.

[12] Miller, J. and Mukerji, J. *MDA Guide Version 1.0.1*, Document Number: omg/2003-06-01, OMG, June 12th, 2003, accessible at http://www.omg.org/docs/omg/03-06-01.pdf

[13] Vanacken, D., De Boeck, J., Raymaekers, Ch., and Coninx, K. NiMMiT: A Notation for Modeling Multimodal Interaction Techniques. In *Proceedings of the Int. Conf. on Computer Graphics Theory and Applications GRAPP'2006* (Setúbal, February 25-28, 2006).

[14] Vanderdonckt, J. A MDA-Compliant Environment for Developing User Interfaces of Information Systems. In *Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05* (Porto, 13-17 June 2005), LNCS, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31.

[15] Vitzthum, A. SSIML/AR: A Visual Language for the Abstract Specification of Augmented Reality User Interfaces. In *Proceedings of the IEEE Virtual Reality Conference VR'2006* (March 25-29, 2006). IEEE Computer Society Press, Los Alamitos, 2006, pp. 135-142.

[16] Willans, J. and Harrison, M.D. A Toolset Supported Approach for Designing and Testing Virtual Environment Interaction Techniques. *International Journal of Human-Computer Studies*, Vol. 55, No. 2, August 2001, pp. 45-165.