Chapter 7

A METHOD FOR DEVELOPING 3D USER INTERFACES OF INFORMATION SYSTEMS

Juan Manuel González Calleros[1], Jean Vanderdonckt[1], and
Jaime Muñoz Arteaga[2]
[1] *School of Management (IAG), Université catholique de Louvain*
Place des Doyens, 1 – B-1348 Louvain-la-Neuve (Belgium)
E-mail: {gonzalez, vanderdonckt}@isys.ucl.ac.be – Web: http://www.isys.ucl.ac.be/bchi
Tel: +32 10 47{8349, 8525} – Fax: +32 10 478324
[2]*Universidad Autonóma de Aguascalientes, Dpto. de Sistemas de Información*
Av. Universidad # 940 – C.P. 20100 Aguascalientes (México).
E-Mail: jmunozar@correo.uaa.mx – Web: http://148.211.40.92:8080/jaime/index.htm

Abstract A transformational method for developing tri-dimensional user interfaces of interactive information systems is presented that starts from a task model and a domain model to progressively derive a final user interface. This method consists of three steps: 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. The method is defined by its steps, input/output, and exemplified on a case study. 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.

Keywords: 3D user interfaces, Model driven engineering, Scene model, Transformational approach, Virtual reality, World model.

1. INTRODUCTION

Today, the development life cycle of 3D User Interfaces (UIs) mostly remains an art more than a principled-based approach. Several methods [1,3,8,9,15] have been introduced to decompose this life cycle into steps and sub-steps, but these methods rarely provide the design knowledge that should be typically used for achieving each step. In addition, the development life cycle is more focusing directly on the programming issues than on the design and analysis phases. This is sometimes reinforced by the fact that available tools for 3D UIs are toolkits, interface builders, rendering engines, etc. When there is such a development life cycle defined, it is typically structured into the following set of activities:

1. The **conceptual phase** is characterized by the identification of the content and interaction requests. The meta-author discusses with the interface designer to take advantage of the current interaction technology. The interface designer receives information about the content. The result of this phase is the production of UI schemes (e.g., written sentences, visual schemes on paper) for defining classes of interactive experiences (e.g. class Guided tour). Conceptual schemes are produced both for the final users and the authors. The meta-author has a deep knowledge of the content domain and didactic skills too. He/she communicates with the final user too, in order to focus on didactic aspects of interaction.
2. In the **implementation phase**, the UI designer builds the final user interface and the author interface on the basis of the UI schemes. The results of this phase are available as tools for the authors, which can be manipulated without a deep knowledge of computer science world. It is important to note that this implementation phase can be a personalization or a subsetting of existing tools, rather than a development from scratch.
3. In the **content development phase**, authors choose among the available classes of interactive experiences and instantiate the one that fits their particular needs (e.g. a guided tour, paths). They take advantage of a number of complementary subjects: editors (e.g., writer, 2D graphic artist), 3D modeler, and world builder.
4. In the **final user interaction phase**, the final user interacts with the contents of the 3D world, composed by the author, through the interface implemented by the interface designer. The final user interaction is monitored in order to improve both the usability of the interface and the effectiveness of content communication.

As opposed to a content-centric approach, some other authors advocate a user-centered approach; hence, involvement of users in the requirements analysis and evaluation are essential for achieving a usable product. They also argue for separating the conceptual part from the rest of the life cycle to

identify and manage the Computing-Independent Models (CIM as defined in the Model-Driven Engineering –MDE) from the Computing-Dependent part. This part is in turn typically decomposed into issues that are relevant only to one particular development environment (Platform-Specific Models –PSM) as opposed to those issues which remain independent from any underlying software (Platform-Independent Models–PIM). In the MDE paradigm promoted by the Object Management Group (www.omg.org), it is expected that any development method is able to apply this principle of separation of concerns, is able to capture various aspects of the problem through models, and is capable of progressing moving from the abstract models (CIM and PIM) to the more concrete models (PSM and final code). The goal of this paper is to demonstrate the feasibility of a MDE-compliant method that is user-centered as opposed to contents-centric for developing 3D UIs.

The remainder of this paper is structured as follows: Section 2 summarizes related work, Section 3 outlines the general method and progressively explains all steps of the method based on models. Section 4 concludes the paper and presents some avenue for future work.

2. RELATED WORK

Different categories of software exist to support the rendering of 3D UIs ranging from the physical level to the logical level. At the lowest level are located APIs such as OpenGL, Direct3D, Glide, and QuickDraw3D, which provide the primitives for producing 3D objects and behaviors. They offer a set of powerful primitives for creating, manipulating 3D objects, but these primitives are located at a level that does not allow any straightforward use for rendering higher level widgets. Several 3D desktop replacements for Microsoft Windows XP exist such as Microsoft Task Gallery (http://research.microsoft.com/adapt/TaskGallery/), the Infinite3D Cube (http://www.infinite-3d.com/), SphereXP (http://www.hamar.sk/sphere/) which is taking the known concept of three-dimensional desktops to its own level. It offers a new way to organize objects on the desktop such as icons and applications. SphereXP, like other similar environments, are usually limited to presenting existing interactive applications and their UIs in a flat 2D way, even if they are working in a 3D world (Fig. 1). Similarly, SUN has initiated the Looking Glass Project (http://wwws.sun.com/software/looking_glass/index.html) as a 3D desktop environment for Linux workstations. These environments are very powerful for their manipulation of windows in 3D, but they are not intended to render 2D UIs with 3D effects. Beyond existing 3D desktop environments is Metisse [4]. It consists of an X-based window system for two purposes: it should facilitate the development of innovative window man-

agement techniques and it should conform to existing standards and be robust and efficient enough to be used on a daily basis. Metisse is not focused on a particular kind of interaction (e.g., 3D), it should be considered rather a tool for creating new desktops, including 3D. On the other hand, it is actually possible to directly implement 3D UIs on top of 3D development toolkits such as Contigra, Croquet (http://croquetproject.org/). The advantage of these environments is that true 3D widgets (e.g., a ring menu could be implemented with an appropriate presentation and behavior). However, this assumes that we have to redevelop all widgets traditionally found in 2D UIs (e.g., a list box, a drop-down list) in these environments and that 3D containers are required to gather them, as windows play the role of containers for 2D widgets. RealPlaces (http://www-3.ibm.com/ibm/easy/eou_ext.nsf/publish/84) is a particular case where all office 3D widgets are already predefined with their behavior, but they cannot be changed or they do not necessarily correspond to their 2D counterpart.

Another series of approaches is often referred to as a model-based one [15] as they exploit specifications of the widgets, of the UI or of the complete scene to automatically generate VRML97 or X3D code of these UIs. The underlying model is frequently expressed in a XML-compliant language as the syntax of such a language is nowadays very widespread. Typical examples of such approaches include InTml (http://www.cs.ualberta.ca/~pfiguero/InTml/Introduction.html), VRIXML [6], and Flatland (based on 3dml, see http://mr.3dml.free.fr/).

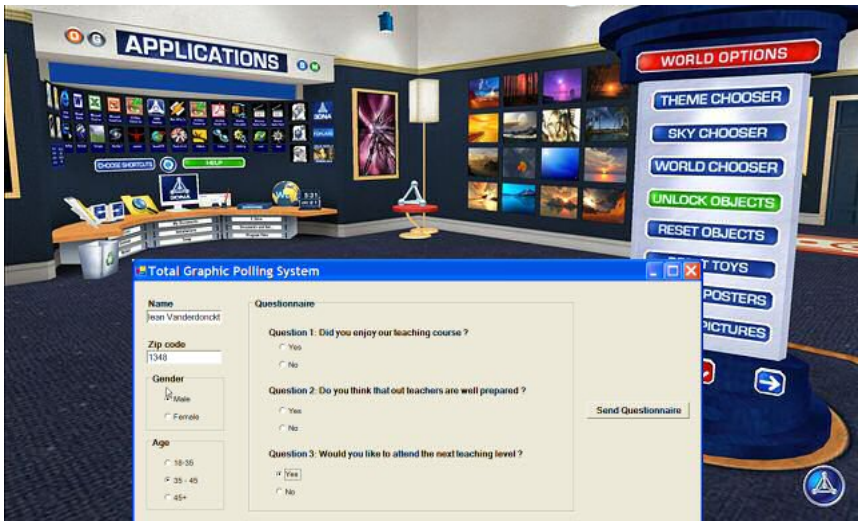


Figure 1. Flat rendering of a 2D window in a 3D world (Source: Windows 3DNA environment)

From these existing environments, we can observe that most of them are more oriented towards facilitating the life of the developer, but do not necessarily address the concerns of the designer and often forget the user requirements. It is not their purpose to provide designers and analysts with a complete environment that support them throughout the development life cycle. Therefore, such environments could be considered as software that could be complemented by design tools supporting more the conceptual phase than the development phase. In addition, they do not offer many choices in exploring design options and design alternatives during the design phase. These environments are usually restricted to one programming or markup language and do not allow easy porting code from one platform to another.

3. METHOD OUTLINE

To address the aforementioned shortcomings, a method is now introduced that structures the development life cycle for 3DUIs from the conceptual phase to the final user interaction phase by incorporating explicitly user's requirements from the beginning. Since the method should be compliant with MDE and its principle of separation of concerns, the method (Fig. 2) is itself decomposed into a sequence of four steps. Each following subsection is dedicated to the definition, the discussion, and the exemplification of these steps on a running example: a virtual polling system for which different versions will be obtained.

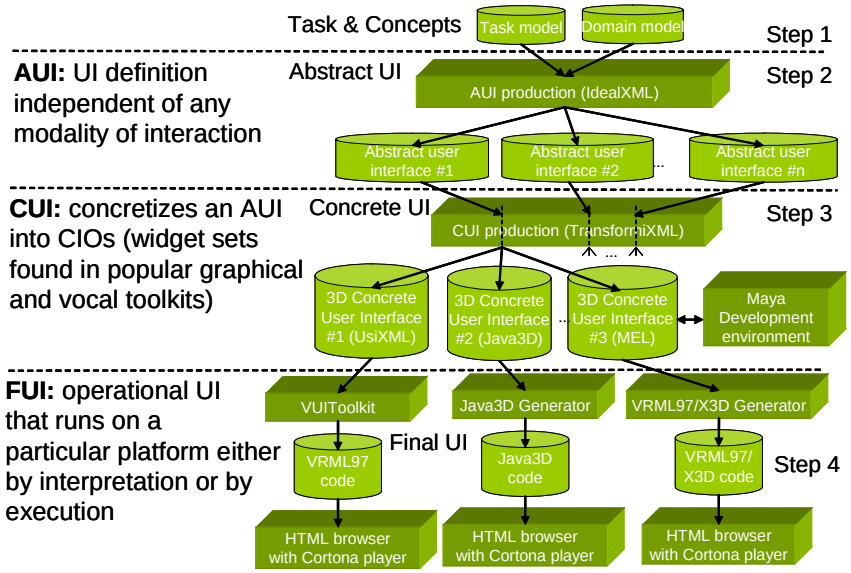


Figure 2. Outline of the method for developing 3D user interfaces.

3.1 Reference Framework for Multi-target UIs

Prior to defining the concepts on which the rest of this paper will rely, we assume to rely on the Cameleon 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. We refer to [3] for more details and to [12] for its translation into models uniformly expressed in the same User Interface Description Language (UIDL), which is selected to be UsiXML, which stands for User Interface eXtensible Markup Language (http://www.usixml.org). Any other UIDL could be used equally provided that the used concepts are also supported. The Context of use describes all the entities that may influence how the user's task is carrying out with the future UI. It takes into account three relevant aspects, each aspect having its own associated attributes contained in a separate model: user type (e.g., system experience, task experience, task motivation), computing platform type (e.g., mobile platform vs. stationary one), and physical environment type (e.g., office conditions, outdoor conditions).

3.2 Step 1: The Task and Domain Models

The task model, the domain model, and the mappings between, are all graphically described using IdealXML tool [14], an Interface Development Environment for Applications specified in UsiXML. Fig. 3 depicts the domain model of our UI as produced by a software engineer. A participant participates to a questionnaire. A questionnaire is made of several questions. A question is attached to a series of answers. The domain model has the appearance of a class diagram. Fig. 3 illustrates a CTT representation of the task model envisioned for the future system. The root task consists of participating to an opinion poll. In order to do this, the user has to provide the system with personal data. After that, the user iteratively answers some questions. Answering a question is composed of a system task showing the title of the question and of an interactive task consisting in selecting one answer among several proposed ones. Once the questions are answered, the ques-

tionnaire is sent back to its initiator. All temporal relationships are enabling which means that the source task has to terminate before the target task can be initiated.

The dashed arrows between the two models in Fig. 3 depict the model mappings, such as manipulates relationships between the task and the domain model as dashed arrows. Provide Personal Data is mapped onto Participant class. Show Question is mapped onto the attribute title of class Question. The task Select Answer is mapped onto the attribute title of the class Answer. Finally, the task Send Questionnaire is mapped onto the method sendQuestionnaire of the class Questionnaire**.** The initial task may be considered as not precise enough to perform transformations. Indeed, the task Provide Personal Data is an interactive task consisting in creating instances of Participant. In reality, this task will consist in providing a value for each attribute of Participant. This could mean that the task model is not detailed up to the required level of decomposition.

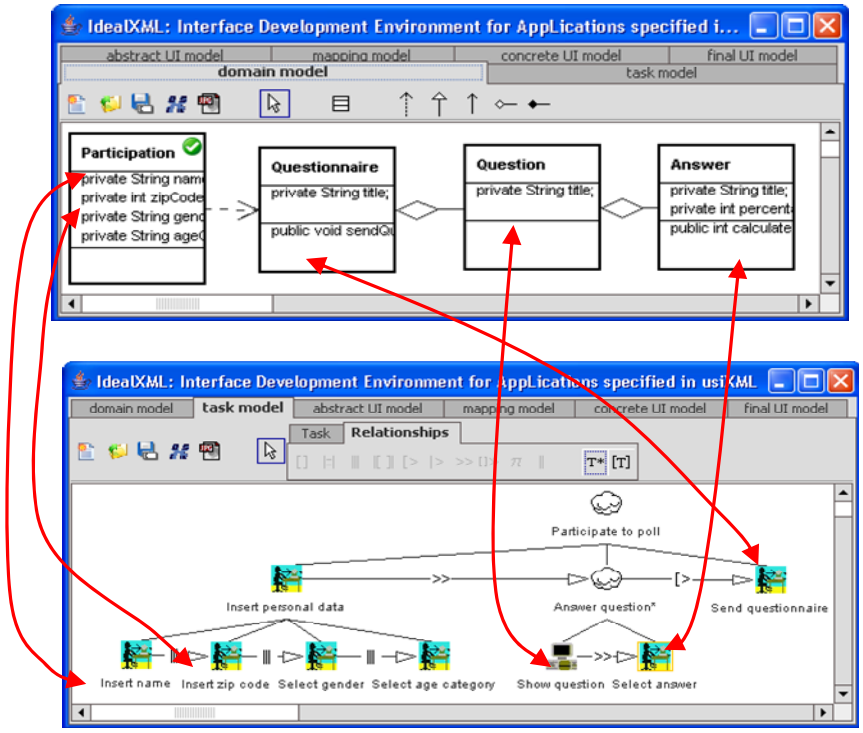
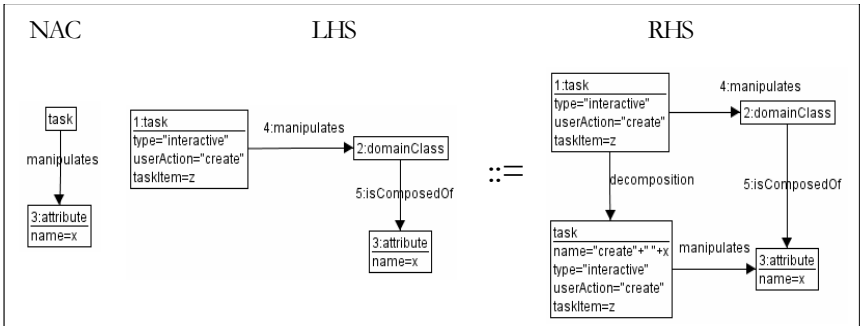


Figure 3. Process to create 3D user interfaces.

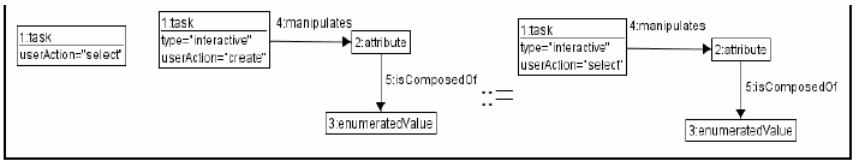
Rule 1 is applied to the task and domain models. The Left-Hand Side (LHS) contains an **interactive** task (1) where the user action required to perform the task is of type **create**. This task manipulates a class from the do-

main model (2), which is composed, of an attribute that takes the value of a variable **x**. The Negative Application Condition (NAC) specifies that a task manipulates an attribute (3) whose name is stored in the same variable **x**. The Right Hand Side (RHS) specifies the decomposition of the task described in LHS (1) into an interactive task (2), which requires a user action of type **create**. Note the way they are named using a post-condition on their **name** attribute. The mappings between nodes and between edges belonging to the three components of a rule (i.e., NAC, LHS, and RHS) are specified by attached numbers. The application of this rule on the task and domain model represented in the form of a graph G is the following: when the LHS matches into G and the NAC does not match into G, the LHS is replaced by the RHS, resulting a transformed graph G'. Therefore, Rule 1 decomposes the task Provide Personal Data into four new sub-tasks, each of them manipulating an attribute of class Participant.



Rule 1. Consolidation of the task model.

Consequently, to the execution of this rule, four new tasks are created: create name, create zipCode, create ageCategory and create gender. Fig. 3 contains the mapping model containing the mappings between the refined task model and the domain model of the opinion polling system. Each of the four new sub-tasks will be mapped on the corresponding attribute of the class Participant, the rest of the mappings remaining the same. Due to the fact that "create" is a very general action type and that both ageCategory and gender attributes hold an enumerated domain, "create" can be specialized into "select". Rule 2 is applied in order to achieve this goal. Rule 3 provides a default temporal relationship (set to enabling) when two sister tasks have no temporal relationship.



Rule 2. Specializing a user action.

3.3 Step 2: From Task and Domain Models to Abstract Model

The second transformation step involves a transformation system that contains rules applied to transition from the task and domain model to the abstract model. Those rules create an abstract container (AC) for task that has task children, i.e. participate poll, insert personal data, and answer question for this example. Following the same mechanism of rule transformation, an abstract individual component (AIC) is created for every leaf task found the task model: insert name, insert zip code, select gender, select age category, show question, select answer and send questionnaire. Each AIC can be equipped with facets describing its main purpose/functionality. These facets are derived from the combination of task model, domain model and the mappings between them. Task definitions have information that is relevant for the mappings, such as: userAction, which could be: create, delete, modify, among others. According to these mappings it can be derived that AICs create name and create zipCode are equipped with an input facet of type "create attribute value". The generated abstract user interface is shown in Fig. 4, detailed description of the mapping rules applied are found in [16].

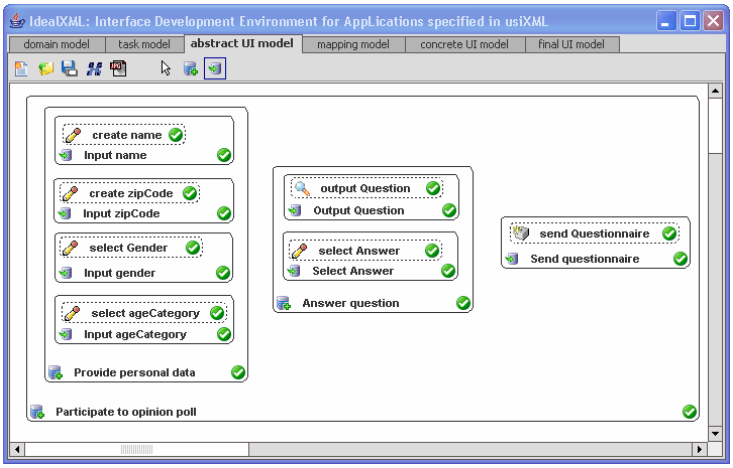


Figure 4. IdealXML Mapping from Task and Domain model to Abstract Model.

3.4 Step 3: From Abstract Model to Final User Interface

The third step implies a transformational system that is composed of necessary rules for realizing the transition from AUI to CUIs. For this purpose, other design rules could be encoded in UsiXML so as to transform the AUI into different CUI depending the options decided. Since the AUI model is a

CIM, it is supposed to remain independent of any implementation. However, when it comes to transform this AUI into a corresponding CUI or several variants of it, platform concerns come into consideration. For this purpose, several design rules exist that transform the AUI into CUIs with different design options that will then be turned into final code when generated. We need to encode components that correspond to the meta-model of 3DUI in UsiXML. All information manipulated by all sub-tasks are all gathered in one container. In the 3D space we could imagine an infinity set of objects that could be used as containers. The virtual space is the basic container for all the concrete interface objects (CIO), i.e., entities that users can perceive and/or manipulate. So we could have **2D renders** such as *Polygons*, irregular or regular, n-sized; **3D renders** such as: polyhedrons, which involves prisms, parallelepipeds, pyramids, cones, spheres; also we consider the fact that any combination of surfaces and shape could be created and function as a container. See in Figure 5 the meta-model corresponding to the definition of the environmental model, which is responsible for describing the world in which any 3D UI could be rendered.

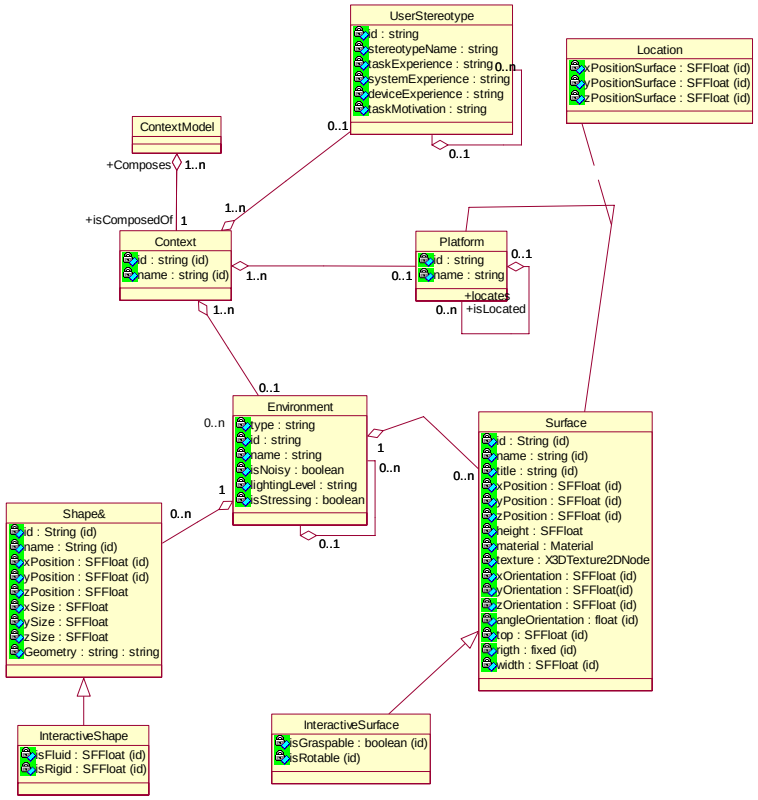


Figure 5. Environmental Model.

To adhere to the principle of separation of concerns, the model is itself decomposed into parts gathering attributes of the same area of interest. The mapping rules applied to transform the AUI specification to CUIs. In this case, CUI specifications result from the application of design rules in TransformiXML. In Fig. 6, the screenshot reproduces the two worlds generated for a Java3D environment, where each AC (provision personal data and answer question) is mapped onto one scene at a time. All AICs belonging to each AC are then mapped recursively onto Java3D widgets depending on their data type. In this particular case, the designer selected also the graphical representation if any, along with the textual representing.

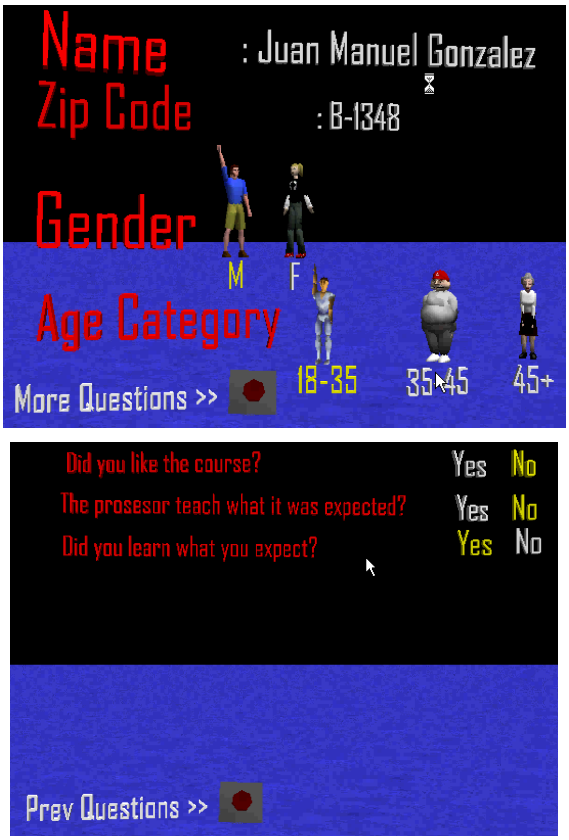


Figure 6. Java 3D representation of the polling system.

In this visualization, we propose another way to represent the category selection. Instead of using a comboBox, or the traditional view of icons attached to radio button, we proposed the use of 3D personages instead of icons. This 3D graphic representation of the option could reinforce the understanding, notice that we keep the text below the personages.

Figure 7. Polling system rendered in VRML

In Fig. 7, the decomposition of ACs is more fine-grained than in the previous cases: the information related to the person are first acquired in a rotating cube (which was selected as the container), then each pair of questions is presented at a time with the facilities of going forward or backward like a wizard. Since only 3 questions and one set of person information are considered, a cube is selected to present each part. If for any reason, more questions were defined, let us say 5, a regular volume with 6 faces would be generated instead. The description of the UI is not enough; we need an editor to manipulate the 3D objects easily with an automatic feedback of the modifications done by the user. We use for this purpose Maya, by specifying a Maya ASCII file as a result of the Abstract specification of the 3DUI. The files is opened in the Maya editor (Fig. 8) and finally exported in a target markup or programming language for virtual reality. Maya plug-ins offers, among others exporters, RawKee (http://rawkee.sourceforge.net/), an open source (LGPL) X3D plug-in, that exports Maya's 3D data as an X3D or VRML file with support for scripting. Fig. 9 reproduces some snapshots of the 3DUI rendered in VRML (Virtual Reality Markup Language).

The UsiXML specifications at the CUI could also be interpreted in VUIToolkit, a rendering engine for 3D UIs specified in UsiXML in VRML97/X3D. In the screenshot of the Fig. 10, we show the result of using the Toolkit that generates the 3D rendering of how our polling system could look in a 2D user interface. The 2D components have been enriched with volumes. One can discuss that the components are rendered as 3D widgets in a way that remains similar to the “Look & Feel” of 2D widgets, except that the “Feel” is a genuine 3D behavior. According to this view, this kind of FUI can be interpreted only as a 3D rendering of 2D UIs, even if their specifications are toolkit-independent [13]. This approach provides an option to the

use of Java applets UIs to manipulate virtual applications in the Web, instead, the use of the VUIToolkit would not disrupt the 3D "look".

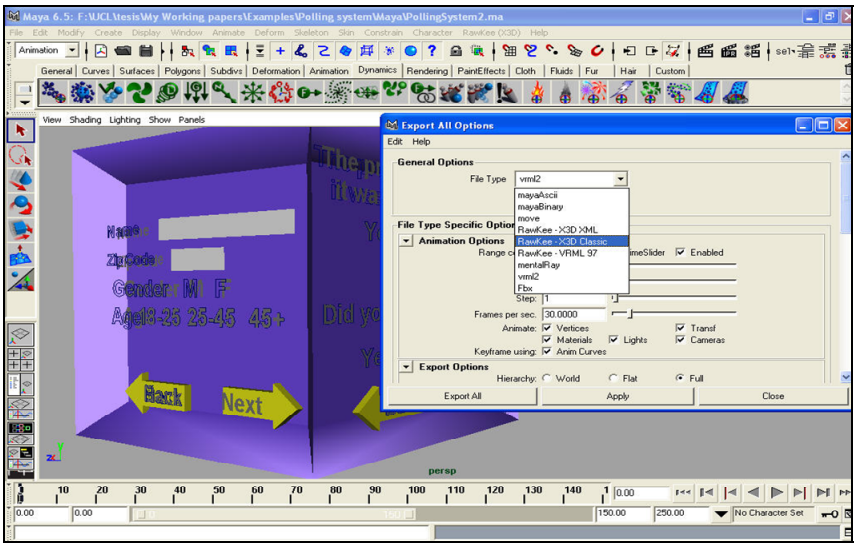


Figure 8. Edition of the 3DUI in Maya.

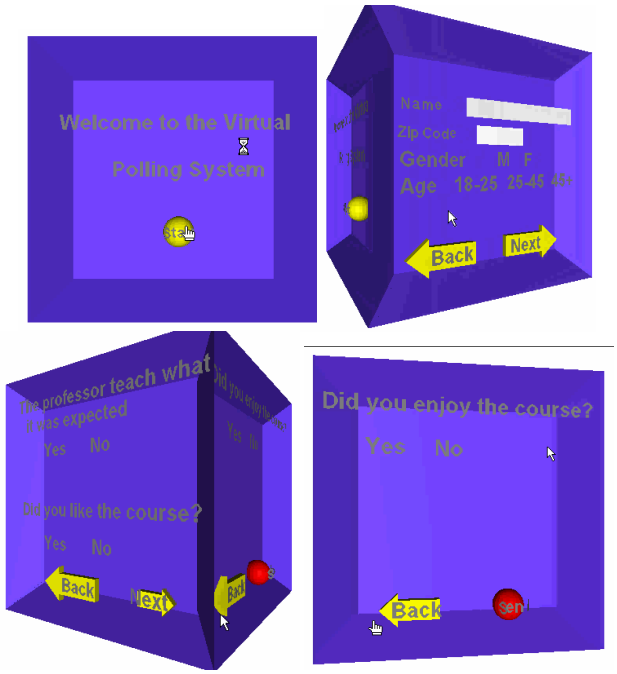


Figure 9. Rendering of the 3DUI interface for the polling system in VRML.

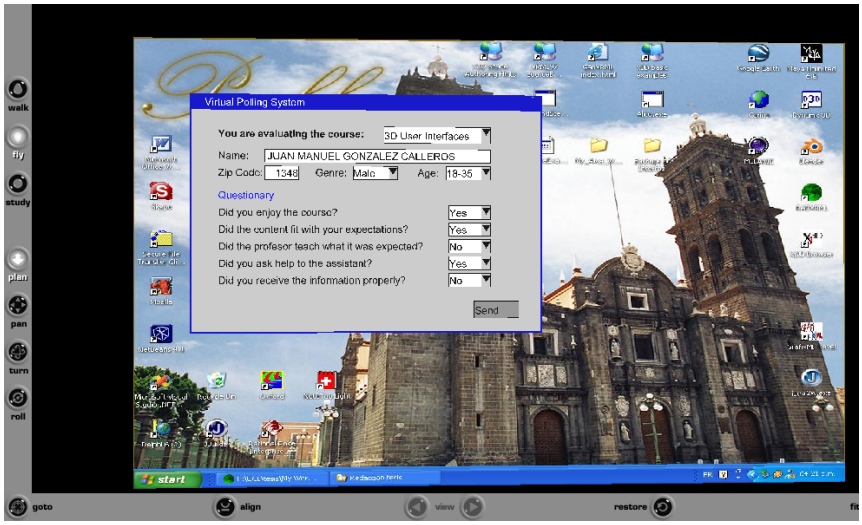


Figure 10. 3D rendering of the 2D interface for the polling system in VUIToolkit.

4. CONCLUSION

A method has been presented that decomposed the 3D UI development life cycle into four steps ranging from the most abstract (CIM) to the most concretes (PIM, then PSM) according to the principles of Model-Driven Engineering. The first step is intended to capture user requirements through a task model manipulating information contained in a domain model. The second step transforms this task model into an abstract UI model that is computing-independent. The third step supports in our case three transformations so as to obtain three types of final rendering: interpretation of the CUI UsiXML specifications in VUIToolkit (a 3D rendering engine that has been developed for this purpose), in Java3D and in VRML97/X3D.

The feasibility of the approach is much depending on the amount and the quality of the design rules that are also encoded in UsiXML. If a reasonably extensive set of rules is used, the generated results are usable. If this is not the case, the model resulting from the transformation could be considered as underspecified. It is then required to manually edit within a XML-compliant editor. Future work will therefore be dedicated to exploring more design options and encode them in UsiXML so as to serve better transformations. This does not mean that a generated 3D UI is as usable or more usable than a manually-produced one, but at least it could be obtained in a very fast way. Moreover, the exploration of alternative design options could be facilitated since they are operated at a higher level of abstraction than the code level.

ACKNOWLEDGEMENTS

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, under contract FP6-IST1-2003-507609 (http://www.similar.cc), the Alban program (www.programalban.org) supported by European Commission, and the CONACYT (www.conacyt.mx) program supported by the Mexican government. All information regarding UsiXML is accessible through http://www.usixml.org.

REFERENCES

[1] Bowman, D.A., Kruijff, E., LaViola, J., and Poupyrev, I., *3D User Interfaces: Theory and Practice*, Addison Wesley, Boston, July 2004.

[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] Celentano, A. and Pittarello, F., *A Content Centered Methodology for Authoring 3D Interactive Worlds for Cultural Heritage*, in D. Bearman, F. Garzotto (eds.), Proc. of Int. Cultural Heritage Informatics Meeting ICHIM'2001 (Milan, 3-7 September 2001), "Cultural Heritage and Technologies in the Third Millennium", Vol. 2, 2001, pp. 315-324.

[4] Chapuis, O. and Roussel, N., *Metisse is not a 3D Desktop*, in Proc. of ACM Symposium on User Interface Software and Technology UIST'2005 (Seattle, 23-26 October 2005), ACM Press, New York, 2005, pp. 13-22.

[5] Conner, D.B., Snibbe, S.S., Herndon, K.P., Robbins, D.C., Zeleznik, R.C., and van Dam, A., *Three-Dimensional Widgets*, in Proc. of the 1992 Symposium on Interactive 3D Graphics, Special Issue of Computer Graphics, ACM Press, New York, pp. 183-188.

[6] Cuppens, E., Raymaekers, Ch., and Coninx, K., *VRIXML: A User Interface Description Language for Virtual Environments*, in Proc. of the 1st ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages" UIXML'2004 (Gallipoli, May 25, 2004), LUC-EDM, 2004, pp. 111-118.

[7] Fencott, C. and Isdale, J., *Design Issues for Virtual Environments*, in Proc. of Int. Workshop on Structured Design of Virtual Environments and 3D-Components at the Web3D'2001 Conference (Paderborn, 19 February 2001).

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

[9] Geiger, C., Paelke, V., Reimann, C., and Rosenbach, W., *Structured Design of Interactive Virtual and Augmented Reality Content*, in Proc. of Int. Workshop on Structured Design of Virtual Environments and 3D-Components at the Web3D'2001 Conference (Paderborn, 19 February 2001).

[10] Katsurada, K., Nakamura, Y., Yamada, H., and Nitta, T., *XISL: A Language for Describing Multimodal Interaction Scenarios*, in Proc. of 5th Int. Conf. on Multimodal Interfaces ICMI'2003 (Vancouver, 5-7 Nov. 2003), ACM Press, New York, 2003, pp. 281-284.

[11] Larimer, D. and Bowman, D., *VEWL: A Framework for Building a Windowing Interface in a Virtual Environment*, in Proc. of IFIP TC13 Int. Conf. on Human-Computer Interaction Interact'2003 (Zürich, 1-5 Sept. 2003), IOS Press, Amsterdam, 2003, pp. 809-812.

[12] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., and Lopez-Jaquero, V.,

UsiXML: a Language Supporting Multi-Path Development of User Interfaces, 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, 11-13 July 2004), Lecture Notes in Computer Science, Vol. 3425, Springer-Verlag, Berlin, pages 207-228, 2005.

[13] Molina, J.P., Vanderdonckt, J., Montero, F., and González, P., *Towards Virtualization of User Interfaces based on UsiXML*, in Proc. of the 10th Int. Conf. on 3D Web Technology Web3D'2005 (Bangor, 29 March–1 April 2005), ACM Press, New York, 2005, pp. 169-178.

[14] Montero, F., López-Jaquero, V., Vanderdonckt, J., Gonzalez, P., and Lozano, M.D., *Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML*, in Proc. of 12th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSVIS'2005 (Newcastle upon Tyne, 13-15 July 2005), Lecture Notes in Computer Science, Vol. 3941, Springer-Verlag, Berlin, 2005.

[15] Neale, H. and Nichols, S., *Designing and Developing Virtual Environments: Methods and Applications*, in Proc. of Visualization and Virtual Environments Community Club VVECC'2001 Workshop, Designing of Virtual Environments, 2001.

[16] Stanciulescu, A., Limbourg, Q., Vanderdonckt, J., Michotte, B., and Montero, F., *A Transformational Approach for Multimodal Web User Interfaces based on UsiXML*, in Proc. of 7th Int. Conf. on Multimodal Interfaces ICMI'2005 (Trento, 4-6 October 2005), ACM Press, New York, 2005, pp. 259-266.

[17] Sutcliffe, A., *Multimedia and Virtual Reality: Designing Multisensory User Interfaces*, Lawrence Erlbaum Associates, Mahwah, 2003.

[18] Waterworth, J.A. and Serra, L., *VR Management Tools: Beyond Spatial Presence*, in Proc. of ACM Conf. on Human Aspects in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993), Addison-Wesley, Reading, 1993, pp. 319-320.

[19] Zakiul, S., *Week 15 report on Project 6*, accessible at http://www.public.asu.edu/~zakiul/vrml/week15/week15.htm