A structured approach to support 3D User Interface Development

Juan Manuel González-Calleros[1]
Jean Vanderdonckt[1]
[1]Université catholique de Louvain,
Louvain School of Management
Place des Doyens, 1 – B-1348
Louvain-la-Neuve (Belgium)
{juan.m.gonzalez, jean.vanderdonckt}@uclouvain.be

Jaime Muñoz-Arteaga[2]
Sistemas de Información
Universidad Autónoma de Aguascalientes
Av. Universidad No. 940, Col. Bosques, 20100
Aguascalientes, Aguascalientes (México)
jmunozar@correo.uaa.mx

***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 present an attempt to structure an approach to support 3DUIs development by introducing a MBDUI compliant method. The development method is articulated on three axes: models and their specification language, approach, and tools that support the method based on the underlying models.**

***Keywords:* 3D user interfaces, transformational approach, virtual reality, world model, scene model, model driven engineering.**

I. INTRODUCTION

All Few works have been reported in the literature employing formalisms to describe Three-Dimensional User Interfaces (3DUI) [15]. Many reports in the literature are devoted to describe new interaction techniques or describe software and hardware settings used in specific applications, most of them presenting also user studies for experimental evaluation. To cope with the ever increasing diversity of markup languages, programming languages, toolkits and interface development environments, including 3DUIs, UsiXML (which stands for User Interface eXtensibe Markup Language) proposes a conceptual modeling for developing User Interfaces UIs [20]. The conceptual framework was created for specifying, designing, and developing 3DUIs at a level of abstraction that is higher than the level where code is merely manipulated. For this purpose, a complete environment has been created based on conceptual modeling of user interfaces of information systems structured around three axes: the models that characterize a 3DUI from the end user's viewpoint and the specification language that allows designers to specify such interfaces, the method for developing interfaces in forward engineering based on these models, and a suite of tools that support designers in applying the method based on the models. This environment is compatible with the Model Driven Architecture (MDA) recommendations in the sense that all models adhere to the principle of separation of concerns and are based on model transformation between the MDA levels. The models and the transformations of these models are all expressed in UsiXML and maintained in a model repository that can be accessed by the suite of tools. Thanks to this environment, it is possible to quickly develop and deploy a wide array of user interfaces for different computing platforms, for different interaction modalities, for different markup and programming languages, and for various contexts of use.

The remainder of this paper is structured as follows: section 2 summarizes related work, section 3 outlines the general method, Section 4 introduces a case study and progressively explains all steps of the method, and Section 5 concludes the paper and presents some avenue for future work.

II. RELATED WORK

Different categories of software exist to support the rendering of 3DUIs providing primitives for producing 3D objects and their behaviors. These primitives are located at a level that does not allow any straightforward use for rendering higher level 3D widgets [15]. 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 [9]. These environments are usually restricted to only one programming or markup language and do not allow easy porting of code from one platform to another.

Even that several methods have been introduced [2][6][7][8][15] [17] to develop 3DUIs. They decompose the software life cycle into steps and sub-steps, but these methods rarely provide the design knowledge that should be typically used for achieving each step. Our approach [9] considers the complete life-cycle software development for 3DUI following a user-centered approach. Our method also is user centered as explicitly models user's task using the task model. Also it relies on a MDA which is explicitly based on the Cameleon Reference Framework [4], which

978-0-7695-3529-6/09 $25.00 © 2009 IEEE
DOI 10.1109/ACHI.2009.14

defines UI development steps for multi-context interactive applications.

III. STRUCTURED DEVELOPMENT OF 3DUI WITH USIXML

In software engineering, specification-based (or model-driven) approach relies in the power of models to construct and reason about software systems. This approach is based on models. To generate them we need to identify the main properties of real life objects. To do so some kind of judgment is required. The goal of model-based approach, for user interface development is to propose a set of abstractions, development processes and tools enabling a engineering approach of user interface development. The characteristics of an engineering approach are its systematic (development based of rational principles), its reproducibility, its orientation towards quality criteria.

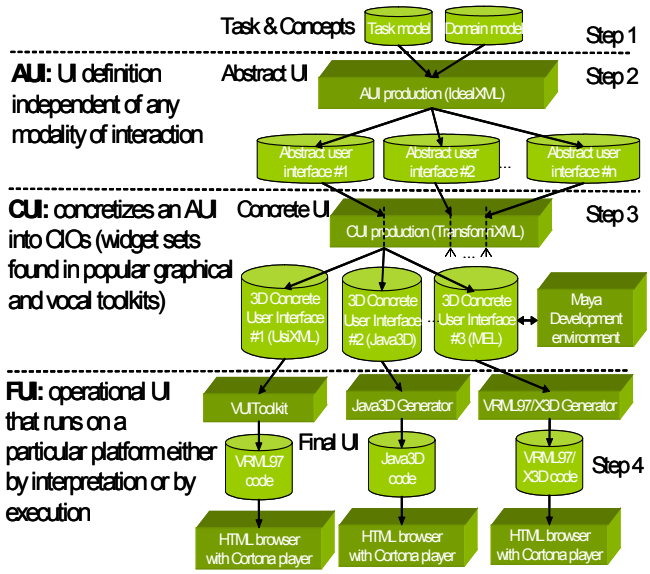


Figure 1. The Simplified User Interface Reference Framework. Source [9]

Our methodology, Figure1, structures development processes into four development steps:

1) Task & Concepts (T&C): describe the various user's tasks to be carried out and the domain-oriented concepts as they are required by these tasks to be performed.

2) Abstract UI (AUI): defines abstract containers (AC) and individual components (AIC) [11] two forms of Abstract Interaction Objects (AIO)[19] by grouping subtasks according to various criteria (e.g., task model structural patterns, cognitive load analysis, semantic relationships identification), a navigation scheme between the container and selects abstract individual component for each concept so that they are independent of any modality.

3) Concrete UI (CUI): concretizes an abstract UI for a given context of use into Concrete Interaction Objects (CIOs) [19] so as to define widgets layout and interface navigation. It abstracts a FUI into a UI definition that is independent of any computing platform. For example, in Envir3D [21], the CUI consists of a description of traditional 2D widgets with mappings to 3D by relying on different mechanisms when such a mapping is possible.

4) Final UI (FUI): is the operational 3DUI i.e. any 3DUI running on a particular computing platform either by interpretation (e.g., through a Web browser) or by execution (e.g., after compilation of code in an interactive development environment.

IV. CASE STUDY

In this section we use the student/trainer system case study to illustrate our method. The development scenario is the following: a forward engineering path is applied from a definition of the task and domain viewpoint to produce both an abstract user interface (AUI) and concrete user interface (CUI). For that purpose several are applied. For space reasons just those more relevant are shown.

The following scenario illustrates the problems and the need for such a toolkit: due to beginning of the new academic year an education institute specialized in courses for people who is working needs to calculate the number of student per trainer for the each course accordingly to different variables: number of available trainers, salary per trainer, trainer working days, annual working days for students, students salary average and number of students. So far the school has been using an excel spreadsheet however manipulating variables imply some usability disadvantages, boundaries for the variables are not visually available, also, manipulate the value of a variable has two steps type the value then press enter, any other change has to follow the same procedure. They would prefer to manipulate variables using a dedicated UI more interactive were they can select the value of the variables in a flexible way, such is the case is they were using sliders.

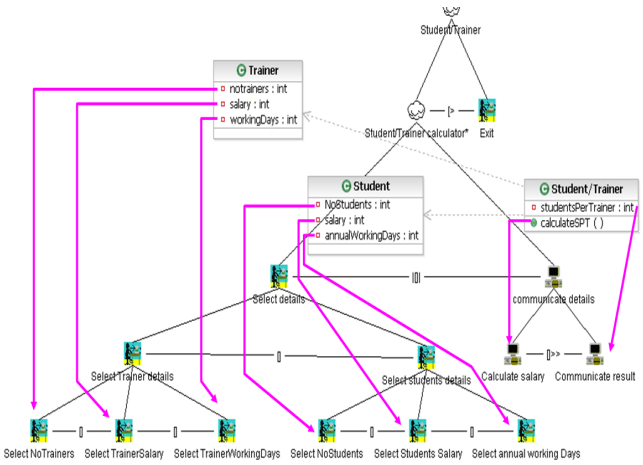


Figure 2. Task & Concepts of the student/trainer system

A. Step 1: From task to Abstract User Interface Model

The task model, the domain model Figure 2, and the mappings between, are all graphically described using IdealXML tool [13], an Interface Development Environment

for Applications specified in UsiXML. The task tree, depicted using CTT notation, shows the envisioned system. The root task consists of student/trainer is decomposed in two task, one for calculating student per trainer and the second task to exit the system at any time. Calculating student/trainer is divided in two tasks; one where the user selects the variables and the second is a system task where the result is calculated and shown to the user. The user can change any variable iteratively but any change is transferred automatically to the system task to update the result.

UsiXML transformational approach [20] is based on graph transformations. The final result of the rules applied to the task model to obtain an AUI is depicted in Figure 3.

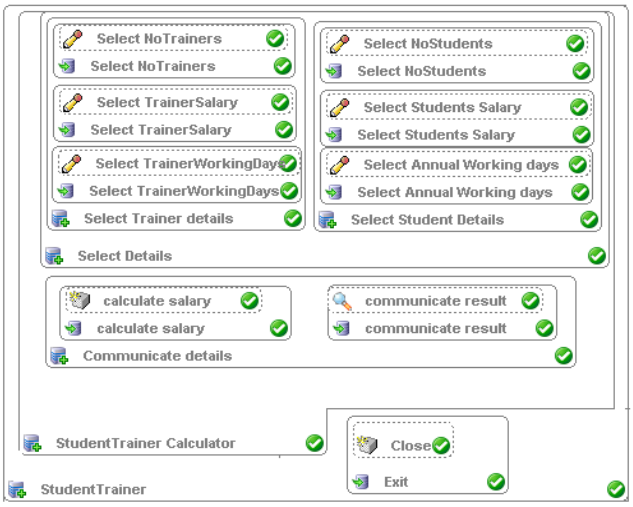


Figure 3. An Abstract User Interface of the of the student/trainer system in IdealXML [13]

In Figure 3 the big square denotes an abstract container (AC). For the example all task that are not leafs became ACs. They were obtained using the following transformation rule (Figure 4): for each task 1:Task decomposed in any other task 2:Task the Left Hand side (LHS) is the rule is matched then the right hand side (RHS) creates for the task 1:Task a relationship that tells that the task is executed in an AC. Negative Application Conditions (NAC) determine conditions to prevent the rule to apply. In this case the NAC prevent infinite loops because if a task has been already assigned with a relationship IsExecutedIn an AC then no new relationship is created.

The abstract individual components (AIC) depicted inside the big rectangles corresponds is an abstraction that allows the description of interaction objects in a way that is independent of the modality in which it will be rendered in the physical world. An AIC may be composed of multiple facets. Each facet describes a particular function an AIC may endorse in the physical world. Four main facets are identified: an input facet describes the input action supported by an AIC, for instance selecting a value from a range; an output facet describes what data may be presented to the user by an AIC, for instance, showing the summary of the bank transfer to a client; a navigation facet describes the possible container transition a particular AIC may enable, for instance, each navigation arrow in a browser has a navigation facet; and a control facet describes the links between an AIC and system functions i.e., methods from the domain model when existing. For this example task the prevalent task of the system is the selection of a value then the AIC must have an input facet ( ). The transformation rule to create this facet on each AIC (Figure 5) creates this facet in the RHS side when the rule match LHS a task that manipulates an element where the actionType attribute of the task is select. We will come back later to this attribute of the task that play an important role in further transformations.

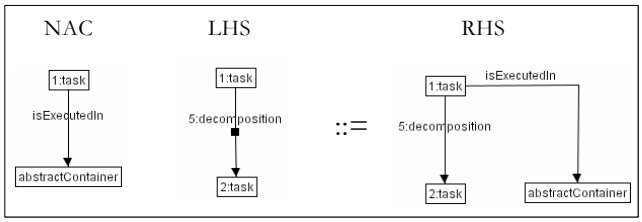


Figure 4. Create an AC for task that has task children

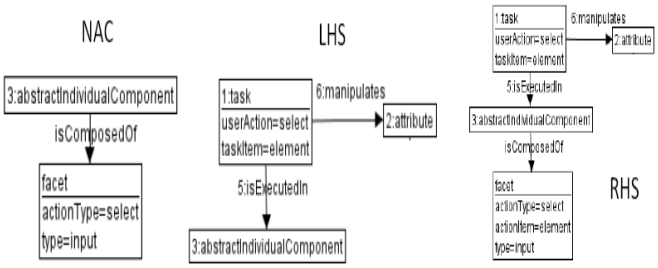


Figure 5. Create an input facet of type select

In this example no navigation facets are needed but for the system tasks AICs with control ( ) and output ( ) facet were created.

B. Step 2: From Abstract User Interface Model to Concrete User Interface Model

Before describing the next transformation the concrete user interface objects are described. Creating 3D widgets demands a lot of effort at several levels, first of all abstracting the attributes of the 3d widgets, then implementing the 3D widget and finally defines when this widget is appropriate to concretize the AIO. To illustrate this process the 3D slider is discussed in this section.

The attributes of the 3D slider are: *label*: string containing the label to be displayed on the slider; *Min*: integer value denoting the minimum value; *Max*: integer value denoting the maximum value; *Orientation*: string value denoting whether the slider is horizontal, vertical or inclined; *current value*: integer value denoting the current selected value of the slider; *shape*: string value denoting the shape of the object cylinder, sphere, cone, or any valid value depending on the implemented version of the 3D slider; *step*: integer value determining the number of steps dividing the

range of values for slider; *isVisible*: boolean attribute that determines if the slider is visible or not; *isEnable*: boolean attribute that determines if the slider is enable or not; *transparency rate*: double value indicating the transparency of the object rangin from 0.0 to 1.0; *paint Track*: boolean value indicating if the track is shown or not; *SnapToTicks*: Boolean value indicating if the selected value must move to the closest mark.

The next step consists on identifying the possible representation of the 3D widget. In Figure 17 the different variant for the slider are shown. We rely on a 3D widgets Taxonomy to determine the possible graphical representation of the slider, see Figure 6.

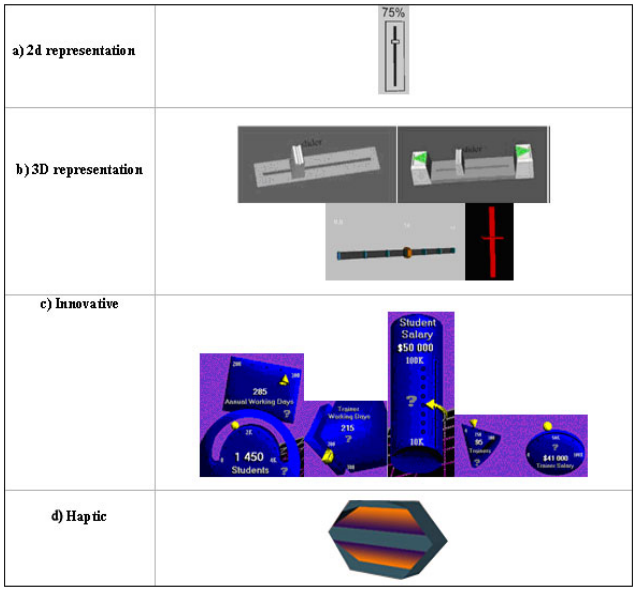


Figure 6. Three-Dimensional presentation of a slider

Then the representation of the 3D widgets was selected using the question and answer mechanism proposed in [12]. Accordingly to this method, a set of criteria can be evaluated when a question is raised and a set of possible answers exists (3D renderings in this case). In this case four criteria were identified to consider: 2D to 3D consistency, easy to develop, intuitional and the usability. The weight attributed to each criterion was based on the authors' past experience though the development of 3D and haptic applications [10].

In Figure 7 this process for the 3D slider is presented. The meaning of the links are: the darkest solid line (++) means strongly supported, dark solid line (+) means supported, solid line (~) means neutral, dash lines (-) means denied and dot lines (..) means strongly denied. In this case the question is "What will be the representation of a 3D slider?". The possible answers are four: The 2D representation, 3d representation, innovative and haptic. The evaluation shows that the if the basic criterion is to keep consistency between 2D and 3D , the 2D presentation should be chosen; in terms of complexity of the development the switch was identified as the most complicated to be developed. Regarding the intuition criterion, the sphere seems to be is the only one that could produce misunderstanding problems.

Finally, the usability of any of these presentations would not be a problem; however is considered that the haptic and the 2D presentation could be more effective as they are more intuitional and have more consistency with the 2D already known component. Notice that this analysis only provides a general view of the different properties related to the development of the 3D widgets and does not arrives at a conclusion about which representation is generally the best.

Questions	Options	Links	(Weight) Criteria
What will be the representation of a 3D Slider?	2D Representation	++	(1) 2D-3D Consistency
		~	(3) Easy to develop
		++	(4) Intuitional
		+	(5) Usability
	Innovative	+	(1) 2D-3D Consistency
		~	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	3D Representation	+	(1) 2D-3D Consistency
		-	(3) Easy to develop
		+	(4) Intuitional
		+	(5) Usability
	Haptic	--	(1) 2D-3D Consistency
		+	(3) Easy to develop
		-	(4) Intuitional
		+	(5) Usability
		+	(5) Usability

Figure 7. Criteria to select the 3D slider

The next step is the implementation of the widget. The rage of values of the slider even that rendered in 3D is just in one dimension. Depending on the orientation the corresponding axis is considered, for instance if the slider is vertical then x-axis is used. Another issue to consider is that while the knob is moved in 3D coordinates the corresponding mapping to an integer value for the real value for the slider is calculated using the formula shown in Figure 8.

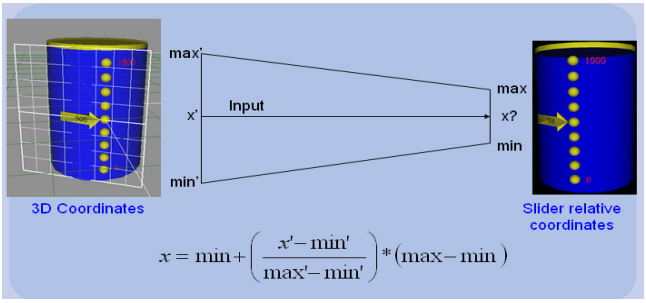


Figure 8. Implementation of the 3D slider

Another interesting problem was the track line to move along with the knob of the slider (Figure 9). It involves scaling and translating the track accordingly to the current value of the knob. Following the same principle used for the knob value (Figure 8), 3D virtual coordinate values were mapped for both scale and translation parameters of the track line. Scaling and translating for this particular example is commutative, translation after scaling and scaling after translation has the same result.

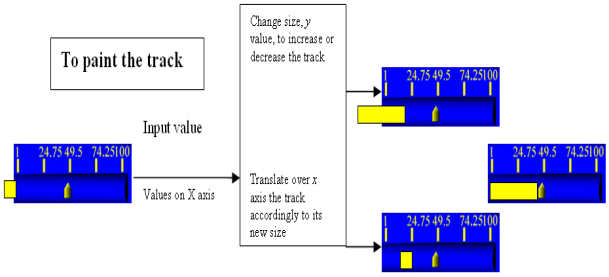
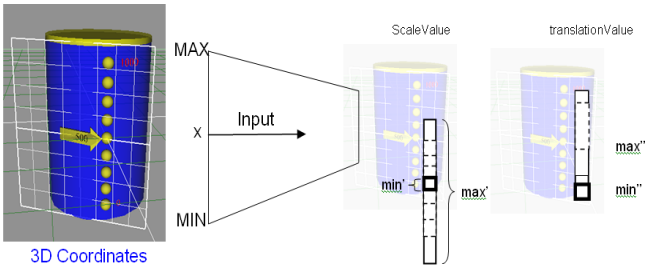


Figure 9. Track following the knob for the 3D slider



$$scaleValue = \min' + \frac{x*(\max' - \min')}{MAX - MIN} \qquad translationValue = \min'' + \frac{x*(\max'' - \min'')}{MAX - MIN}$$

Figure 10. Solution to the scale and translation for the track of the 3D slider

The next step of the transformation consists on transforming the AUI into CUI. This step is achieved by relying on a transformation tool TransformiXML [20]. The CUI can be edited in a 3D editor as shown in Figure 11. Using a 3D editor allows manipulation of 3D widgets to determine their position on the 3D scene. So, based on existing 3D widgets the concretization of the abstract interaction objects (AIOs), i.e. AC and AIC into Concrete Interaction Objects (CIOs), i.e., 3D graphical concrete interaction objects (3DGCIO) which can be 3D graphical containers (3DGC) and 3D graphical Individual Components (3DGIC).

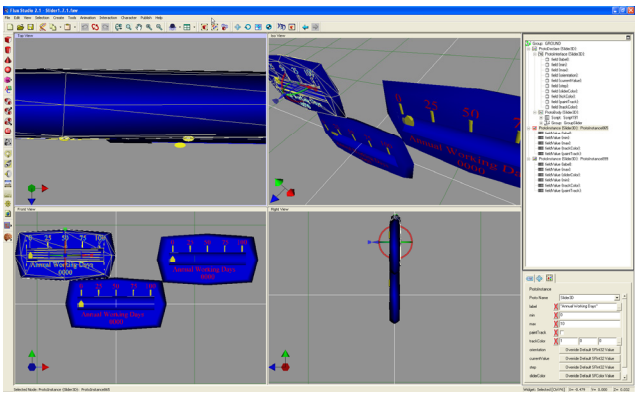


Figure 11. A C UI editor with the student/trainer system

The transformation rules used for this step use the attributes of the AUI action type and action item. In our example so far the user action is of type select and the task item is an element (Figure 5). Combining these attributes and attributes from the input facet (input card Min and max denoting a range of values) then a slider is selected for each AIC (Figure 12). The element to show the result of the calculation is transform into an output text (Figure 13).

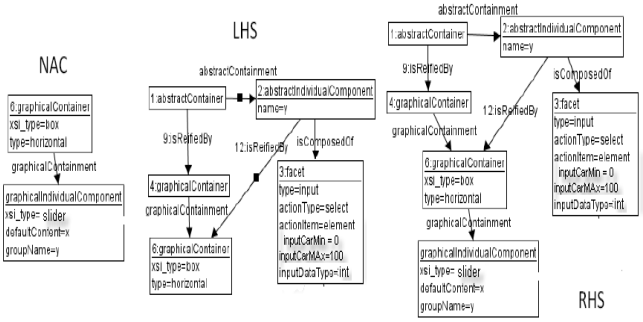


Figure 12. Mapping rule for slider selection

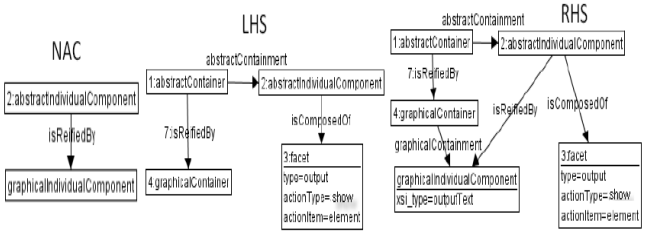


Figure 13. Mapping rule for output text selection

C. *Adding the behavior to the CIO*

Using formalisms to model interaction in virtual environments has many benefits: allows measuring the impact of changing input devices and/or interaction techniques before actually implementing them, detecting similarities and dissimilarities in the behaviors, and evaluating the effects of these dissimilarities in the prediction of user performance [15]. In our case we rely on the formal method based on Petri Nets presented in [22]. Petri net is a behaviour-based formal method, which is a promising tool for modeling concurrent systems [17]. Petri nets were selected because of their sound mathematical foundation which provides precise notations for constructing mathematical models of a target system. Using this formal specification allows reasoning about the system, either formally or informally but rigorously [22].

The Petri net model of that environment captures the state change of each object and describes the whole system. The event model (Figure 15) provides the necessary information about possible user interactions. In this case the example shows the meta-model two event sensors. The sensor is compatible with the definition promoted by the standard Extensible 3D (X3D) for cross-platform, inter-application 3D content delivery.

In our example, the Petri net (Figure 14) denote the min and max values for the correct calculation of the current value of the slider, these attributes were used for selecting the slider as a CIO in Figure 12. Each slider is denoted as an object (ovals in Figure 14) a sensor is attached to it.

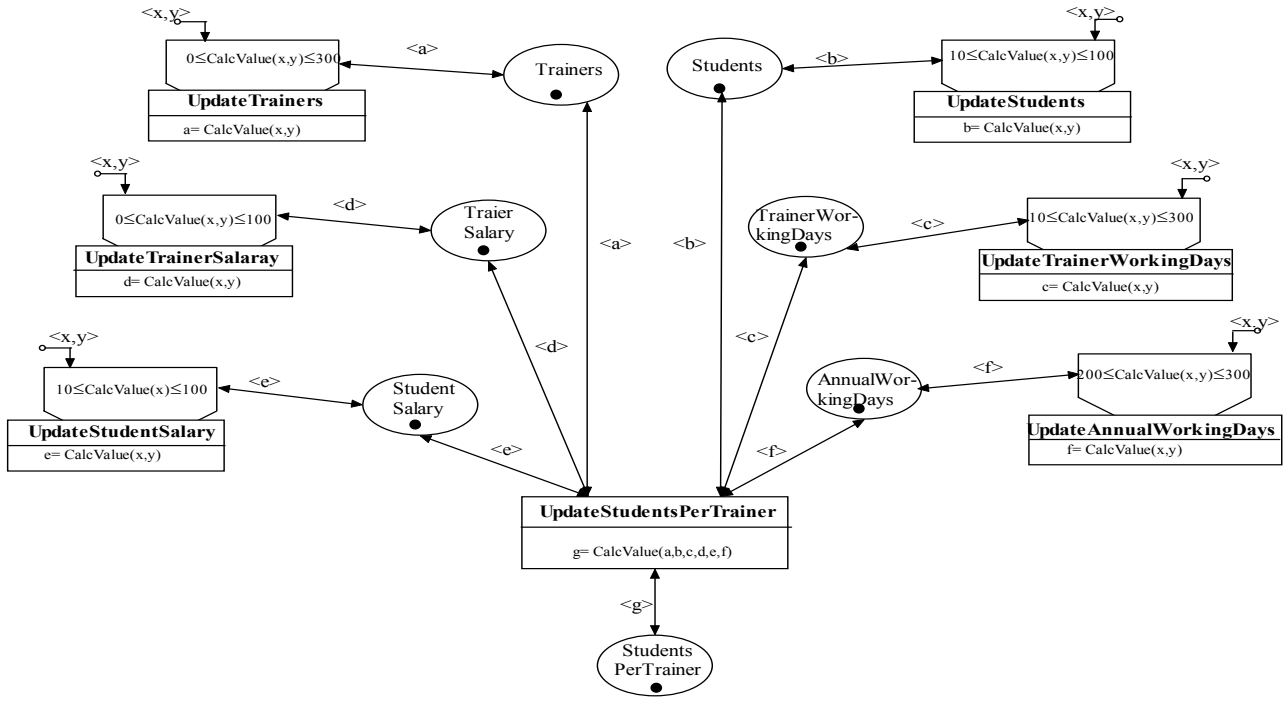


Figure 14. Dialogue on Object Oriented Petri Net for the student/trainer system

Each sensor (rectangles in Figure 14) has its own code to calculate the appropriate value depending on their min, max value. Any change on any slider will then trigger an event to the main sensor that is connected with all the sliders. This event modifies the value of the student/ trainers object.

The Petri net can be mapped into the CUI specification using the same editor, Vivaty Studio (www.vivaty.com), as the one used for editing the CUI, From the Petri net every slider has a sensor connected to it, this sensor evaluates applies when there is a translation and modifies the current value of the slider. By analogy the rest of the sensors are interconnected with the corresponding objects.

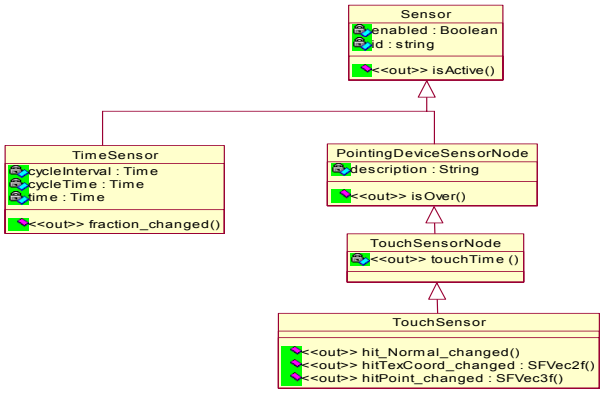


Figure 15. Event Model for the Mapping rule for output text selection

D. *Step 4: From Concrete User Interface Model to Final User Interface*

The last step is to generate the code for the 3D scene. Vivaty studio support several file type options for the final execution of the 3D world. In our case we were just interested in X3D and VRML code. The final result of our method can be seen in Figure 16.

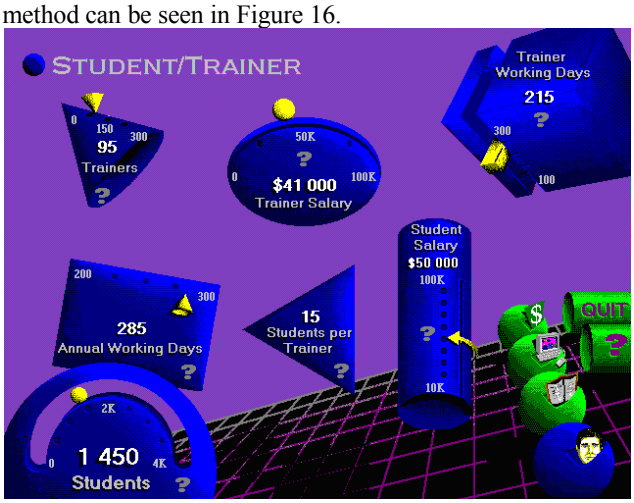


Figure 16. A final rendering of the of the student/trainer system

V. CONCLUSION

In this paper, we have introduced a 3DUI Engineering methodology articulated on three axes: models and their

specification language, method, and tools that support the method based on the underlying models. All aspects are stored in UsiXML (www.usixml.org) files that can be exchanged, shared, and communicated between stakeholders (designers, developers, and end users). It has been demonstrated that the global methodology adheres to the principles of MDA and is therefore compliant, except for the standardization process which is ongoing.

The advantages of using this method are that supports: Modifiability: If there is a change in a model then the 3DUI changes accordingly; Complexity: As it provides ways to address complexity, huge quantity of code, as well as the reliability; Safety Critical: to warranty and investigate 3DUI's behavior, models are needed. The use of a formal specification technique is extremely valuable, because it provides non-ambiguous, complete and concise ways of describing the behavior of the systems; Rigorous: The development life cycle of the 3DUI involves the same level of rigor that is typically used in software engineering; Reasoning: Because from the models describing the 3DUI some reasoning is possible, such as: Automatic. Computer based system might analyze data related to the 3DUI automatically and might be able to predict pilots behavior; Production of errors; Processable. Models can be processed and studied by devoted systems; Checking properties. Analysis of the different effects produced in the 3DUI by modifying properties of the components, for instance, changing background color, fonts of labels, etc; Human readable. This is not necessarily always achieved but model are expected to be understandable for humans.

Still this method presents some disadvantages; our main concern is that it is likely that the model transformations of bigger systems will be more complex to discover and to apply, so it is not clear if the solution is computationally feasible considering the amount of operations needed to perform graph transformations.

REFERENCES

[1] Bastide, R., Navarre, D., Palanque, P., Schyn, A., Dragicevic, P. A Model-Based Approach for Real-Time Embedded Multimodal Systems in Military Aircrafts. Sixth International Conference on Multimodal Interfaces (ICMI'04), Pennsylvania State University, USA. October 14-15, (2004)

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

[3] Buxton, W. A three-state model of graphical input. In: 3rd IFIP International Conference on Human-Computer Interaction, INTERACT'90, Cambridge, UK, 27-31 August (1990) 449-456

[4] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers, Vol. 15, No. 3 (June 2003) 289–308.

[5] Campos, J. C., Harrison, M. D. Formally verifying interactive systems: A review. In Design, Specification and Verification of Interactive Systems '97, Springer Computer Science, Springer-Verlag/Wien (1997), 109-124

[6] Celentano, A. and Pittarello, F. (2001). "A content centered methodology for authoring 3d interactive worlds for cultural heritage". D Bearman, F Garzotto, Eds., International Cultural Heritage Informatics Meeting, ICHIM, Cultural Heritage and Technologies in the Third Millennium, Milán, 3-7 September 2001, Vol. 2 pp. 315-324, Politecnico di Milano, Italia y Archives & Museum Informatics, Pittsburgh, PA, USA, Italia, 2001.

[7] Fencott, C. and Isdale, I. "Design Issues for Virtual Environments". International Workshop on Structured Design of Virtual Environments and 3D-Components at athe Web3D 2001 Conference. Paderborn, Alemania, 19 Febrero 2001.

[8] Geiger, C., Paelke, V., Reimann, C. C., and Rosenbach, W. "Structured Design of Interactive Virtual and Augmented Reality Content". International Workshop on Structured Design of Virtual Environments and 3D-Components at the Web3D 2001 Conference. Paderborn, Alemania, 19 February 2001.

[9] Gonzalez Calleros, J.M., Vanderdonckt, J., Arteaga, J.M., A Method for Developing 3D User Interfaces of Information Systems, Proc. of 6th Int. Conf. on Computer-Aided Design of User Interfaces CADUI'2006 (Bucharest, 6-8 June 2006), Chapter 7, Springer-Verlag, Berlin, 2006, pp. 85-100.

[10] Kaklanis, N., González-Calleros, J. M., Vanderdonckt, J. Tzovaras, D.: A haptic rendering engine of web pages for blind users. AVI 2008, ACM press, pp. 437-440.

[11] Limbourg, Q., Multi-path Development of User Interfaces. Ph.D. thesis. Université catholique de Louvain, IAG-School of Management. Louvain-la-Neuve (Nov. 2004).

[12] MacLean, A., Young, R.M., Bellotti, V., and Moran, T.P. Questions, Options, and Criteria: Elements of Design Space Analysis. Human-Computer Int. 6, 3-4 (1991) 201–250.

[13] Montero, F., Lozano, M., González, P.: IDEALXML: an Experience-Based Environment for User Interface Design and pattern manipulation. Technical Report DIAB-05-01-4. Universidad de Castilla-La Mancha, Albacete (2005).

[14] Navarre, D., Palanque, P. Schyn, A., Winckler, M., Nedel, L.P. and Freitas, C.M.D.S. A Formal Description of Multimodal Interaction Techniques for Immersive Virtual Reality Applications. In: Proceedings of INTERACT 2005, Rome, Italy, Springer-Verlag GmbH, v. 3585, p. 170-183, 2005.

[15] Neale, H. and Nichols, S. Designing and developing Virtual Environments: methods and applications. Visualization and Virtual Environments Community Club (VVECC) Workshop, Designing of Virtual Environments. 2001.

[16] Nedel, L.P., Freitas, C.M.D.S., Jacob, L.J., and Pimenta, M.S. "Testing the Use of Egocentric Interactive Techniques in Immersive Virtual Environments". In: Proceedings of INTERACT 2003, IOS Press, p. 471–478, 2003.

[17] Peterson, J. L. Petri Net Theory and the Modeling of Systems. Englewood Cliffs, New Jersey: Prentice-Hall, 1981.

[18] Sutcliffe. A. "Multimedia and Virtual Reality: Designing Multisensory User Interfaces". Lawrence Erlbaum Associates, 2003.

[19] Vanderdonckt, J., Bodart, F.: Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection. In: Proc. of the ACM Conf. on Human Factors in Computing Systems INTERCHI'93 (Amsterdam, 24-29 April 1993). ACM Press, New York (1993) 424–429.

[20] Vanderdonckt, J. A MDA-Compliant Environment for De-veloping 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.

[21] Vanderdonckt, J., Bouillon, L., Chieu, K.C., Trevisan, D.: Model-based Design, Generation, and Evaluation of Virtual User Interfaces. In: Proc. of 9th ACM Int. Conf. on 3D Web Tech. Web3D'2004 (Monterey, April 5-8, 2004). ACM Press, New York (2004) 51–60.

[22] Ying, J., & Gračanin, D. An approach to Petri net based formal modeling of user interactions from X3D content. In Proceedings of the eleventh international conference on 3D web technology, ACM Press, New York (2006) 153-157.