Modelling Interaction in Virtual Environments

Shamus Smith, David Duke, Tim Marsh, Michael Harrison and Peter Wright
Department of Computer Science
University of York
York YO10 5DD, UK
{shamus, duke, tmarsh, mdh, pcw}@cs.york.ac.uk

Abstract

This paper presents some early research from the INQUISITIVE project on abstracting models of interaction in virtual environments away from technological constraints with an aim to providing general guidelines for system developers. The use of Statechart based modelling techniques have been investigated. Unfortunately state based techniques do not show the features of the interaction which make different techniques distinctive in virtual environments. The use of Communicating Sequential Processes (CSP) and a process driven approach has been used to try and overcome this limitation.

Key Words: Modelling - Interaction - Virtual Environments - CSP - Statecharts

1 Introduction

Research into virtual reality (VR) and virtual environments (VEs) has been predominantly lead by the development of new technologies. Although many of these technologies have matured, there are still many issues about virtual environments that remain unanswered.

The INQUISITIVE project is a three year research effort between groups at the University of York and the Rutherford Appleton Laboratory (RAL). Its aim is to understand how to design and implement better user interfaces for VR systems and to develop methods and principles that can be used at an early stage in the design life cycle and system evaluation. The concern is not so much with the physical devices such as headsets and data gloves that have come to characterise interaction within VR, but rather with addressing the highly interactive and dynamic nature of user-system interaction that this technology supports.

One important distinction that has been noted is that of the differing requirements of the users and designers of virtual environments. In the context of research tools, designers and users are closely bound. Typically, system designers are either the main users or closely associated with or working in the same domain as the target users. However, as VR technologies reach a larger audience, new ways of capturing user requirements explicitly for designers will be needed.

This paper presents some early research from the INQUISITIVE project on abstracting models of interaction in virtual environments away from technological constraints with an aim to providing general guidelines for system developers. An initial attempt at building a general framework to classify interaction in virtual environments, the consideration of several modelling techniques and how they may be applied to virtual environment interaction techniques has been investigated [1]. This paper describes the modelling of interaction component of that work.

2 Why Model Interaction?

Precise interaction is difficult in the virtual world. Mine [2] notes that many virtual worlds lack haptic feedback (something we take for granted in the real world). However, this is increasingly not so in high-end systems. Mine also observes that current alphanumeric input techniques (what we use for precise interaction in the computer world) for virtual worlds are ineffective. He suggests that we must learn how to interact with information and controls distributed about the user instead of focused on a terminal in front of him/her. If natural forms of interaction can be identified, and possibly extended in the VE, then more usable VE interfaces can be constructed [2].

Modelling interaction is both important from the user and designer of VEs perspective. The users require interaction techniques which allow them to complete interaction tasks in a particular application and designers wish to build systems that make the required interaction possible.

There are three general areas which are of interest when examining interaction techniques, namely, their *specification*, *modelling* and *evaluation*.

2.1 Specification

An initial place to start when looking at characterising an interaction technique is to define what the technique provides and what the technique requires. Techniques provide elements both to the user, e.g. visible clues, feedback and affordances, and to the system, e.g. state change information and virtual logical devices. Also special provision for disabled or learner users may be provided.

Technique requirements can be system, technology or user based. System requirements may involve a particular interface metaphor or logical devices, while technological requirements are usually more physically based, e.g. frame refresh rate and physical device input/output. User requirements may involve a certain skill level, both in technique knowledge and physical dexterity.

Two related questions are which requirements are provided by the system, the technology and the user and when are these requirements needed? User based requirements may require advanced training and system requirements may need technological advances before they become feasible.

2.2 Modelling

Techniques for modelling interaction are still an active area of research [3, 4], even without considering the apparent complexity of VR. Numerous techniques for defining models have been suggested. Many of these are specialised either to classes of interfaces, classes of problem, or are scoped by a set of assumptions about the design process (e.g. formal notations). This makes a direct comparison between techniques difficult; nevertheless, a number of dimensions can be identified.

- Concern: state, process, other.
- Form of representation: diagram, text, hybrid.
- Level of rigor: informal, formal.

Also, there is the question of what should be in the model. Can the interaction technique stand alone or are models of the user and the system also required? If the latter, can they be modelled within the same framework, and if not, how can they be combined to provide an overview of the environment?

2.3 Evaluation

Typical evaluation in the area of VEs has focused on ad hoc user studies [5, 6]. These are usually carried out with a small number of users, and typically not from a broad user base. For well defined applications with a specific user group, this may be adequate, but it is unclear how these results can be generalised to help during the design phase. Some method of evaluation grounded on a theoretical understanding of virtual environments would be more useful and could be used to contrast different models. However, what do we require to be evaluated?

There are several areas of evaluation which seem relevant. Usability (ease of use, affordances), ergonomics and human factors (motion sickness, arm fatigue and eye strain), resource usage (user cognitive load and system requirements), robustness and ease of implementation are all valuable areas of study.

In this paper, we are interested in the modelling component and how an appropriate modelling technique can be chosen or developed. However, before interaction techniques can be defined, consideration of their context of use, or *mode*, is required.

3. Interaction Modes

Modes are implemented, in one way or another, in almost any modern machine or piece of equipment [7]. Andre and Degani [8] comment that modes represent the different behaviours, or functions, of a given system. Many interaction techniques in VEs are mode oriented, e.g. whether the user is navigating, selecting objects or manipulating objects. System behaviour defines what the user perceives of the environment and how their interaction is progressing. Therefore knowing what mode the system is in can be very important to the user in VEs. Many problems in human interaction can be found in users confusion over mode. Degani, Shafto and Kirlik [9] observe that human interaction with complex systems is a non-trivial problem and it appears that mode confusion is a significant contributor to mishaps in many domains.

System mode is also relevant when trying to define the usability of a system. The user should be able to tell what mode they are currently in [8]. Mine, Brooks and Sequin [10] note that constantly switching between object interaction and movement control breaks the natural rhythm of the operation and adds significant cognitive overhead. Although there are a variety of different interaction techniques, many can be grouped loosely into the interaction mode which they apply, e.g. *navigation*, *object selection*, *object manipulation* or *environment commands*.

Some examples are :

- Navigation: Walking [2], Flying [2], Worlds in miniature (WIM) [10, 11], Head-butt zoom [10], Two handed flying [10].
- Selection: Laser pointing [2], Cone-casting, Head crusher select [12], Go-go interaction [13], Look-at menus [10].
- Manipulation: Scaled world grab [10], Orbital mode [14], Hand-held widgets [10, 15], Over-the-shoulder deletion [10].
- Environment Commands: Pull-down menus [10], Voice recognition [16], Gesture recognition [17, 18].

These groups are not totally rigid as some techniques combine elements of two or more modes. For example, the Scaled world grab [10] has components of object selection followed by object manipulation. Also, Scaled world grab can be used as a technique for navigation where, instead of bringing objects to the user, the world is moved in relation to the selected object.

Following are a brief overview of three interaction techniques which will be used as examples later in this paper.

3.1 Head Crusher Select

The Head Crusher Select (HCS) [12] is based in an immersive virtual environment and involves operating on 3D objects at a distance by interacting with 2D projections on the image plane. In this case, the thumb and forefinger is shaped around the desired object as if the environment was 2D. A projection is taken from the users hand and head position to determine what object is to be selected (Figure 1). This technique was developed as one of four techniques, with the Sticky Finger, Lifting Palm and Framing Hands [12], to demonstrate the use of 2D projections on a 3D image plane.

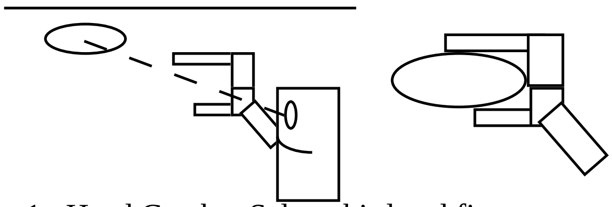
Figure 1 : Head Crusher Select third and first person view.

The following three techniques are example techniques developed by Mine, Brooks and Sequin [10] to exploit user proprioception in virtual environment interaction.

3.2 Two Handed Flying

Flying is a navigation technique which has been popular in many virtual environment implementations. However, there are several disadvantages to this technique including arm fatigue and user disorientation caused by misunderstanding the relationship between hand orientation and flying direction [2]. Two Handed Flying (THF) [10] is a specialised type of flying which exploits proprioception, the person's sense of the position and orientation of their body and limbs. Direction of flight is defined by the vector between the users two hands and the flight speed is specified by the distance between the users hands. Flight is stopped by moving the hands into a *deadzone*, a minimum hand separation distance.

3.3 Head-butt Zoom

A more complex technique is the Head-butt Zoom (HBZ) [10]. It uses both hand and head motion to control the interaction. Initially, the user forms a *view rectangle* using their hands to mark the positions of the top-right and bottom-left corners of a rectangle (Figure 2). Once this is completed, head motion is used to switch between environment views. If the user leans forward through the plane of the rectangle, the view is zoomed in and if the user leans backward, the view is zoomed out. This technique can also be used for navigation. If the user steps through the plane of the view rectangle, they are moved to the zoomed location. If they then step back, they are returned to their original position.

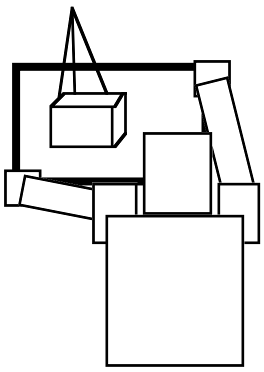
Figure 2 : Head-butt Zoom view rectangle creation.

4 Modelling Techniques for Interaction

Many components of virtual environments are tightly coupled together and their explicit separation may be ill advised, if not impossible. Therefore, modelling particular interactions is a non-trivial task. One question we may ask is what do we want to achieve with the modelling? Modelling interaction is one component of a framework being developed in the current project for the design and evaluation of virtual environment systems. Currently, we are interested in investigating ways of describing interaction concisely.

Initially, the use of a Statechart [19] based technique seemed appropriate as Statecharts provide many powerful modelling mechanisms which are desirable in the modelling of virtual environments. Two techniques investigated were Ofan [8] and RSML (Requirements State Machine Language) [20].

One interesting issue in interaction is the possibility of mode conflicts. An example of this in a virtual environment could be two interaction techniques being used in parallel where one is activated using hand positions and the other on hand gestures. Using a modelling technique that allowed such conflicts to be identified seems a good place to begin and initially lead to the investigation of Ofan due to it being a mode oriented technique.

The Ofan modelling framework is based on the Statecharts and Operator-Function models [21]. In Ofan, five concurrently active modules are used to describe the human-machine-environment, namely the *Environment*, the *Human Functions/Tasks*, the *Controls*, the *Machine*, and the *Displays*. Applying the Ofan framework allows the identification of potential mismatches between what the user assumes the application will do and what the application actually does [22]. For a more detailed description of the Ofan framework see [8, 9, 22].

The Ofan framework attempts to separate out the components of the whole environment. Unfortunately, in virtual environments there is not always a clean distinction between components such as the environment, the display and the control. In particular the linkage between the control and the display of the environment is very strong within virtual environments. Also specifying the control behaviour of every object in an environment would be complex and time consuming. Degani and Shafto [22] observed in the specification of a WWW browser, with a mouse and keyboard as input devices, that their specification became extremely complicated. Virtual environments typically have input devices more complex than a mouse and keyboard and would therefore be at least as difficult to specify.

Similar to Andres and Degani, Leveson, et al. [20] also utilised the power of Statecharts in their specification language RSML. RSML provides a modelling environment which utilises both graphical, textual and tabular notations. Statecharts are used to expresses the state changes and textual and tabular descriptions are used to describe transitions, input/output specifications, interface components and system macros.

However, a problem with diagram based modelling techniques is that depending on what level of detail is needed, a complete definition of even a simple interaction techniques can involve several pages of documentation. Spreading models over many documents can make them hard to comprehend and error-prone to define. As with graphical notations in general, it is difficult to display the whole diagram at one level and some type of decomposition or abstraction is usually required. This can make models difficult to initially develop, validate after definition and compare with other models.

One solution to this problem is to move away from graphical notations and to look at a more formal text-based notation. Also, as Statecharts focus on the state of an environment and we are primarily interested in the processes of getting from state to state, something that is hidden on Statechart based notations, a possible process based notation is Communicating Sequential Processes (CSP) [23].

5 Communicating Sequential Processes (CSP)

Communicating sequential processes is a modelling technique which takes the basic concept of a process as a mathematical abstraction of the interactions between a system and its environment [23]. This concept is extended to include the use of concurrency, nondeterminism and interprocess communication.

What we have looked at here is at interactions in virtual environments as a set of concurrent processes. User processes, e.g. head movement and hand movement, provide certain outputs which are processed by tracking devices (system processes) which work in parallel with an interaction definition to provide the interaction model.

As the purpose of this section is to illustrate the potential use of CSP, rather than provide definitive accounts of particular techniques, the approach taken is informal. Only a small number of the operators available in CSP for modelling processes have been used; these are introduced below.

In CSP, a process models the observations that can be made of a system, as it evolves by engaging in *events*. For each process *P*, the finite or countable set of events that *P* can engage in is called the *alphabet* of *P*, written αP. Process expressions are formed from the names of events, usually written lowercase, names of processes, normally written uppercase, and simpler process expressions, combined using various operators.

STOP The simplest process in CSP is the process that refuses to engage in a given set of events *A*, written $STOP_A$ (the annotation with the alphabet becomes important when one begins to combine processes).

Prefix Given a process *P* and an action *a* in the alphabet of *P*, the "prefix" constructor is used to define the process that is first observed to do *a*, and subsequently behave like *P*. This is written *a --> P*.

Parallel Two processes *P* and *Q* can be combined in parallel to produce a new process, *P || Q*. When restricted to the alphabet of *P*, each behaviour of *P || Q* must be a behaviour of *P*, and similarly for restriction to the alphabet of *Q*. For example, the process

$$a \dashrightarrow b \dashrightarrow STOP_{\{a,b\}} \,||\, c \dashrightarrow b \dashrightarrow STOP_{\{b,c\}}$$

can generate the behaviours *a, c, b* and *c, a, b*, but can not generate *a, b, c,* since *b* is in the alphabet of both the left and right process and the right process requires that the *b* event must follow a *c* event.

Choice The choice operator $\Box$ represents an external choice between the behaviours offered by two processes. Here "external" means that the environment in which the processes are operating will select a process that enables progress to be made. A separate operator, $\sqcap$ internal choice, represents non-deterministic choice on the part of a process itself.

Labelling It is sometimes useful to work with multiple distinct copies of a process. These can be defined by using the labelling operator; if *P* is a process with alphabet *{a, b, c, ...}* and *s* is some label, the process *x:P* is like *P* except that its alphabet becomes *{ s.a, s.b, s.c, ... }*.

Events in CSP represent *observations*, for example that a button has been pressed or a signal has been sent. In some cases, it is useful to view events as the transmission or reception of data along some channel. The notation *c!m* is defined to be the event in which the message *m* is send on the channel *c*, while *c?v* represents the arrival of a value *v* on the channel *c*.

5.1 CSP Examples

Within virtual environments are there are several components which are shared by most interaction techniques, e.g. the position of the logical hand and the logical head. These can be defined as concurrent process which operate to provide *user frame* based output. We are thinking of a "frame" as representing the state of the user and other objects within the environment, including for example the position, orientation and other status information of each such object. The idea is that input devices (both physical and logical) can be modelled as processes that construct and/or modify frames which are passed from the interface towards the application. For example, a hand tracker process might simply report the configuration of the hand as a series of frames sent along a particular channel:

Hand = hand.posn!frame --> Hand

We would hope eventually to describe interaction by using a number of basic messages that can be sent along channels; the composition of these messages over time represents the evolution of those parts of the system state that are of interest. Possible messages might include: *GetFrame*, *SetFrame*, *SaveFrame*, *ModifyObject(select,object)*, *ModifyObject(deselect,object), ModifyObject(move,object)* and *ModifyObject(delete,object).*

The following sections contains sketches of how some of the interaction techniques discussed in this document could be represented as CSP processes.

5.1.1 General CSP

Hand = posn!Frame -->Hand
Head = posn!Frame -->Head

5.1.2 Head-butt Zoom CSP

Rect = posn!Frame -->Rect

HBZ = MkRect --> rect.posn?r --> head.posn?h --> PerformHBZ(r,h) (1)

(1) Once the HBZ has been activated, construct the initial view rectangle and perform the HBZ based on the current positions of the rectangle and the users head.

MkRect = left.posn?lh --> right.posn?rh --> SetRect(lh,rh)

PerformHBZ($curr_r$, $curr_h$) =
head.posn?h --> MkRect --> rect.posn?r --> PerformHBZ(r,h), $curr_r \neq h$ (2)
□ *head.posn?h --> ZoomIn(h) --> PerformHBZ($curr_r$,h), $curr_h <= h$ & $h = curr_r$* (3)
□ *head.posn?h --> ZoomOut(h) --> PerformHBZ($curr_r$,h), $curr_h > h$ & $h = curr_r$* (4)

(2) Head and rectangle are not intersecting so must be resizing the rectangle.
(3) Moving the head forward through the rectangle so zoom in.
(4) Moving the head backward from the rectangle so zoom out.

ZoomIn($curr_h$) =

$\quad GetFrame(v) \dashrightarrow SetFrame(v + zoomfactor),\ curr_h \approx v \qquad (5)$

$\square\ GetFrame(v),\ curr_h \neq v \qquad (6)$

(5) If not zoomed in then zoom the view by static zoom factor
(6) Already zoomed in, so do nothing.

$ZoomOut(curr_h) =$

$\quad GetFrame(v) \dashrightarrow SetFrame(v - zoomfactor),\ curr_h \approx v - zoomfactor \qquad (7)$

$\square\ GetFrame(v),\ curr_h \neq v - zoomfactor \qquad (8)$

(7) Zooming out so reset the view.
(8) Already zoomed out, so do nothing.

$HBZ_{IT} = HBZ\ ||\ left : Hand\ ||\ right : Hand\ ||\ head : Head\ ||\ rect : Rect \qquad (9)$

(9) The interaction technique has several components operating in parallel.

5.1.3 Two Handed Flying CSP

THF =

rel?(l,r) --> SetFrame(l-r) --> THF, l-r > deadzone (10)

□ *rel?(l,r) --> THF, l-r <= deadzone* (11)

(10) If hands are not in the deadzone then base the view on relative hand positions.
(11) The users hands are in the deadzone, so do nothing

RelMovement = RelHand || left : Hand || right : Hand

RelHand(l,r) =

left.posn?x --> rel!(x,r) --> RelHand(x,r)

□ *right.posn?x --> rel!(l,x) --> RelHand(l,x)* (12)

(12) Output the relative hand positions.

$THF_{IT} = THF\ ||\ RelMovement$

5.1.4 Head Crusher Select CSP

Hand =

posn!Frame -->Hand

⊓ *gesture!Frame -->Hand* (13)

(13) Redefine the Hand process as HCS is based on a hand position and a hand gesture.

HCS =

right.gesture?g --> right.posn?r --> head.posn?h -->

$\qquad ModifyObject(select,r,h) \dashrightarrow HCS,\ g = HCS_{pose} \qquad (14)$

$\square\ right.gesture?g \dashrightarrow HCS,\ g \neq HCS_{pose} \qquad (15)$

(14) If the current gesture is the HCS pose then select the object.
(15) Otherwise do nothing.

$HCS_{IT} = HCS\ ||\ head : Head\ ||\ right : Hand$

5.2 CSP Observations

When defining a particular interaction technique within CSP, is it straightforward to check that all the possible actions have been enumerated. This is helpful when evaluating a interaction technique as it can identify possible errors in a techniques specification. As users are especially reliant on system feedback within virtual environments, identifying such deficiencies could help definite more robust interaction techniques.

Another advantage is the trace mechanism which is provided with CSP. Traces can be used to produce traces of users actions. Building this on top of interaction techniques and user models can allow process conflicts to be identified. This can help to refine the techniques so that they more closely match user expectations. There also exists systems for automatic CSP model validation. This increases the usefulness of the notation.

One limitation with CSP is that it deals only with process and not state information. Typically, current state information is generated on-the-fly or passed as parameters. With large models, passing large numbers of parameters can make the notation clumsy.

6 Conclusions

Due to the continuous and highly visual nature of virtual environments, defining salient and useful aspects of interaction is non-trivial. This is especially true as many aspects of VEs are still without concrete definitions and thus are difficult to measure. Research into this unknown territory requires careful consideration. We feel that as the final objective is unclear, beginning research at an abstract modelling level is an appropriate starting point.

The use of Statechart based modelling techniques have been investigated. Unfortunately state based techniques do not show the features of the interaction which make different techniques distinctive. The use of CSP and a process driven approach has been used to try and overcome this limitation.

However, CSP also has limitations when based in highly dynamic virtual environments. In particular, the sense of continuity of action and feedback within particular phases of interaction is not well captured by the discrete nature of the CSP model. Also, virtual environments are typically visually based and some aspects of them can therefore be difficult to model when using text based notations. It is possible that some mix of state and process based techniques will be required to enable effective modelling of the interaction techniques; techniques for dealing with hybrid systems may also provide useful insight [24].

There remains the issue of the user and their cognitive model. To effectively evaluate VEs for usability, some kind of user model will need to be integrated with the system and interaction models. This and the modelling of the visual component of virtual environments are the next areas of interest for the current research.

References

1. Smith, S., Duke, D., Marsh, T., Harrison, M., and Wright, P. *Defining Interaction in Virtual Environments*. 1998, Work in progress at the University of York, UK.
2. Mine, M.R., *Virtual Environment Interaction Techniques*. 1995, University of North Carolina.
3. Kaur, K., A. Sutcliffe, and N. Maiden. *Improving Interaction with Virtual Environments*. in *The 3D Interface for the Information Worker*. 1998. London: IEE.
4. Traetteberg, H. *Modelling direct manipulation with Referent and Statecharts*. in *5th Eurographics Workshop on Design, Specification and Verification of Interactive Systems*. 1998. Abingdon, UK: DSV-IS98.
5. Higgett, N. and S. Bhullar. *An Investigation into the Application of a Virtual Environment for Fire Evacuation Mission Rehearsal Training*. in *Eurographics 16th Annual Conference*. 1998. Leeds.

6. Pausch, R., D. Proffitt, and G. Williams, *Quantifying Immersion in Virtual Reality*. 1997, University of Virginia.
7. Monk, A., *Mode errors: a user-centered analysis and some preventative measures using keying-contingent sound.* International Journal of Man-Machine Studies, 1986. 24: p. 313-327.
8. Andre, A. and A. Degani. *Do You Know What Mode You're In? An Analysis Of Mode Error In Everyday Things*. in *2nd Conference on Automation Technology and Human Performance*. 1996. Daytona Beach, FL, USA: University of Central Florida.
9. Degani, A., M. Shafto, and A. Kirlik. *Modes in Automated Cockpits: Problems, Data Analysis and a Modelling Framework*. in *36th Israel Annual Conference on Aerospace Sciences Conference*. 1996. Haifa.
10. Mine, M.R., F.P. Brooks. Jr, and C.H. Sequin. *Moving Objects in Space: Exploiting Proprioception In Virtual-Environment Interaction*. in *SIGGRAPH 97*. 1997. ACM SIGGRAPH.
11. Pausch, R., T. Burnette, D. Brockway and M.E. Weiblen. *Navigation and Locomotion in Virtual Worlds via Flight into Hand-Held Miniatures*. in *ACM SIGGRAPH '95*. 1995. Computer Graphics.
12. Pierce, J.S., A. Forsberg, M.J. Conway, S Hong, and R Zeleznik. *Image Plane Interaction Techniques In 3D Immersive Environments*. in *1997 Symposium on Interactive 3D Graphics*. 1997.
13. Poupyrev, I., M. Billinghurst, S. Weghorst, and T. Ichikawa. *The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR*. in *UIST 96*. 1996. Seattle, WA: ACM.
14. Chung, J., *Intuitive Navigation in the Targeting of Radiation Therapy Treatment Beams*. 1994, University of North Carolina.
15. Conner, D.B., S.S. Snibbe, and K.P. Herndon, *Three-Dimensional Widgets*, in *Symposium on Interactive 3D Graphics: Special Issue of Computer Graphics 1992*, D. Zeltzer, Editor. 1992, ACM Press: Cambridge, Massachusetts. p. 183-188.
16. Stansfield, S., D. Shawver, and A. Sobel. *MediSim: A Prototype VR System for Training Medical First Responders*. in *IEEE 1998 Virtual Reality Annual International Symposium*. 1998. Atlanta, Georgia: IEEE Computer Society.
17. Gobbetti, E., J.-F. Balaguer, and D. Thalmann. *VB2: An Architecture for Interaction in Synthetic Worlds*. in *ACM Symposium of User Interface Software and Technology*. 1993. ACM Press.
18. Vaananen, K. and K. Bohm. *Gesture Driven Interaction as a Human Factor in Virtual Environments - An Approach with Neural Networks*. in *Virtual Reality Systems*. 1992. Academic Press.
19. Harel, D., *On Visual Formalisms,* in *Communications of the ACM*. 1988, p. 514-530.
20. Leveson, N.G., M.P.E. Heimdahl, H. Hildreth, and J.D. Reese. *Requirements Specification for Process-Control Systems*. IEEE Transactions on Software Engineering, 1994. 20(9): p. 684-707.
21. Jones, P.M., R.W. Chu, and C.M. Mitchell, *A Methodology for Human Machine Systems Research: Knowledge Engineering, Modelling and Simulation.* Transactions on Systems, Man and Cybernetics, 1995. 25(7): p. 1025-1038.
22. Degani, A. and M.G. Shafto, *A Framework for Modelling Human-Machine Interaction*. 1997, CHI 97 Position Paper.
23. Hoare, C.A.R., *Communicating Sequential Processes*. Prentice-hall International Series In Computer Science, ed. C.A.R. Hoare. 1985, New Jersey: Prentise-Hall International.
24. Chaochen, Z., W. Ji, and A.P. Ravn, *A Formal Description of Hybrid Systems,* in *Hybrid Systems III : Verification and Control*, R. Alur, T.A. Henzinger, and E.D. Sontag (Eds). 1996, Springer: Berlin. p. 511-530.