Comparing NiMMiT and Data-Driven Notations for Describing Multimodal Interaction

Joan De Boeck, Chris Raymaekers, and Karin Coninx

Hasselt University, Expertise Centre for Digital Media (EDM)
and transnationale Universiteit Limburg
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{joan.deboeck,chris.raymaekers,karin.coninx}@uhasselt.be

Abstract. In the past few years, multimodal interaction is gaining importance in virtual environments. Although multimodality makes interaction with the environment more intuitive and natural for the user, the development cycle of such an environment is often a long and expensive process. In our overall field of research, we investigate how model-based design can help shorten this process by designing the application with the use of high-level diagrams. In this scope, we developed 'NiMMiT', a graphical notation especially suitable for expressing multimodal user interaction. We have already experienced the benefits of NiMMiT in several in-house applications, and are currently assessing the value of NiMMiT with respect to existing notations. In this paper we report on our comparison of NiMMiT against some well known data-driven modeling notations.

Introduction

Interactive Virtual Environments (IVEs) are computer generated worlds that allow the user to intuitively interact with the data or objects in this world. To improve the intuitiveness of the interaction, the communication between the human and the system often is multimodal: information is not only exchanged visually and kinesthetic, but also via haptics, sounds or spoken messages. Due to the complexity of the virtual worlds, and the complexity of the human senses, designing such IVEs is more often than not a very time consuming and expensive process.

For several years, our lab has been conducting research on optimising interaction in IVEs using multimodal techniques. Several solutions using force feedback [1], speech or two handed input [2] already have been described in our former work. Even though we have gradually developed a code framework, each solution ended up in hundreds of lines of code to be written before being able to use the proposed interface. To shorten the design and development cycle, we developed NiMMiT (Notation For Multimodal Interaction Techniques), a visual notation optimised to describe human interaction in such IVEs.

NiMMiT has already shown its usefulness in several in-house applications, and will soon be integrated in other domains and other frameworks. At the same time, there is a growing need for tool support. NiMMiT has been developed, driven by the specific needs to model multimodal interaction techniques at a high level, minimising the amount and the complexity of the code to be written. For the development of NiMMiT we conducted a thorough research and studied several other existing graphical notations. As our current experience, and the future applications allow us to conclude that the NiMMiT approach looks promising, we decided to perform a more formal evaluation of the current version of NiMMiT against other existing solutions. In this paper, we will specifically focus on the data-driven notations, other state-driven notations will be evaluated similarly in our next work.

In the next section, we first briefly explain how we developed NiMMiT. Thereafter, we show the basic building blocks of our notation. In section 3, as an example, we explain the Voodoo-Dolls interaction technique [3] using NiMMiT. Subsequently, we try to express the same technique using some other diagrams. Finally we discuss the comparison of the different notations and state our conclusions.

1 Describing User Interaction

A possible solution to shorten the development cycle of an interaction technique (IT) is to *describe* the interaction, rather than *implementing* it. The description can be performed using a high-level graphical notation. The proposed notation should be suitable for

- allowing designers to communicate about the functionality of an IT, using an easy-to read diagram
- allowing an application framework to interpret (an XML-based equivalent of) the diagram, so it can be used for automatic execution.

The notation must provide enough low-level information for a framework to execute the diagrams, but it also needs to be high-level and easily readable for a designer to reason about the IT. Several general notations exist, such as State Charts [4], Petri-nets, Coloured Petri-nets [5] and UML [6]. Other notations, are extended or optimised to support user interaction in particular. Examples are InTml [7], ICon [8] and ICO [9][10].

To our opinion, to describe interaction, a notation should also satisfy the following criteria:

Event Driven: Interaction techniques are inherently driven by user-initiated actions, which we define as events. Human interaction is multimodal by nature. Multimodality can be seen as a combination of unimodal events (e.g. pointer movement, click, speech command, gesture, etc.). Hence, events are 'the initiators' of the interaction.

State Driven: While interacting with it, the system not always has to respond to all available events. Mostly, certain events must have been occurred before other events are enabled. Therefore, we define an IT as a finite state machine,

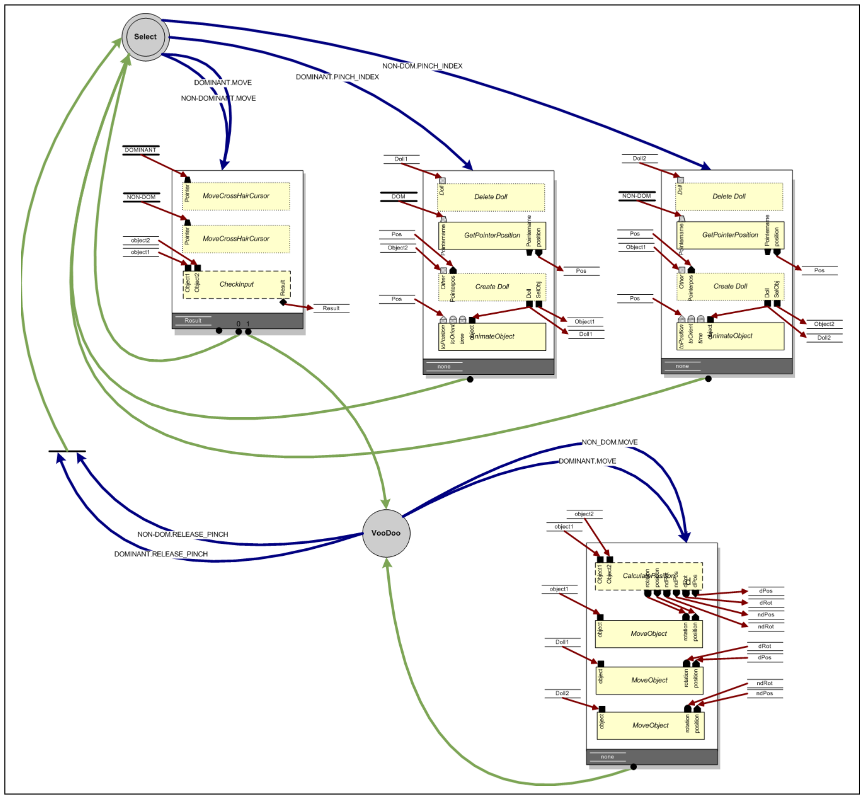

Fig. 1. NiMMiT Diagram of the VooDoo-doll interaction Technique

in which each state defines to which events the system will respond. The occurrence of an event invokes some action to the system, followed by a state transition.

Data Driven: Limiting our vision on interaction techniques solely to a finite state machine would be too restrictive, and it would violate the requirement of automatic execution. It is clear that user interaction implies some important data flow, internally (e.g. the collision between the virtual hand and an object in order to move that object). Obviously, certain data must be exchanged between the different actions within the interaction technique.

Hierarchical Reuse: Some subtasks of interaction techniques recur rather frequently. Selecting objects is an example of a very common component. When modelling a new interaction technique, the designer should be able to reuse descriptions that were created earlier. Therefore, the notation should support a hierarchical build-up, so an existing diagram of an interaction technique can be reused as a subtask of a new description.

Based upon the aforementioned existing notations, combined with the requirements to describe interaction, we developed NiMMiT. In the next section, we will briefly give an overview of the NiMMiT syntax.

2 NiMMiT Primitives

Based upon the strengths of the existing notations, we developed NiMMiT, to fulfill the special needs described in section 1. More details and other examples can be found in our previous publications [11][12]. In this section, we will briefly introduce the basic building blocks of the notation, after which, in the next section, the VooDoo-Doll interaction technique is shown as an example.

Basically, a NiMMiT diagram can be seen as a state transition diagram. At a certain time, the IT resides in a certain state, responding to certain events. These events can be multimodal by nature: gesture recognition, speech recognition, button clicks,. . . The recognition of an event triggers an activity, changing the inner state of the application. Thereafter a transition to a next state is performed.

We define the following basic elements (these can be recognised in figure 1):

State: A state is depicted as a circle. The interaction technique starts in the start-state, and ends with the end-state (if applicable). A state defines a set of events to which the system responds.

Event: An event is generated by the framework, based upon the user's input. A combination of events can be multimodal, containing actions such as speech recognition, gestures, pointer device events and button clicks. A single event or a specific combination always triggers the execution of a task chain. Events are depicted as an arrow pointing away from the state.

Task Chain: A task chain is initiated by an event and is depicted as a shaded rectangle in the diagram. A task chain is a linear succession of tasks, which will be executed one after the other.

Task: A task is a basic building block of the actual execution of the interaction technique. Typically, tasks access or alter the internal state of the application. E.g., when running in a typical 3D environment, a task can be 'collision detection', 'moving objects', 'playing audio feedback',. . . Tasks can be predefined by the system, but designers can also define their own custom tasks using C++ or a scripting language. Tasks can have input and output ports, on which they receive or send parameters or result values. Input ports are required or optional, indicated by a black or a grey input port respectively.

Labels: As data can be shared throughout a diagram, NiMMiT needs a system to (temporarily) store values. This is done in 'labels', which can be seen as high-level variables. Labels are depicted in the diagram beside the task chains and are always connected to the input or output ports of a task.

State Transitions: Finally, when a task chain has been executed completely, a state transition moves the diagram into the next state. A choice between multiple state transitions is also possible, based upon the value of a certain label.

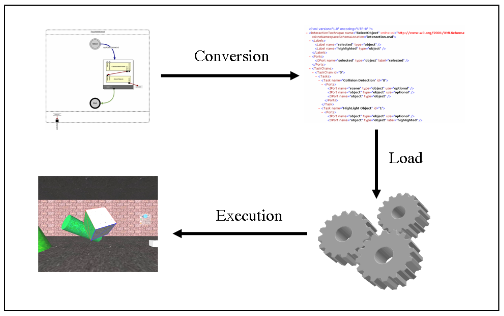

Fig. 2. Execution of a NiMMiT Diagram

A NiMMiT diagram does not support concurrency in principle, but several diagrams can run simultaneously and can be synchronised at any time. More advanced multimodality such as complementary and redundancy as described in [13] are possible. This is described more in detail in [12].

To meet our requirement of execution, a diagram is saved to an XML equivalent. That XML is read and interpreted by an application framework, as shown in figure 2.

3 Evaluation Approach

In this section, we will first define the evaluation criteria, which we will adopt for the assessment. Then, we will explain the VooDoo-Dolls interaction technique [3] as a case study. In the next section, we first elaborate on VooDoo-dolls, using NiMMiT. Subsequently, we try to describe the same IT using other data-driven notations. We opted to elaborate on InTML and UML: InTML, because it is a purely data-driven notation, and UML as it is the most fundamental standard. Finally, we discuss the different diagrams and their suitability to describe ITs against the defined criteria.

3.1 Evaluation Criteria

The evaluation of graphical notations is not an exact process which often results in variegated and/or subjective conclusions. To minimise the impact of subjective impressions, we will first define on which criteria the notations will be evaluated, however one can criticise that selection of criteria is subjective in some respect, as well. The criteria are based on Green's ‘Cognitive Dimensions of Information Artefacts’ [14][15]. Cognitive Dimensions are, as the author says, ‘discussion tools that give names to concepts that their users may only have half-formulated’. The CD framework comes with 14 dimensions which focus on different aspects of the notation. Each dimension can either be positive or negative, dependent on the application in which the notation is applied. Green distinguishes four possible applications:

Incrementation: Adding new information; e.g. writing a new piece of programming code.

Transcription: Translating from one system to another; e.g. copying book details to an index card.

Modification: Changing existing information; e.g. rearranging and changing some parts of a flowchart.

Exploratory design: Combining incrementation and modification, with the characteristic that the desired end state is not known in advance; e.g. sketching, programming on the fly (hacking).

In our research domain, exploratory design is one of the most important applications of a high-level notation. As the notation is mainly intended to easily design, try and modify interaction techniques, a diagram is often designed step by step in a trial-and-error manner. Since the evaluation of a dimension is highly dependent on the application, we evaluate the selected notations in the scope of 'Exploratory design'.

The CD framework defines the folowing cognitive dimensions: abstraction, hidden dependencies, premature commitment, secondary notation, viscosity, visibility, closeness of mapping, consistency, diffuseness, error-proneness, hard mental operations, progressive evaluation, provisionality, role-expressiveness.

In the scope of this paper, and the evaluation of the particular notations, the following dimensions are relevant:

Abstraction is a grouping of elements to be treated as one entity. The **abstraction barrier** is the minimum number of abstractions that must be known before the notation can be used. **Abstraction tolerant** systems allow users to make their own abstractions, but don't require to do so. In our solution we strive to a low abstraction barrier, but an abstraction tolerant notation.

Diffuseness expresses the verboseness of a notation. A notation for interaction techniques must be not to diffuse.

Role-Expressiveness describes how easy it is to distinguish the different components or logical blocks in a notation. It is clear that the notation in our application must be as role-expressive as possible.

Viscosity is the resistance of a notation to change, or in other words the implications of a small change on the entire diagram. As we take the exploratory design as the main application for the notation, a high viscosity is adverse.

Progressive Evaluation means that the work in progress can be easily checked for as far as it is finished. This means that only a part of the entire solution can be evaluated. It is clear that in exploratory design, this is important.

Premature Commitment expresses in what amount a designer must make some decisions in advance before the proper information is available.

3.2 VooDoo-Doll Interaction Technique

In order to assess the selected notations, we have chosen to elaborate on the VooDoo-Dolls interaction technique [3], because it is a well known two-handed metaphor, which results in fairly easy diagrams in all notations, while still demanding most of the requirements of multimodal interaction. We will assume that if this IT can be easily modelled using the evaluated notations, there is a good chance that other techniques will fit, as well.

Voodoo-Dolls is a two-handed IT used to manipulate (distant) objects in 3D. By moving one of the hands, a crosshair cursor (attached to that hand) is moved accordingly. If the cursor moves over an object and the index and thumb of that hand are closed ('pinch'-event) a 'doll' is created and attached to the corresponding hand. A doll is a representation of the original virtual object, but scaled and attached to the movements of that hand. As soon as two dolls are defined, the manipulation phase of the IT starts. By moving the doll of the dominant hand, with respect to the doll of the non-dominant hand, the original object is moved with respect to the other object. When the thumb and the index are released, the doll is removed and the original object keeps its new location. The aim of this IT is to manipulate objects at any scale (close-by and far) with respect to each other. E.g. if the user creates a doll of a tea-pot in the dominant hand, and a table on the other side of the room in the non-dominant hand, he can easily put the pot on the table by moving both dolls with respect to each other.

4 Evaluation Results

4.1 NiMMiT

NiMMiT Diagram

Referring to figure 1, the IT starts in the 'Select'-state. Here the system responds to four events: either a move of the dominant or the non-dominant hand, or a 'pinch' (closing thumb and index) of one of the hands. If a 'pinch' occurs, dependent on the hand, one of the task chains at the right side of the diagram is executed, creating a doll for that hand. Each time one of the hands moves, the crosshair cursors of both hands are updated and the system checks if there are already two dolls present. If not, a state-transition to the original state is performed. Otherwise, we arrive at the 'VooDoo'-state. If the hands are moved now, the activated task chain calculates the object's new position with respect to the reference object (attached non-dominant hand's doll), and the object and the dolls are moved accordingly. When the user releases the pinch of one of the hands, a transition to the 'Select'-state is performed and a new doll can be made.

Evaluation

If we compare the NiMMiT diagram against the aforementioned criteria, we see that NiMMiT has a reasonable **abstraction** barrier, since it consists of about 10 constructs. It is allowed for a designer to define custom abstraction at the level of hierarchical diagrams. Concerning the **diffuseness**, there are no unnecessary structures, although one can have objections against the explicit internal labels within a task chain (see bottom-most task chain in figure 1). As there is a clear syntactic difference between states, task chains, events and transitions, the notation is enough **role-expressive**. The **viscosity** is reasonable, although changes in a task-chain can have an influence on the data flow of the entire diagram. As the diagram must not be fully functional before it can be run, **progressive evaluation** is supported. Finally, as a drawback of any graphical notation, some **premature commitments** are required for positioning the primitives within the diagram.

4.2 InTML

InTML Diagram

InTML [7] is a purely data-driven notation to describe the execution of an application at a high level. It does not support states, but uses 'filters' instead. A filter can be seen as a 'black box', performing some activity based upon its inputs, while returning the output at its output ports. Important to know is that the control flow of the diagram is implicitly driven by the data, as a filter is only executed as soon as it contains a valid input on all of its input ports.

Figure 3 shows a possible diagram in InTML of the VooDoo-Dolls interaction technique.

At the left hand side, the user's input is shown. The positions of the user's dominant and non-dominant hand always are valid inputs. Hence, the filters moving the cursors (crosshairs) are constantly executed. As soon as the user closes the thumb and the index of a hand, the 'Pinch'-variable becomes valid, and a doll is created. The 'Create Doll' filters create a doll from the object which is indicated by the crosshair cursor. It returns the object and its doll. As soon as both dolls are created, the 'checkdolls' filter returns 'true', as it has a valid input on all ports. The returned value activates the 'MoveDolls' and 'MoveObject' filters, which moves the dolls and the object according to the movements of the user's hands. As soon as a pinch is released and a valid doll exists, the doll is released in the 'ReleaseDoll' filter.

Evaluation

InTML has a minimal **abstraction barrier**, as it only consists of three constructs. However, to our knowledge, the notation does not allow users to define their own abstractions. The low barrier, however, means that the **diffuseness** will increase, as for more complex structures more primitives are needed. This

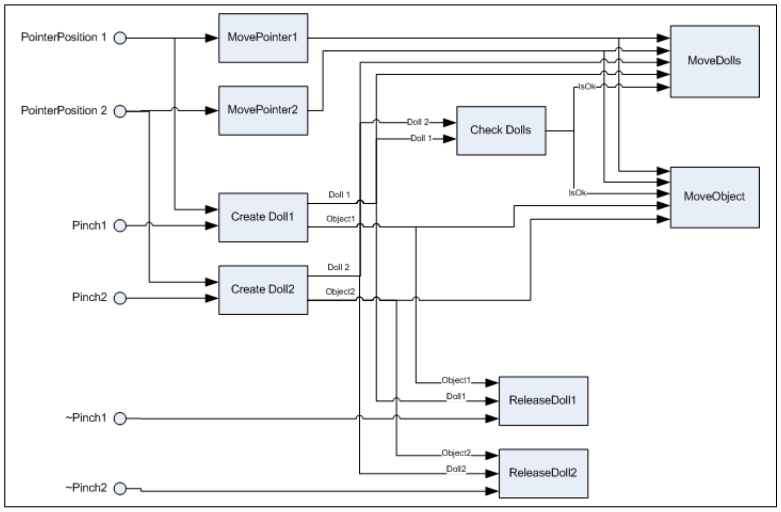

Fig. 3. InTML Diagram of the VooDoo-doll Interaction Technique

will have its implications to the **viscosity**, as well. E.g. if the 'Object In Hand' metaphor [2], with more mutual exclusive phases should be described, more additional filters will be necessary to enable or disable certain parts of the diagram. Moreover, if one of the first filters must be changed, it has implications on all subsequent filters. Next, in our opinion, the **role-expressiveness** of the notation is not optimal, since recognising functional blocks in the diagram is only based upon their location. Although difficult to check in practice, we can imagine that **progressive evaluation** is supported so that it is not required to finish the entire diagram before it can be tested. Finally, here again **premature commitments** are required for the positioning of the notation's primitives.

4.3 UML Activity Diagrams

UML Diagram

Figure 4 shows the same interaction technique, modelled using UML activity diagrams. We start at the topmost node, where we wait for the occurrence of one of the events. When a pinch is recognised, the doll belonging to that hand is created, and it is saved in a datastore. When the hands are moving, the crosshair cursors are constantly updated, and the 'CheckInput' activity checks if two dolls are currently present. When the check fails, the current node is repeated.

As soon as two dolls become available, the bottommost node is executed. Each time one of the hands is moved, the new position of the dolls and the objects is calculated. The objects are moved accordingly, restarting the same activity node again. When a pinch of one of the hands is released, an interruptible edge brings the token back to the first activity node.

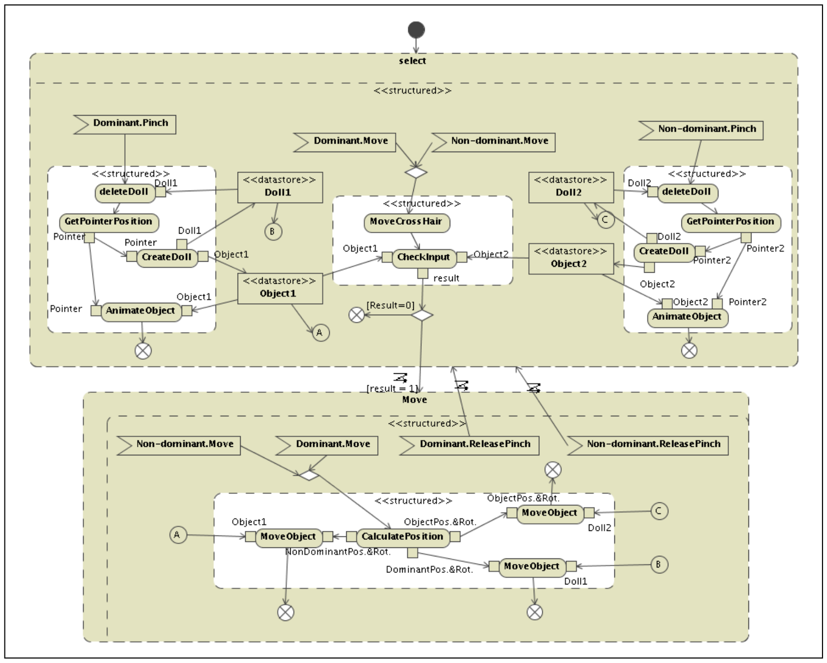

Fig. 4. UML Activity Diagram of the VooDoo-doll interaction Technique

Evaluation

The UML **abstraction barrier** is somewhat higher. In this diagram only 11 constructs are used, but inherently, as the standard is more extensive, it is not inconceivable that more structures must be known. UML allows for user defined abstraction but does not requires this, which is the approach we prefer. The **diffuseness** of the notation is good, however when we do not define specialised stereotypes, there is some overhead of specific structures (such as datastores, structured activity nodes and interruptable regions) which are needed for syntactic and semantic correctness. The **role-expressiveness** and the **viscosity** are acceptable, although there is no syntactic difference between control flow and data flow, which can be seen as a minus. Finally, it appears that **progressive evaluation** can be supported. Finally, within UML, once again, **premature commitments** are required for the positioning of the primitives, resulting in a higher viscosity.

4.4 Discussion

We can summarise the results for the cognitive dimensions in the main application of an exploratory design as shown in table 1.

Comparing the three notations to each other, we can notice that InTML has a minimal **abstraction** barrier with only three constructs. However this will lead

Table 1. Cognitive Dimensions for the notations

	InTML	UML	NiMMiT
Abstraction	Minimal barrier No additional abstr	Acceptable barrier Abstr Tollerant	Acceptable barrier Abstr Tollerant
Diffuseness	Quite Verbose	Some overhead of extra structures	Some overhead of internal labels
Role-Expressiveness	Functional blocks only recognisable by location	Acceptable	States and Tasks are well distinguishable
Viscosity	Strong implications in subsequent filter	Some data flow issues	Some data flow issues
Progressive Evaluation	Supported	Supported	Supported
Premature Commitment	Only for location of primitives	Only for location of primitives	Only for location of primitives

to a higher **diffuseness**. Moreover, InTML has no support for other user-defined abstraction. In contrast, UML and NiMMiT use a higher but comparable number of constructs. This makes the abstraction barrier higher, but still acceptable. Both notations have some syntactical overhead, but they do not form a real problem concerning the readability of the diagrams.

The **roll-expressiveness** of InTML is low. As a result of the low number of constructs, the only way to distinguish functional blocks is by their location. The role-expressiveness of UML is significantly better, but we regret that there is no syntactic difference to indicate data flow and control-flow, as we prefer to see both as separate entities within an interaction technique. Because NiMMiT makes a clear syntactic distinction between states, events, tasks and transitions, we prefer the role-expressiveness of NiMMiT.

Diagrams in InTML can quickly grow as the interaction technique consists of more mutual exclusive phases, resulting in a higher **viscosity**. Moreover, InTML is based on the principle of executing filters as soon as it receives a legal value on all of its input ports. This means that a change to a filter, will inherently propagate to the next filters, resulting in a higher viscosity, again. UML and NiMMiT appear to have a similar viscosity. Changes to the diagram will inherently have implications to the data flow of the remainder of the diagram; however, as data flow is not the main aspect of the diagram, the impact will be lower as with InTML.

Finally, it appears that all notations support **progressive evaluation**, which is desirable in the context of an exploratory design. Since InTML, NiMMiT and UML are all graphical notations, they all require some **premature commitments** with respect to the location of the primitives. As far as we can see, this is the only occurrence of premature commitments, which is acceptable.

In summary, we can conclude that, according to the proposed criteria, InTML is less suitable to describe user interaction, in the experimental environment in which we want to apply it. NiMMiT and UML are very similar and are both suitable. The overhead of the UML notation in the example can be easily reduced by defining suitable stereotypes. However, we are somewhat concerned about the underlying complexity of the entire UML standard, which must be known for the integration of frameworks for automatic execution and the development of tool support. Using stereotypes will facilitate the use of the notations, but will even increase this underlying complexity. UML is indeed a very general and powerful standard, but only a very small part of it is applicable for describing user interaction. On the other hand, NiMMiT is especially designed for describing user interaction, and is therefore more dedicated to this domain.

Keeping the findings above in mind, it is difficult to make an exclusive choice between both notations. However, the research in this paper has led to some concrete recommendations for the NiMMiT syntax to be improved, such as the removal of internal labels within a task chain, or an improved abstraction tolerance so that complex diagrams can be simplified by user defined abstractions.

5 Conclusions and future work

We evaluated NiMMiT, a graphical notation dedicated to describe interaction techniques against other existing data-driven notations. Compared against criteria, based on Green's 'Cognitive Dimensions', we can conclude that purely data-driven notations (such as InTML) are less suitable to describe interaction techniques. The general standard UML appears to be comparable to NiMMiT. However, we still have our concerns to adopt the UML standard, as this will complicate the development of tool support and the integration of interpreters in our model-based approach.

As a result of this research, we will concretely update the NiMMiT notation, and at the same time keep an eye on the usage of UML in the domain of VE-development.

As stated in the introduction, this paper focussed on the comparison of NiMMiT against data-driven alternatives, in particular. In a next step, we plan to evaluate NiMMiT in a similar way against the family of the state-driven notations, such as Petri-Nets and ICO.

Acknowledgements

Part of the research at EDM is funded by ERDF (European Regional Development Fund), the Flemish Government and the Flemish Interdisciplinary Institute for Broadband Technology (IBBT).

NiMMiT has been developed within the VR-DeMo project (IWT 030248), which is directly funded by the IWT, a Flemish subsidy organization.

The authors also want to thank Jan Van den Bergh for his valuable contributions applying the UML notation.

References

1. De Boeck, J., Raymaekers, C., Coninx, K.: Aspects of haptic feedback in a multimodal interface for object modelling. Virtual Reality Journal **6** (2003) 257–270
2. De Boeck, J., Cuppens, E., De Weyer, T., Raymaekers, C., Coninx, K.: Multisensory interaction metaphors with haptics and proprioception in virtual environments. In: Proceedings of the third ACM Nordic Conference on Human-Computer Interaction (NordiCHI 2004), Tampere, FI (2004)
3. Pierce, J., Stearns, B., Pausch, R.: Voodoo dolls: seamless interaction at multiple scales in virtual environments. In: Proceedings of symposium on interactive 3D graphics, Atlanta, GA, USA (1999)
4. Harel, D.: Statecharts: A visual formalism for complex systems. In: Science of Computer Programming. Volume 8. (1987) 231–274
5. Jensen, K.: An introduction to the theoretical aspects of coloured petri nets. In: W.-P. de Roever, G. Rozenberg (eds.): A Decade of Concurrency, Lecture Notes in Computer Science. Volume 803., Springer-Verlag (1994) 230–272
6. Ambler, S.: Object Primer, The Agile Model-Driven Development with UML 2.0. Cambridge University Press (2004)
7. Figueroa, P., Green, M., Hoover, H.: InTml: A description language for VR applications. In: Proceedings of Web3D'02, Arizona, USA (2002)
8. Dragicevic, P., Fekete, J.D.: Support for input adaptability in the ICON toolkit. In: Proceedings of the 6th international conference on multimodal interfaces (ICMI04), State College, PA, USA (2004) 212–219
9. Navarre, D., Palanque, P., Bastide, R., Schyn, A., Winckler, M., Nedel, L., Freitas, C.: A formal description of multimodal interaction techniques for immersive virtual reality applications. In: Proceedings of Tenth IFIP TC13 International Conference on Human-Computer Interaction, Rome, IT (2005)
10. Palanque, P., Bastide, R.: Petri net based design of user-driven interfaces using the interactive cooperative objects formalism. In: Interactive Systems: Design, Specification, and Verification, Springer-Verlag (1994) 383–400
11. Coninx, K., Cuppens, E., De Boeck, J., Raymaekers, C.: Integrating support for usability evaluation into high level interaction descriptions with NiMMiT. In: Proceedings of 13th International Workshop on Design, Specification and Verification of Interactive Systems (DSVIS'06), Dublin, Ireland (2006)
12. Vanacken, D., De Boeck, J., Raymaekers, C., Coninx, K.: NiMMiT: A notation for modeling multimodal interaction techniques. In: Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP06), Setbal, Portugal (2006)
13. Coutaz, J., Nigay, L., Salber, D., Blandford, A., May, J., Young, R.M.: Four easy pieces for assessing the usability of multimodal interaction: The CARE properties. In: Proceedings of INTERACT95, Lillehammer (1995) 115–120
14. Green, T.: Cognitive dimensions of notations. In: People and Computers, Cambridge University Press, Cambridge, UK (1989) 443–460
15. Green, T., Blackwell, A.: Cognitive dimensions of information artefacts: a tutorial. http://www.ndirect.co.uk/ thomas.green/workstuff/papers/ (2005)