SPECIFYING THE INTERACTION ASPECTS OF A VIRTUAL TRAINING USING VIRTUAL REALITY MULTI-FLOW CRAPHS

C. N. Diplas(1)
cdplus@math.upatras.gr

A. D. Kameas(2)
kameas@math.upatras.gr

P. E. Pintelas(1)
pintelas@math.upatras.gr

ABSTRACT

Advances in Virtual Reality (VR) Systems, Intelligent Tutoring Systems and Agent Technology make it possible to design and develop Virtual Training Environments, where the trainees can immerse themselves and interact directly with the learning domain. This paper presents the Virtual Reality-MFG formal model for the specification of interaction and the design of Virtual Reality Agent-based Training Applications. VR-MFG gives the interaction designer the capability to apply user-centered design and adopt alternatively a task-based or a goal-oriented approach, for the design of a specific Virtual Training Environment (VE). Moreover, allows the designer to split the design effort for any VR Training Environment among a number of multiple micro-worlds that constitute the entire application. Finally, as a case study of VR-MFG application, the interaction aspects of DrIVE virtual environment are specified.

INTRODUCTION

Virtual Reality (VR) applications tend to unify the application program and its user interface. This means that the user interface is transparent and is not distinguished from the "pure" application, as it happens in conventional programs. Virtual Reality uses techniques in order to immerse the user into a computer generated environment where natural behavior is the interaction paradigm.

Into a Virtual Environment (VE) the application functionality and the application interface are not visually or physically separated, but only conceptually. The user interfaces of conventional application programs (even those with Graphical User Interfaces) serve as a representation of that functionality, and are constructed in order to tell the people what the program can do [5].

Moreover, the purpose of VR applications is to provide the user with the capability to perceive with more natural ways the information produced by an application program and interact with complex data. This means that the user's goals and

(1) Division of Computational Mathematics and Informatics and Educational Software Development Laboratory Department of Mathematics, University of Patras, Hellas.

(2) Educational Software Development Laboratory, Department of Mathematics, University of Patras, Hellas.

intentions are translated not to abstract actions (e.g. click a button, roll up/down a slider, etc.) but to more physical movements, directly onto the elements of the VE. This imposes on VE developers the need to provide the user with a unambiguous representation of the goals that must be accomplished and tasks that must be carried out inside the VE.

Every user's goal and intention affect one or more specific objects in an application program: e.g. a database which has to be modified, a document which has to be deleted, a site on the web that must be accessed, or even better a learning unit that must be experienced by the user into an educational program. But, at the same time a problem arises: *what are the appropriate controls and widgets that must be used by the user in order to accomplish the desired tasks in a GUI environment?*

Concerning VEs, this problem is transposed on *how to interact with the VE's objects* that are also, the subject of the user's goals and intentions. The user has not to worry about specific controls that exist or not in the environment but has to concentrate solely on his specific goals.

In the area of Distributed Artificial Intelligence (DAI) and Intelligent Agents, a lot of research has been presented on models based on formal languages and on architectures which are based on these models. Most of these modeling approaches emphasize on the properties, the classification, the social behavior, the goals, the tasks, the actions and the communication of agents and include mostly formal languages (e.g. KQML, April), algebra of actors and languages based on temporal logic [7], [1], [13], [16], [18], [24]. In addition, Artificial Intelligence issues are involved more and more in VR applications, introducing their own constraints that are related to the interaction aspects of the VR application.

VEs for training [14], especially those combined with "real world" simulations seem to be one of the new ambitious trends on the domain of Computer Assisted Instruction (CAI). Computer generated virtual agents may exist as autonomous objects, with which users may interact into a Virtual Training Environment (VTE). Functions that are common to intelligent tutoring systems (ITSs) (e.g. learner modeling, coaching) are assigned to individual intelligent animated agents, rather than being part of the environment as a whole. Also, the specification of tutor-learner interaction sessions leads to the need of establishing techniques for specifying models of instruction.

The domain of VR systems into which interactive intelligent agents with pedagogical capabilities are incorporated, is the current trend in the area of Virtual Training Environments. The approaches that have been proposed focus: on the development of automated agents with pedagogical capabilities that can participate in training exercises [23], [11], and on the modeling of the computer-generated tutors pedagogical aspects [10] (e.g. plan recognition and execution, demonstration and explanation capabilities). Although these approaches have led to efficient systems, it will be extremely useful for the interaction designer to take advantage of an interaction specification model which is adapted to the needs of three-dimensional agent-based VTEs, incorporating features that capture both the user-system (external communication) and the agent interaction (internal communication) aspects, and the pedagogical issues of the produced VR systems, as well.

Virtual Reality Multi Flow Graphs (VR-MFG) is a specification model which is defined as an extension of IMFG [12] and can be used for the specification and design of Virtual Reality Agent-based Training Applications. It incorporates the cognitive features of IMFG, and the powerful analysis techniques of Petri Nets. The VR-MFG enriched interaction model is derived from the IMFG and adopts the basic formalism of its predecessor. Virtual Environments due to the three-dimensional graphics they provide, are the ideal platform for representation of the elements used by IMFG (boxes, circles, arrows etc.) and the visualization of the concepts these graphical elements reflect. On the other hand, the graphical notation of VR-MFG (mostly adopted by IMFG) makes the specific model one of the most suitable, among other similar approaches [9] for interaction specification in VEs, since the three-dimensional graphs provide a high degree of supervision as shown in [26]. Since this work is in progress, the pedagogical characteristics of the VR-MFG model are not exhaustively presented in this paper, nevertheless the way those pedagogical features are represented is described in the next section.

In section two, the description and the analytical definition of the VR-MFG model is presented, along with a short discussion about the application of the model and the extended design framework into which the VR-MFG belongs. Section three is a case study of VR-MFG application, and includes the interaction specification graphs for DrIVE virtual environment. The paper concludes with relevant further work on the domain.

VIRTUAL REALITY – MFG DESCRIPTION

As a Virtual Environment consists of *interactive* (active) and *non-interactive* (passive) virtual elements (e.g. objects, concepts, abstractions, information), the VR-MFG which models this VE consists of active (actors) and passive (links) components. Although there is a close relation between the VE's elements and the VR-MFG components, there is an indirect correspondence between them. An active VR-MFG component does not refer directly to an *interactive element* of the VE, but to the task, goal or high and lower level action into which this element is involved.

The correspondence between a passive VR-MFG component and a *non-interactive* VE's element, is analogous, since, a passive VR-MFG component refers to the visualization of the different information flows that occur in a Virtual Reality interactive application.

As proposed in AVIARY [20] architecture, everything that lies inside a Virtual Environment is treated as an *object.* Then each single object which is presented to the user is an *artifact,* and objects that cause the artifacts are called *demons* [19]. In VR-MFG, active and passive components are used in order to describe these *demons* and *artifacts* respectively. In addition, the data or/and control information that flows into the VE are represented by data or/and control structures into VR-MFG.

1. Definition of the model

Analytically, the components of Virtual Reality-MFG model are:

- **actors** (that correspond to the actions, tasks or goals of the VE's elements), which model the interactive responses that must be performed by the agents that participate in a VE, as a consequence of the occurrence of an event. Events may be caused by other agents, since the user, the objects of the VE and the VE itself, are all agents that act into a common space. Actors are always preceded and followed by,
- **links** (that correspond to the non-interactive VE's elements), which describe the situation that precedes and that results from a user action, through the storage of,
- **tokens** (that correspond to the data or/and control information that exist into the VE), which represent abstract data or control structures that are produced or consumed by the VR-MFG components.

All the actors that are ready-to-fire (that is, which may fire after the next event) are maintained in the *actor-ready list*. Moreover, an actor can be viewed as an integral goal, which can be achieved by the satisfaction of a number of sub-goals.

Actors are described by:

- a *name*,
- a *set of input* and a *set of output links*, which precede and follow the actor,
- a *set of firing rules*, which represent the actor's behavior, since the left hand side forms the pre-conditions (that are kept in its input links) and the right hand side includes the post-conditions (that are kept in its output links),
- a *method*, which represents the lower level actor's functionality, that is, how the actor handles its input data and produces its output ones.
- a *type*. There are four types of actors, namely:

1. *Action actors*: they represent a single response which is performed by a VE agent. These actors do not have any VR-MFG represented internal structure. The rules part of each action actor defines how this single task is implemented. VR-MFG does not give a clear description of the task to be done, but specifies the goal decomposition (or the task analysis) in order for the goal to be achieved (or the task to be completed). Action actors define the way the VE's agents interface with the domain-dependent functional core of the application, via single interactive responses.
2. *Context actors*: their internal structure represents the task or goal decomposition into sub-tasks or sub-goals, via a number of other context or action actors.
3. *Guide actors*: similar to Library IMFG actors, are used to represent the way the task or goal decomposition is achieved. AND and OR decomposition are provided, since these two fundamental actions can model any task or goal decomposition.

4. *Virtual actors*: used for a graphical grouping of actors, without any other significance, but may serve as reusable components during design process.

Virtual Reality applications are characterized mostly by their interactivity requirements that must be fulfilled in order to provide the user with the capability to perceive the virtual environment, with more natural ways. But, as shown and statistically measured in [15], most users focused on what there was to do, and what is already done, inside the virtual environment and not only on how to do it. The contribution of VR-MFG into this direction is that the action actors can be used either to describe the computational capabilities of the application (e.g. what the VE's agents can do, including the user, as well), or to link VR-MFG model with another specification model that describes the lower-level computational tasks that constitute the application (e.g. how VE's agents communicate with the external environment-devices or with each-other, or which exact way is used by the user-student trying to interact with the learning domain into a Virtual Training Environment).

Links are described by:

- a *name*,
- a *set of input* and a *set of output* actors that produce and consume the tokens stored in this link,
- a *method*, which is performed upon the link's tokens, and
- a *type*. The definition of link types distinguishes among the type of tokens they store. Each link type stores a specific kind of tokens. This allows the designer to model the different types of information (e.g. data, control) that exist inside a VR Application, and permits system design from alternative perspectives.

VR-MFG links are typed so that the different information flows that occur in a VR application are distinguished and each information flow can have its own visualization.

The seven types of links, are:

1. *Event links*: describe the events that are caused by the agents that participate in the VR application. Users are also treated as agents in VR-MFG, so event links can be used in order to describe any *external* or *internal* communication of events. In VEs, events may be composite, having their own existence, unlike applications that use a 2D GUI, where events can be caused by simple actions (e.g. click, scroll, key-press, etc.). For example, event generation process in numerous cases [4] (e.g. where gesture interaction methods are used), includes an internal structure. Moreover, this is recommended by a number of diverse interaction methods which have already proposed in [3], [8], [22]. VR-MFG provides *event links decomposition*, so the designer can explicitly specify how events can be caused, representing the internal structure of event links.
2. *Perception links*: a special kind of event links, which are used in order to represent system responses that are directed to the input-output devices, in order to

provide the user of the VR application with the capability to perceive with natural ways of special VE responses (e.g. haptic feedback, position orientation feedback activities etc.). These are event links with internal structure, that represent any cognitive perceptual state of the user agent and the tokens they contain are produced and consumed exclusively by the user (or the user agent). *Perception links* are designed in order to make VR-MFG model adaptive to any kind of interaction technique (e.g. direct or indirect manipulation of VR objects, immersion techniques where input is entered via sensors and output is processed by advanced hardware interface devices) and permit platform-independent interaction specification.

3. *Condition links*: represent the global or local conditions that precede and result from any agent action that takes place into the VE. Consequently represent priority of execution and availability of actors. Moreover condition links describe whether any of the VE agents is ready to process another agent's action, that will lead to the achievement of a subgoal, or to the completion of a specific task, into the VE.
4. *Data links*: represent the data or control flow, into the VR application. Moreover, they represent the content of messages that pass between the participating agents.
5. *Context links*: represent the context into which, a number of interactions are performed. Consequently, context links describe the context to which a specific goal or task belongs and indicate whether a major goal is decomposed into subgoals, so a new "session" starts for the accomplishment of this sub-goal. Moreover, context links contribute to the representation of the *memory* and the *knowledge* issues which are of major importance for any VR agent-based Application. Every actor that belongs inside a specific context *knows* the goal of all the other actors that belong in the same context and the VR-MFG can model long-term memory by maintaining the actor ready list and the content of context-out links, since the short-term memory is represented directly inside the current context.
6. *Communication links*: model the effects that the separate micro-worlds existing inside the entire VR application may have on one another, since there exists a separate VR-MFG for each virtual micro-world.
7. *Learning links*: represent the learning issues which rule the training interactions and the student-trainee dialogue. These links are not completely integrated into VR-MFG yet, but are going to be founded upon alternative instructional strategies and provide representation of autonomous agents with instructional capabilities [17].

2. Application of the model

VR-MFG, being graphical tool, has its special symbolism which is shown in Figure 1.

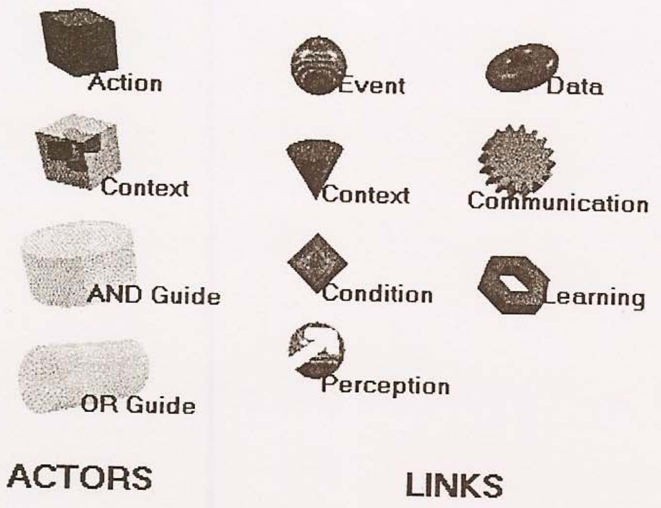

Figure 1: VR-MFG symbolism.

Any VR Application can be viewed as a collection of different VEs (micro-worlds) that constitute the entire application. These multiple worlds may be concurrently active, just like a conventional GUI application, where several windows are open, running different applications but only one can have the user focus. The aim of VR-MFG is not to specify the concurrency for the entire application, but to split the design effort for concurrency, among these multiple worlds.

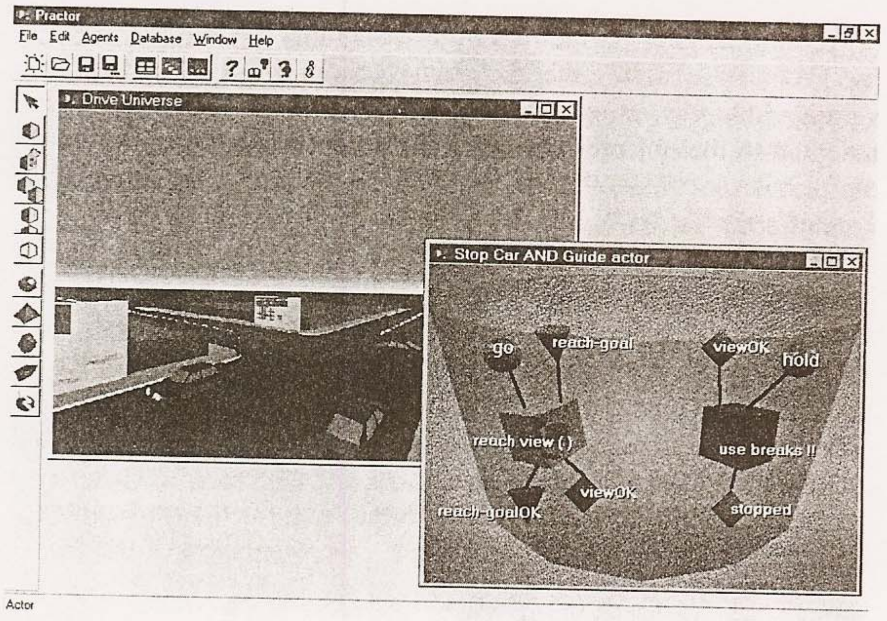

Figure 2: Snapshot of the common virtual workspace where abstract objects (the components of the VR-MFG) are associated with "actual" objects of the target virtual environment and appropriate agent templates with instructional capabilities are applied.

VR-MFG is used as the underlying model of the Interaction Specification Workspace (ISW) architecture, which is under construction from the authors of this paper. The designer is provided with the capability to associate the abstract objects (the components of the VR-MFG) with "actual" objects of the target virtual environment kept in the object database, and apply a number of agent templates, inside a common virtual workspace (Figure 3).

A CASE STUDY OF VR–MFG APPLICATION

DrIVE [6] is a virtual training environment intended to train novice car-drivers in common driving situations. It is based on desktop VR, with a minimum of requirements in processing power and data storage. DrIVE consists of three main parts: *driving lessons*, *tests for the trainee*, and *free driving* with on-line guidance.

The aim of the application is to transfer experience on the domain of driving behavior, which can be done using the synthetic experience that virtual environments are able to provide rather than actual practicing which involves obvious dangers. To this end, feedback from users of this first prototype is extremely encouraging and several ideas for its improvement have been suggested to the design team.

DrIVE has been developed with Superscape VRT software [25]. The physical properties were attached to the objects which constitute the DrIVE environment using the appropriate editors.

In the first system version the interactive properties were extracted informally and code was assigned incrementally and directly into the objects using SCL (a C-like programming language with event driven code execution) in order to implement these properties. In the second system version, the third part of *free driving* has been designed afresh, using VR-MFG for the interaction design, before any code was assigned to the objects. Then, the implementation of this part was realized, using this formal specification. The benefit was twofold: evaluation of VR-MFG and a substantiated system.

The Crossroads example refers to the third part of the application where the user-trainee is the driver of one of the virtual environment's cars. His car reaches the crossroads, and two other cars are coming from the opposite directions.

The user plan-goal decomposition approach will be applied in order to specify the way the user must pass the crossroads safely, which is the main user-goal. This goal is directly decomposed in three sub-goals: *Stop* the car before the crossroads, *Check* and give priorities, and *Pass*.

Three representative graphs will be presented, one for the representation of the overall goal (*Cross-Goal*) and the other for the representation of the *Stop*-subgoal, and one for the *Hold action* event link generation. In Figure 3 the overall goal is decomposed into the three sub-goals, using the *AND guide actor* (represented with a cylinder which includes all the other VR-MFG components).

The *Stop action* event link (represents the user actions in order to stop the car) and the *Stop-subgoal* context input link (shows that the user intents to stop the car)

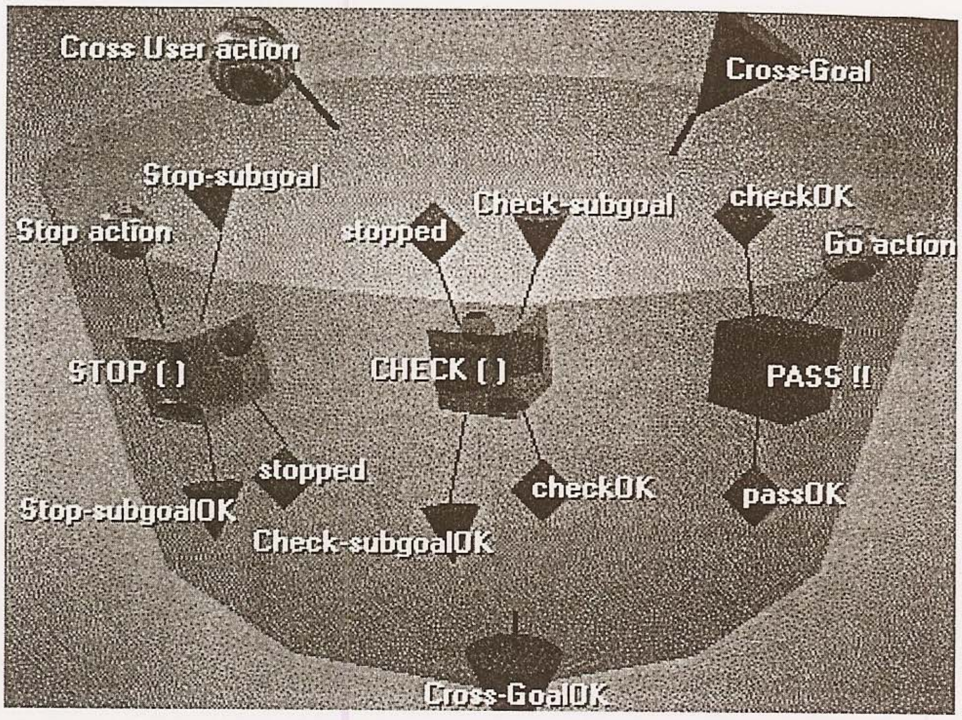

Figure 3: The *Cross-Goal* represents the user-intention, and the *Cross User action* the actions he must perform to achieve this goal. Inside this *AND Guide actor,* there exist the *STOP ( )* and the *CHECK ( )* context actors, and the *PASS !!* action actor.

form the *STOP ( )* context actor set of input links. Also, the *stopped* logical condition link (shows that the car actually stopped) and *Stop-subgoalOK* context link (the existence of a token indicates that the *stop the car* subgoal is being satisfied) form its set of output links.

The *stopped* condition is then checked by the *CHECK ( )* context actor which can be further decomposed in order to fulfill the *checkOK* condition, permitting (along with Go action) the *PASS !!* action actor to fire.

In Figure 4 the *STOP ( )* context actor is decomposed. *STOP ( )* context actor includes the *REACH VIEW ( )* context actor and the *USE BRAKES !!* action actor. The user actions in order to drive the car in a position that gives him an appropriate view of the crossroads, are represented by the *Go* event link. The actor *REACH VIEW ( )* fires and a token is produced in the *viewOK* condition. This token is consumed by the *USE BRAKES !!* action actor, if there is a token in the *Hold action* event link, also. Then a token is produced in the *shakeFeedback* perception link (which can be used in order to provide the user with real feedback through an advanced hardware interface device, e.g. a data-glove or a cyber-data-chair), the *stopped* condition link and in the *Stop-subgoalOK* context link. Five more graphs, with almost the same complexity with those presented, are enough in order to complete the entire interaction specification and design for the *Crossroads* example.

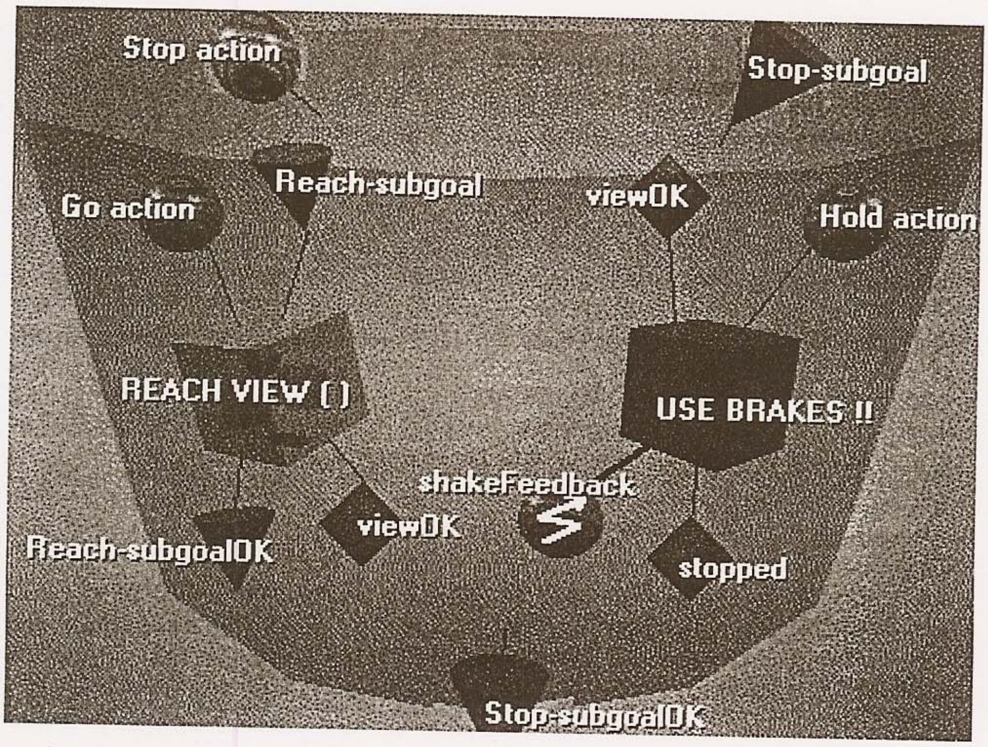

Figure 4: The *STOP ()* context actor set of input and output links are the same with these presented inside the AND Guide actor of the previous figure. *STOP ( )* includes the *REACH VIEW ( )* context actor and the *USE BRAKES !!* action actor. The condition *viewOK* that exist as an output link of *REACH ( )* actor and as an input link of *USE BRAKES !!* is responsible for the preservation of the correct interaction sequence.

VR-MFG provides decomposition capabilities for the event links. This feature is applied for *Hold action* event link. The user has two alternative ways to generate this event: either by using a pointing device (e.g. mouse, data-glove) or directly with the keyboard. This is represented using OR decomposition (the horizontal cylinder for the *OR Guide* actor) and by the two action actors that exist inside this Guide actor: *STOP INCREMENTALLY !!* and *STOP IMMEDIATELLY !!*. The firing of at least one action actor, produces a token to the *velocityZero* condition link, so the *Hold action* event is completely specified.

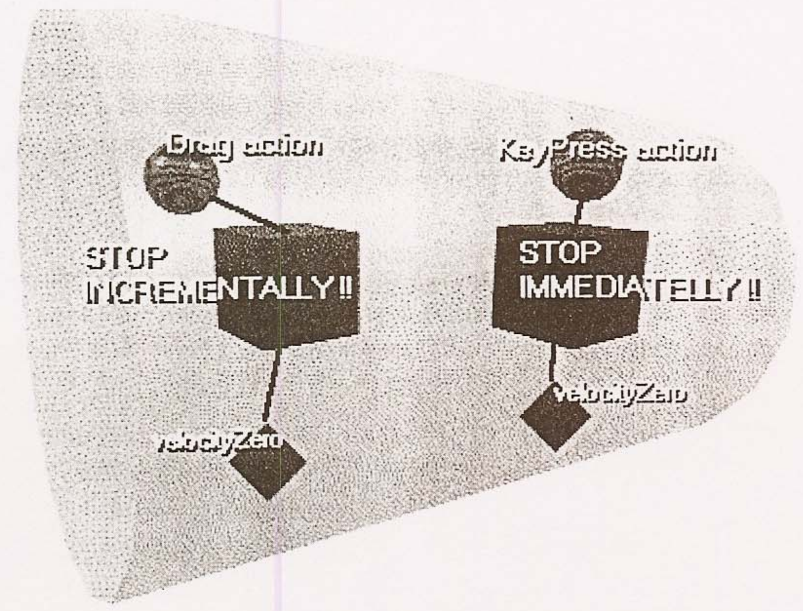

Figure 5: The *Hold action* is decomposed into two action actors: *STOP INCREMENTALLY !!* and *STOP IMMEDIATELLY !!*, using *OR Guide* actor, providing the designer with the capability to specify the interaction style that will be used in order to reduce the car's velocity down to zero.

CONCLUSIONS

Virtual Reality-MFG is an IMFG-based specification model for the specification and design of Virtual Reality Agent-based Training Applications which incorporates the cognitive features of its predecessor, the powerful analysis techniques of Petri Nets, and features found in various VR interaction techniques and multi-agent paradigms [2], [21].

VR-MFG allows the designer to split the design effort for any VR training environment, among a number of multiple micro-worlds that constitute the entire application, and to specify the different flows of data and control, inside the VR application. VR-MFG gives the interaction designer the capability to apply user-centered design and adopt alternatively a task-based or a goal-oriented approach, for the design of a specific VE. The contents of the actor-ready list, along with conditions enables the control of the global VR application state.

The VR-MFG graphs presented in this paper describe the interaction between the user and the system for a crossroads situation inside the DrIVE training application. Future development includes the incorporation of interaction styles that are common to intelligent tutoring systems which can be assigned to individual intelligent animated agents, rather than being part of the environment as a whole.

Moreover, VR-MFG is application and domain independent, and can be used as an underlying model for the specification and design of VR systems that require "true" immersion via the perception links and the link decomposition which are provided. VR-MFG is implemented in the framework of the Interaction Specification Workspace (ISW) architecture, as its underlying interaction specification and design model, enabling fast analysis and prototyping of Virtual Reality Agent-based Training Environments.

REFERENCES

[1] Bates, J. (1994). The role of emotion in believable agents. Communications of the ACM, 37(7):122-125.

[2] Beale, R., Wood, A., "Agent-Based Interaction", in *People and Computers* IX: Proceedings of HCI '94, Glaskow, UK, August 1994, pp.239-245.

[3] Benford, S. et al., "From Rooms to Cyberspace: Models of Interaction in Large Virtual Computer Spaces", in *Interacting with Computers* (Butterworth-Heinmann), 1993.

[4] Bolzoni, M. L. G., "Eliciting a Context for Rules of Interaction: A Taxonomy of Metaphors for Human-Objects Communication in Virtual and Synthetic Environments", Proceedings of the 2nd UK VR-SIG and Contributors, December, 1, 1994, Reading, UK, pp. 78-87.

[5] Burks, L., "Information Architecture: The Representation of Virtual Environments", Harvard University Graduate School of Design, Thesis Document, May 1996.

[6] Diplas C., Giakovis D., Pintelas P., "DrIVE: A Virtual Training Environment For Driving Behaviour", in Proceedings of the First International Conference on Computers and Advanced Technologies in Education (CATE 96), pp.191-200, March 18-20, 1996, Cairo, Egypt.

[7] Finin, T., Weber, J,. Wiederhold, G., Genesereth, M., Fritzson, R., McKay, D., McGuire, J., Pelavin, R., Shapiro, S., and Beck, C. (1992). Specification of the KQML Agent-Communication Language. Technical Report EIT TR 92-04, Enterprise Integration Technologies, Palo Alto.

[8] Fuchs, J., and Bishop, G., "Research Directions in Virtual Environments. An Invitational Workshop on the Future of Virtual Environments". TR92-027, March 1992, The University of North Carolina at Chapel Hill, Department of Computer Science.

[9] Harrison, M., D., and Duke, D., J., "A Review of Formalisms for Describing Interactive Behavior", Amodeus Project Document: System Modelling/WP28, January 1994.

[10] Hill, R. W., Johnson, W. L., "Situated Plan Attribution", Journal of Artificial Intelligence in Education, (6)1, pp. 35-67, 1995.

[11] Johnson, W. L., "Pedagogical Agents for Virtual Learning Environments", Proceedings of the International Conference on Computers in Education, pages 41-48, Singapore, 1995.

[12] Kameas, A., "A Formal Model for the Specification of Interaction and the Design of Interactive Applications". Ph.D. Thesis, Department of Computer Engineering, University of Patras, Greece, 1995.

[13] McCabe, F. G. and Clark, K. L. (1994). April – agent process interaction language. In Wooldridge, M. and Jennings, N. R., editors, Pre-proceedings of the 1994 Workshop on Agent Theories, Architectures, and Languages, pp. 280-296, Amsterdam, The Netherlands.

[14] Mikropoulos, A., Diplas C., Giakovis, D., Halkidis, A., Pintelas, P., "Virtual Reality & Education: New Tool or New Methodology?", Proceedings of 2nd Conference on Informatics in Education, pp.57-67, November, 11-13, 1994, Athens, Hellas.

[15] Pausch, R., et al, "Disney's Aladdin: First Steps Toward Storytelling in Virtual Reality", Proceedings of Computer Graphics, Annual Conference Series, pp. 193-203, 1996.

[16] Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In Fikes, R. and Sandewall, E., editors, Proceedings of Knowledge Representation and Reasoning (KR&R-91), pp. 473-484. Morgan Kaufmann Publishers: San Mateo, CA.

[17] Rickel, J., and Johnson, W., L., "Integrating Pedagogical Capabilities in a Virtual Environment Agent", to be presented in First International Conference on Autonomous Agents, February 1997.

[18] Shoham, Y. (1993). Agent-oriented programming. Artificial Intelligence, 60(1): 51-92.

[19] Snowdon, D. N., West, A. J. and Howard T. L. J., "Towards the next generation of Human-Computer Interface", Proceedings of Informatique '93: Interface to Real & Virtual Worlds, 26-26th March 1993, Montpellier, France, pp. 399-408.

[20] Snowdon, D., West, A., "The AVIARY Distributed Virtual Environment", Proceedings of the 2nd UK VR-SIG and Contributors, December, 1, 1994, Reading, UK, pp. 39-54.

[21] Steels, L. (1990). Cooperation between distributed agents through self organization. In Demazeau, Y. and Muller, J.-P., editors, Decentralized AI – Proceedings of the First European Workshop on Modelling Autonomous Agents in Multi-Agent Worlds (MAAMAW-89), pp. 175-196. Elsevier Science Publishers B.V.: Amsterdam, The Netherlands.

[22] Sturman, J., and Zeltzer, D., "A Design Method for 'Whole-Hand' Human-Computer Interaction", ACM Trans. On Information Systems (11-3), July 1993, pp. 219-238.

[23] Tambe, M., et al, "Intelligent Agents for interactive simulation environments". AI Magazine, 16(1), pp. 15-39, Spring 1995.

[24] Thomas, S. R. (1993). PLACA, an Agent Oriented Programming Language. PhD thesis, Computer Science Department, Stanford University, Stanford, CA 94305.

[25] VRT 3.60, Superscape Ltd., Reference Manual.

[26] Ware, C., Franck, G., "Viewing a graph in a Virtual Reality Display is Three Times as Good as a 2D Diagram", Proceedings of 1994 IEEE Conference on Visual Languages, S. Louis, Missouri, USA, October, 1994, pp. 182-183.