Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

TEMPORALLY PARALLEL COUPLING OF DISCRETE SIMULATION SYSTEMS WITH VIRTUAL REALITY SYSTEMS

Steffen Strassburger

Fraunhofer Institute for Factory
Operation and Automation IFF
Sandtorstrasse 22
39106 Magdeburg, GERMANY

Thomas Schulze

School of Computer Science
University of Magdeburg
Universitätsplatz 2
39106 Magdeburg, GERMANY

Marco Lemessi
Gordon D. Rehn

Industrial Engineering, Simulation Group
Deere & Co.
One John Deere Place
Moline, IL 61265, U.S.A.

ABSTRACT

The coupling of commercial discrete simulation systems with virtual reality (VR) systems opens new possibilities for the temporal interplay of product and process design. Among the possibilities is establishing *virtual training centers* aimed at shortening product ramp-up times. Up to now, coupling has only been employed sporadically because of the need to preserve the autonomy of the tools used. This paper focuses on the problems of synchronization as one of the important basic tasks when coupling discrete simulation and VR systems. Existing techniques of synchronization are examined for their suitability for coupling and a method of synchronization based on self-adapting buffer sizes is described.

1 MOTIVATION

The visualization of simulated processes has successfully augmented simulation technology for many years. The majority of commercial simulation systems provide the respective options for visualizing modeled and simulated systems. Thus, over time, a broad spectrum of objectives for coupling simulation and visualization has developed. The objectives range from simple post-run visualization of simulated processes up through training and thus interaction in simulation-based virtual reality (VR) worlds.

VR itself is defined as a computer-generated three-dimensional environment created by virtual environment systems, which can be interactively experienced and manipulated by the participants (Barfield and Furness, 1995). According to Stuart (2001), VR provides a human-computer interface capable of providing interactive immersive multisensory 3-D synthetic environments.

So far, the coupling of simulations with VR worlds has been intensely developed in training simulators, flight simulators being an example here. Usually these are special and single-purpose hardware and software systems.

Modeling and simulation applications in the fields of logistics and production are done exclusively using commercial discrete simulation systems. Visualizations of simulated processes are an inherent part of corresponding simulation projects. Many simulation systems (e.g., QUEST, Automod, and others) already provide sophisticated and integrated 3-D visualizations. However, they commonly lack features of interactivity required towards the objective of providing training capabilities.

Up to now, couplings of autonomous simulation systems with VR systems have only been sporadically employed in this field of application (Kibira and McLean 2002, Bergbauer 2002). Existing solutions are typically limited to uni-directional communication at runtime.

A coupling in this application world enables new forms of access by complex interactions between the model user and the simulated objects. This generates new potentials for the temporally parallel interplay of product and process design, for establishing virtual training centers and thus for shortening product launch times (Dorozhkin et al. 2004).

The reasons for the sporadic utilization are the autonomy of the commercial tools used for simulation and visualization and insufficient interoperability between the systems. Greater proliferation of this coupling will only be possible when this autonomy of the tools remains preserved in the solutions being developed.

A primary task when coupling simulation with VR is the synchronization of both tools, i.e. the time advance in both tools must be coordinated. Existing approaches to synchronization are only partly suited for this coupling since

- user actions within the VR can also affect the system status of the simulation,
- both tools operate autonomously and on different platforms, and
- a ”fluid” image generation on the VR side has to be ensured.

In order to satisfy these specific requirements, the authors have modified the method of buffered visualization commands frequently applied in this field. This new method is characterized by adaptive control of the buffer size.

In this paper, the term simulation refers to discrete simulation systems. This restriction is valid since discrete simulation systems dominate many fields of application.

The remainder of this paper is structured as follows: Section 2 describes alternatives for coupling simulation and visualization in order to classify the form of coupling of simulation with VR. Next, the requirements from VR applications on synchronization are specified and different forms of synchronization are discussed and analyzed for their suitability for coupling simulation with VR. Lastly, the methods of adaptive buffer size are explained and placed in context with an application.

2 ASPECTS OF CLASSIFICATION IN COUPLING

To describe the various forms of coupling simulations and visualizations, a classification is proposed (Table 1).
This classification is based on four different features:

- temporal parallelism (please refer to Holten-Lund (2001) for a complete definition of *temporal parallelism*),
- interaction,
- hardware
- visualization tool autonomy.

Two alternative characteristics are applied to each attribute. This classification was undertaken in order to be able to classify the coupling of simulation and VR.

From the abundance of theoretically possible combinations of feature characteristics, the forms of coupling listed in Table 2 are presently used in the field of discrete simulation. It can be seen, that most examples are based of unidirectional couplings of simulation and VR.

Table 1: Classification Attributes and Their Characteristics

Feature	Characteristic	
Temporal Parallelism Temporal parallelism between simulation and visualization	**Concurrent** Simulation and visualization run temporally parallel	**Post-run** Visualization runs temporally after the simulation
Interaction Interactions between simulation and visualization	**Bidirectional** Simulation and visualization each react to the other tool’s commands	**Unidirectional** Only visualization reacts to the simulation’s commands
Hardware Platform Hardware platforms on which the simulation and the visualization operate	**Monolithic/ Homogeneous** Simulation and visualization run on one platform	**Distributed** Simulation and visualization operate on different hardware platforms
Visualization Tool Autonomy	**Integrated** Visualization tool is integrated in the simulation tool	**External** Visualization tool works independent of the simulation tool

Table 2: Feature Characteristics with Examples of Coupling

Feature Characteristics	Examples
Post-run / unidirectional / monolithic / external	Visualization tool Proof Animation (Proof Animation 2005)
Post-run / unidirectional / distributed / external	Coupling of SLX simulations with visualizations in a cave (Dorozhkin, Lemessi, Rehn and Vance 2004)
Concurrent / unidirectional / monolithic / integrated	Visualizations in the tools eM-Plant (eMPlant 2005) and ARENA (ARENA 2005)
Concurrent / unidirectional / monolithic / external	Visualization tool Concurrent Proof (Proof Animation 2005)
Concurrent / bidirectional / distributed / external	Coupling of eMPlant simulations with VR (Franke 2004, Bergbauer 2002)
Concurrent / bidirectional / monolithic / {integrated, external}	Training simulators for pilot training

The bidirectional coupling of simulation with VR requires *temporal parallelism* (concurrency) between the simulation and the visualization because the simulation affects the visualization and the user actions affect the simulation through the VR system.

Dorozhkin et al. (2004) describes a coupling of a discrete simulation and a VR system but it is impossible for the user in the immersive virtual reality environment to influence the simulation.

The crucial feature for coupling simulation and VR is bidirectional *interaction*. The work of Franke (2004) and Bergbauer (2002) address this feature, yet the range of interactions from the VR side is extremely limited.

The simulation systems operate predominantly on Windows-based PC platforms and the VR systems reside overwhelmingly on Unix or Linux-based workstations. For this reason, the coupling being analyzed is labeled a distributed mode on inhomogeneous *hardware platforms*.

Simulation and VR systems are complex stand-alone software systems, which must also retain their autonomy during coupling. Retention of the systems' autonomy is a crucial prerequisite for successfully coupling simulation with VR systems. Existing VR or simulation systems must be coupled with other existing VR or simulation systems. This presupposes the autonomy of the tools used.

The above defines the general requirements for coupling simulation and VR. The specific requirements are described below.

3 SPECIFIC REQUIREMENTS ON COUPLING FROM VR APPLICATIONS

Virtual reality is understood as a computer-generated 3-D environment in which one can immerse oneself as user and perceive this artificial world as a close approximation of reality. As user, one is part of this world and can interact with it. (Based on VDI 2003.)

Solutions exist, which claim to link simulation with VR (Whitman et al. 2002, Kibira and McLean 2002). These applications are based on the utilization of 2-D computer interfaces such as monitor, keyboard and mouse. These features are used to view 3-D models of the simulated environment. The interactions used do not fundamentally differ from the standard interactions in a 2-D representation of the simulated environment.

Initial approaches to coupling simulations with VR systems in fields of industrial engineering application are described in (Franke 2004, Bergbauer 2002, Dorozhkin et al. 2004). Other previous work connects VR and simulation, but stays unidirectional in terms of interactivity and communication (Jessen 2001, Ritter et al. 1998) and does not generally allow feedback from the VR into the simulation at runtime.

When simulations are coupled with VR applications, three problems have to be resolved:

- *Mapping* between simulation and VR models,
- *Data exchange* between the two models during the execution
- *Synchronization* of the two time-bound systems.

The first two problems can be resolved using familiar approaches. As for the third problem, incorporating bidirectional interactions makes special demands on synchronization. Identical simulation and visualization times are desirable so that interactions can be reflected in either system at the time they were triggered. Modifications in the VR world made by the user at time t must also be reflected in the simulation world at time t.

For a clearer understanding, the following definitions shall be given:

- *Simulation time* refers to the current time value of the simulation clock,
- *Visualization time* refers to the current value of the visualization clock, and
- *Real time* refers to the wallclock time.

Identical times in both the simulation and the virtual world cannot be achieved. A disparity between the times has to be accepted. Generally, the simulation time is larger than the visualization time since the simulation chiefly affects the visualization. Due to the bidirectional interactions, the simulation model must be able to react to interactions (external events), the timestamp of which is less than or equal to the current time of the simulation model.

The simulation time can be stopped in the simulator when the two times become too far removed from each other. In contrast to this, the visualization time advances proportionally to the real time and "stopping" the visualization time is impossible.

It is unacceptable for VR applications when the simulation time is smaller than the visualization time. Users do not accept "resets" in the visualization model.

To couple simulation with VR, synchronization methods have to be used, which ensure no or only a slight time difference between the two systems. The value of this difference must be selectable dependent on the transmission times in the network and dependent on the factor of the temporal parallelism of the visualization time with the real time. A fast motion visualization does not let the user recognize much of a difference between the two times. In contrast to this, a smaller difference must be guaranteed when a visualization is in slow motion.

The following section examines existing synchronization methods with regard to applying them to coupling simulation with VR.

4 PRINCIPLES OF SYNCHRONIZATION FOR THE TEMPORALLY PARALLEL COUPLING OF SIMULATION AND VISUALIZATION

Synchronization is an extremely important feature for the interaction of two time-bound systems, each with their own paradigms for the advance of time.

In general, discrete event-oriented simulation tries to update time as quickly as possible, with the time advancing in non-equidistant periods.

In contrast, visualization is characterized by a quasi-continuous progression of time in periods that are equidistant and proportional to real-time. Synchronization aims at a virtually equal time value in both systems.

If T designates the real time, $t_{SIMU}(T)$ the simulation time at real time T, $t_{VISU}(T)$ the visualization time at real time T and $\Delta t(T)$ the difference between simulation and visualization time at real time T, then the following equation applies:

$$t_{SIMU}(T) = t_{VISU}(T) + \Delta t(T)$$

The visualization time $t_{VISU}(T)$ can be calculated with the following equation, c being a factor of proportionality of the visualization time at the real time T:

$$t_{VISU}(T) \approx c * T$$

Methods for synchronizing time-bound components in distributed systems have been developed in recent years. These include special methods for distributed simulation if the components are only simulations and more general methods as are used to couple heterogeneous components in the High Level Architecture HLA.

The existing methods are classified as follows.

4.1 Synchronization Using Real-Time

Time advances uniformly in both systems (see Figure 1). Frequently both systems then operate proportional to real-time. In this case, Δt strives for zero. This approach is typical for very close coupling of simulation and visualization. Interactions exchanged between the two systems are processed whenever they arrive. This approach is theoretically suitable for coupling simulations with VR. In practice, it could be used for simulation systems which advance their simulation time proportional to real-time. Adding such a mechanism to simulators which do not yet offer it will entail additional overhead in the simulator. Also, the approach is limited to very fast and efficient simulators.

Another inherent problem which must be taken into account is the following: The basic principle in discrete event simulation is that all state-changes in the model are timeless, i.e., they do not consume *simulation time*. However, the calculation of the change in the model does consume *real time*, because the processor may have to perform complex calculations.

If TR_i designates the real processor time needed for executing an event at simulation time t_i and ΔT designates the interval for time advancement of the simulator, the following relation must hold true:

$$TR_i < \Delta T$$

The time for processing an event must be less then the time advancement. This condition can be fulfilled for sufficiently sized ΔT. However, this contradicts to a smooth animation, for which a smaller ΔT is better suited.

For complex models TR_i can reach values so that

$$TR_i > \Delta T$$

In that case the user will notice an unacceptable interruption in the visualization. Therefore the described approach does not seem optimal for the desired applications described in this article.

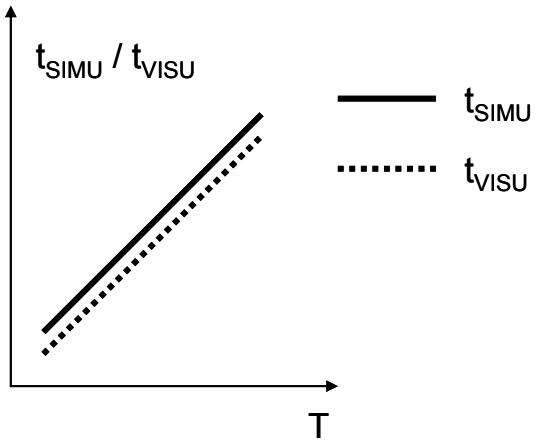

Figure 1: Time Advance with Real-Time Synchronization

4.2 Synchronization Using Logical Time

In this form, time advancement in both systems is coordinated based on their logical simulation clocks. Typically, time advancements have to be requested from a central instance which grants the individual systems a time advance. The synchronization methods provided by the High Level Architecture (HLA) provide these mechanisms (Ritter et al. 1998). In special cases, a central authority can also be dispensed with. Then the systems are synchronized directly on the basis of established synchronization algorithms from distributed simulation.

Two forms of this approach need to be distinguished:

- time-stepped advancement
- event-based advancement.

In the time-stepped approach, the systems could advance with equidistant time steps. In this case Δt strives for zero and time in both systems can advance virtually uniformly (See Figure 2).

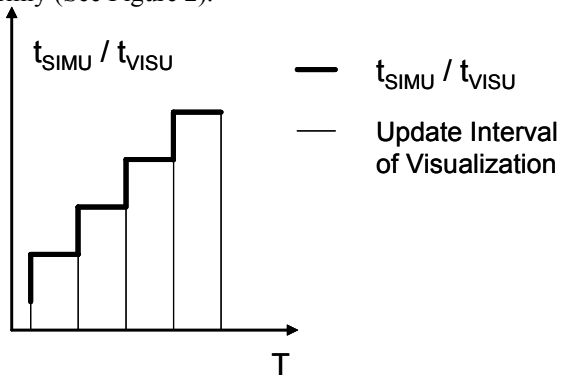

Figure 2: Time Stepped Synchronization of Logical Simulation Time

In the event based-approach, requested time advancements would correspond to actual event-time stamps. Advancement of logical time in both systems would not be uniformly in relation to wall clock time, as the simulation would typically lag behind (Figure 3). This for itself does not necessarily constitute a problem.

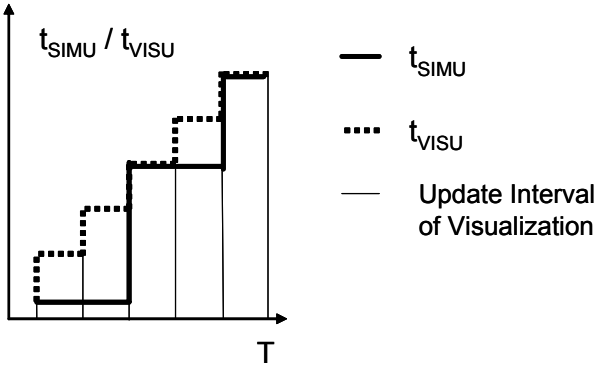

Figure 3: Event Based Synchronization of Logical Simulation Time

The advantage in both variants is that events are processed at the correct time stamp in both systems. This includes interactions in the VR system, which arrive at the correct time in the simulation and can be incorporated without violating the causality in the simulation.

The major problem in this general approach is that the VR system cannot wait for its own time advancement. If the simulation system needs extensive processing time at a certain event time stamp, time advancement of the VR needs to stop during that period. This is illustrated in Figure 4. Such waiting is not acceptable for the user in a VR environment. For these reasons, this approach in its present form is also only conditionally suited for coupling simulation and VR.

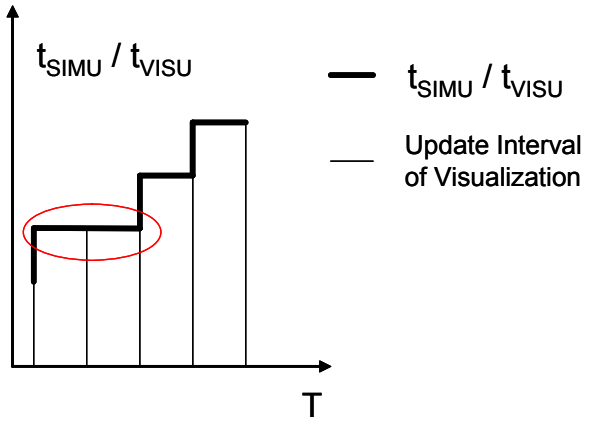

Figure 4 : Synchronization conflict if visualization has to wait for simulation

4.3 Synchronization Using Buffering

In this approach the simulation sends time-bound visualization commands to the visualization. These commands are stored in a buffer and the visualization reads these commands out of the buffer (Figure 5).

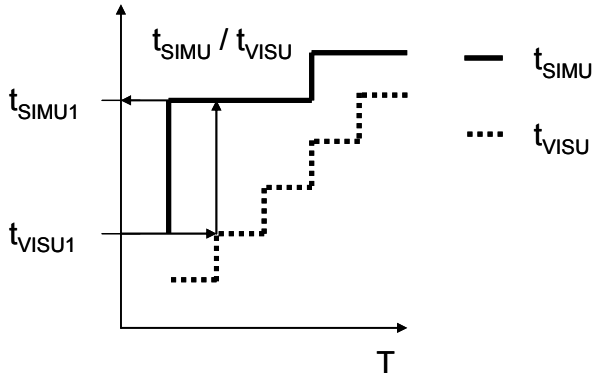

Figure 5: Time Advance with Classical Buffer Synchronization

The write calls in the buffer are synchronous calls. When a call cannot be executed because the buffer is full, then the simulation "waits" until the current contents of the buffer are less than the maximum buffer capacity. The simulation time is always larger than the visualization time, i.e. Δt(T) is always greater than zero. This method is applied for example in coupling with the visualization tool Concurrent Proof (Proof Animation 2004).

One advantage of this form is that, when buffer size is adequate, the visualization does not have to "wait" for the simulation and fluid image generation in the visualization

is provided as a result. The simple implementation of these buffer mechanisms is also an advantage.

Disadvantages of these variants are however the conflicts in the simulation, which can result from the visualization interactions because the visualization time is shorter than the simulation time. The value of the time difference $\Delta t(T)$ is not constant since it results from the difference between the largest and the smallest timestamp of the buffered visualization commands. The buffer capacity only determines the number of commands and has no direct influence on the time difference.

In principle however the method for buffering the visualization commands is suitable for coupling simulation with VR. The conflict for the interactions from the VR system is not resolved. However, intelligent management of the buffer size leads to an acceptable resolution of the conflict. Such management is described in the following.

5 METHOD OF SELF-ADAPTING BUFFER SIZES

Figure 6 shows the working method of the traditional buffer strategy. The visualization tool Concurrent Proof Animation (Wolverine 2004) puts this strategy into action. In this case, the buffer size is limited to a fixed number (e.g. 12) of visualization commands. Every time the simulator, for example SLX, intends to send a visualization command to the visualization tool Concurrent Proof, the current buffer occupancy is checked. If the buffer is in the state of its maximum capacity utilization, then the writing of the command in the buffer is disabled until current buffer occupancy is smaller than the maximum occupancy.

This strategy is well suited for unidirectional coupling of simulation and visualization. Concurrent Proof does not allow interactions with the visualization, which then affect the simulation model. A fundamental feature of the classical buffer strategy is that the buffer contains animation commands for an unknown and undetermined time interval. The buffer is not limited by the time but rather only by the number of commands. Consequently, the buffered commands can span a time interval from the millisecond range up to days.

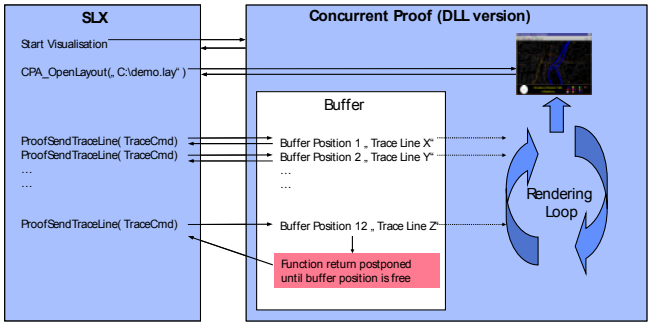

Figure 6: Concurrent Proof Buffer Algorithm

This classical approach cannot be used for coupling simulation with VR systems. When coupling with VR systems, the interactions from the VR world must be incorporated in the simulator with adequate precision. Thus a buffer strategy is required, which on the one hand contains visualization commands within a small time interval, yet on the other hand also contains a sufficient number of visualization commands to render visualization fluid and continuous.

The strategy of a self-adaptive buffer size grew out of these considerations. Core points are a configurable buffer size in time units and the maximum time difference allowed between two visualization commands. The time unit corresponds to the time unit in the simulation or in the VR. When coupling starts, an initial buffer size is defined. The basic principle behind this strategy is shown in Figure 7. The initial buffer size is specified at 5 time units and the maximum time difference at 3 time units. The time unit in both systems is commensurate with seconds.

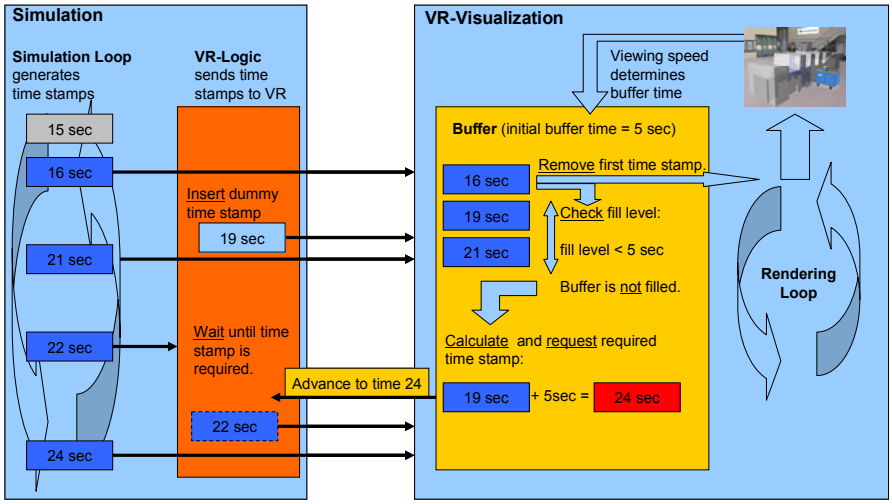

Figure 7: Adaptive Buffer Algorithm

The buffer size is adjustable as a function of the latency times in the network for data transmission and the capacity of the visualization tool.

The maximum time difference affects the generation of visualization commands. If the time difference between two real visualization commands exceeds this maximum time difference, a "dummy" command is inserted into the flow of commands. A dummy command is a simple command to advance visualization time (e.g.,"Time 19") without an animation event at this time stamp.

By inserting such commands, a constant flow of visualization commands is guaranteed, even if the event times in the simulation are widely separated from each other.

The current simulation time advance is dominated by the visualization as shown in Figure 8. The visualization pursues a strategy of the buffer always being filled with the next visualization commands for the buffer size set. This means that, when buffer size in time units is constant, a different number of commands can be found in the buffer. When the visualization deletes a command from the buffer, the remaining time interval in the buffer is calculated. If a value smaller than the buffer size results, a request command is sent to the simulation (e.g. "Advance to time …"). This command includes the necessary time advance.

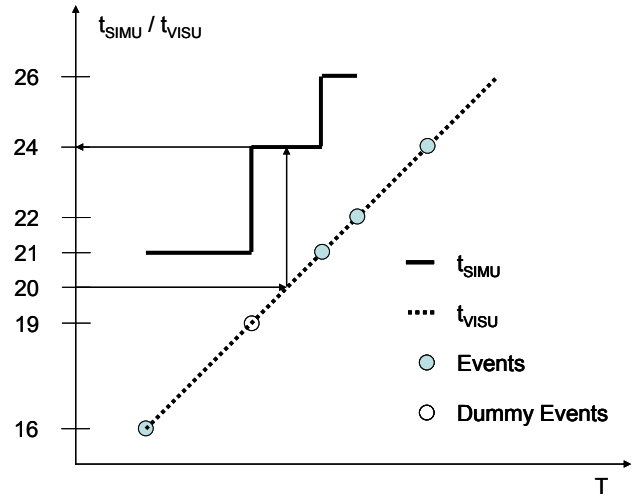

Figure 8 : Time Advance with Adapting Buffer Strategy

Assuming the buffer values from above one example shall be used to illustrate the behavior for interacting with the simulation (compare Figure 8): If an interaction affecting the simulation is triggered at time $t_{VISU}(T)=20$ in the VR system, then the current time in the simulation can already have reached $t_{SIMU}(T)=24$. The interaction is incorporated in the simulation virtually without loss of real time T (neglecting any network latencies). However the time difference is 4 time units. If a common time unit of seconds were specified, then the simulation would react with a delay of 4 seconds. A delay in this magnitude is still acceptable for the applications envisioned. Smaller buffer sizes, down to 1 seconds, are feasible.

The adaptive buffer strategy controls the buffer size as a function of the visualization speed. If the visualization speed increases, the buffer size also increases autonomously. If, for example, assuming a buffer size of 5 seconds, the visualization speed doubled, then a buffer size of 10 seconds would be set. On the other hand, if the visualization speed drops, the buffer size is reduced. If viewing is slow motion with half real-time, the buffer size is adjusted to 2.5 seconds.

That is why this approach is also called the method of self-adapting buffer size. The buffer size autonomously adjusts to changed conditions caused by network delays and visualization speeds. As a result, the delay between simulation and VR is confined to a justifiable minimum while preserving a high level of interactivity in the VR world.

6 PILOT PROJECT

To validate the correctness and functionality of the proposed solution a pilot project has been conducted. In this pilot project the simulation system SLX has been coupled with two different VR systems, one at a time.

- SLX is a commercial discrete event simulation system (Henriksen 1997).
- The interactive visualization system "Virtual Development and Training Platform (VDT)" is a flexible visualization system which can be applied in various VR-environments and settings (Blümel et al. 2004).
- VR Juggler is an open-source VR development environment initiated at the Iowa State University.

For implementing the temporally parallel coupling, all three systems have been extended using a modular approach.

A *communication unit* is responsible for the sending and receiving of messages and commands exchanged between both systems.

A *management unit* is responsible for integrating the received messages into the respective system and for the controlled release of outgoing messages to the other system. Figure 9 shows the basic structure of this pilot project.

The data exchange between the systems is based on a specific format. The chosen network transport protocol is TCP/IP. For reasons of efficiency the data exchange for the visualization only transmits state changes of the visualised objects. A prerequisite for this is that no messages between participating systems are lost and that messages are received by at the targeted receiver in the correct ordering sequence. The TCP/IP protocol guarantees these properties.

For coupling the systems a client-server-architecture has been used. The simulation component acts as the server and transmits by request of the VR component the relevant visualization data. The server functionality has been implemented inside the communication unit of the simulation component.

The buffer algorithm described in the previous section is implemented in the VR component.

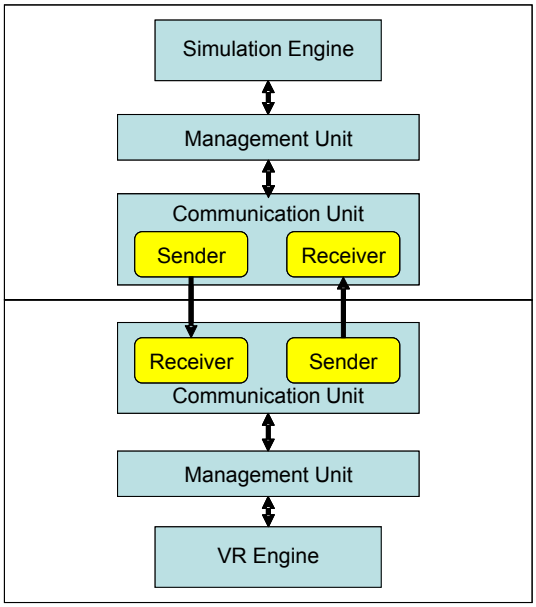

Figure 9: Structure of Components

7 SUMMARY AND OUTLOOK

This article has discussed different possibilities for coupling of commercial discrete simulation systems with virtual reality (VR) systems. The main focus was put on synchronization techniques as they are the key factor to enable a fluent interactive visualization in the virtual world. The key solution suggested in this article is an adaptive buffering strategy which can ensure fluent visualization while minimizing the potential of consistency conflicts between VR and simulation.

The approach described in this article was successfully tested with multiple pilot scenarios. Applications chosen come from the field of production, manufacturing and logistics. In one scenario, an SLX-based simulation model on a PC was coupled with a VR world in a cave. The simulation model has the capability of reacting to selected user interactions in the cave. Potential consistency problems between the two worlds were limited by selecting an initial buffer size of 5 seconds. The systematic errors this caused when the user interacted with the simulation had negligible impacts on the simulation results. Depending on the application scenario, also smaller buffer sizes can be configured.

The simulation's adapting buffer algorithm was encapsulated so that this method is not tied to the model developed. The necessary cave parts were implemented in Java so that porting into other VR systems is also possible.

Versions of the buffering algorithm have been implemented in JAVA and C++ and can be ported to other VR systems. They already have been successfully tested in two different VR systems.

Upcoming research work in this project will focus on porting this method to utilize it in other commercial simulation tools and on testing this buffer strategy in other VR systems. Another focus will be expanding the range of permissible interactions from the VR world, which then have to be incorporated in the simulator.

Future research could also involve using SLX state-saving and rollback capabilities to resolve the consistency conflicts between VR and simulation. The same applies for algorithms used in distributed real-time computer games.

The method for self-adapting buffer size presented proved to be a pragmatic and robust approach to interactive temporally-parallel coupling of simulation and VR and can form the basis for coupling commercial simulation tools with VR in practice.

REFERENCES

ARENA. 2005. Rockwell Software Inc. Available via <http://www.arenasimulation.com>. [accessed March 2005]

Barfield, W., Furness, T.A. III, 1995. *Virtual Environments and Advanced Interface Design*, Oxford University Press, New York, NY.

Bergbauer, J. 2002. Entwicklung eines Systems zur interaktiven Simulation von Produktionssystemen in einer virtuellen Umgebung. Doctoral Dissertation, University Clausthal, Germany.

Blümel, E., S. Straßburger, R. Sturek, I. Kimura. 2004. Pragmatic approach to apply virtual realty technology in accelerating a product life cycle. In: *Proceedings of the International Conference INNOVATIONS 2004*, 199-207. Slany, Czech Republic.

Holten-Lund, H. 2001. Design for scalability in 3D computer graphics architectures. Doctoral Dissertation, Technical University of Denmark. Available via <http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=888>. [accessed July 2005]

Kibira, D. and C. McLean. 2002. Virtual reality simulation of mechanical assembly production line. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charness, 1130-1137. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Dorozhkin, D. V., M. Lemessi, G. D. Rehn, and J. M. Vance. 2004. Integrating operations simulation results

with an immersive virtual reality environment. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 1713-1719. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

eM-Plant. 2005. Technomatix Technologies. Available via <http://www.emplant.de/simulation.html>. [accessed March 2005]

Franke, R. 2004. Kopplung von diskreter Simulation und interaktiver 3D-Visualisierung. (German) Master Thesis, School of Computer Science, University of Magdeburg, Magdeburg, Germany.

Jessen, U. 2001. Konzeptioneller Ansatz für die bidirektionale Kopplung von ereignisorientierten Simulationswerkzeugen und Virtual Reality Systemen. In *Proceedings of the 15th Symposium ASIM 2001*, ed. K. Panreck and F. Dörrscheidt, 121-126. Ghent, Belgium: SCS-Europe BVBA.

Henriksen, J.O. 1997. An introduction to SLX™. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Andradóttir, S., K. Healy, D. Withers, and B. Nelson, 559-566. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Proof Animation. 2005. Wolverine Software. Available via <http://www.wolverinesoftware.com/>. [accessed March 2005]

Ritter, K.-C., U. Klein, S. Straßburger, and M. Diessner. 1998. Web-basierte Animation verteilter Simulationen auf Basis der High Level Architecture (HLA). In *Proceedings of the 1998 Conference for Simulation and Visualization,* ed. B. Preim and P. Lorenz, 41-52. Ghent, Belgium: SCS-Europe BVBA.

Stuart, R., 2001. *The Design of Virtual Environments*, Barricade Books, Ft. Lee, NJ.

VDI. 2003. Richtlinie 3633:Part 11 203. *Simulation und Visualisierung*.

Whitman, L., V. Madhavan, D. Malzahn, and J. Twomey. 2002. Virtual reality model to aid case learning. In *Proceedings of the Industrial Engineering Research Conference*, May 2002.

AUTHOR BIOGRAPHIES

STEFFEN STRASSBURGER is head of the department "Virtual Development" at the Fraunhofer Institute for Factory Operation and Automation in Magdeburg, Germany. Previous professional experience includes two years as researcher at the DaimlerChrysler Research Center in Ulm, Germany, where he was involved with research topics in the Digital Factory and Digital Engineering context, esp. in the area of simulation integration and distributed simulation. He holds a PhD and a Master's degree in Computer Science from the Otto-von-Guericke University in Magdeburg, Germany. His international experience includes a one-year stay at the University of Wisconsin, Stevens Point and a stay at the Georgia Institute of Technology, Atlanta. He actively participates in several international conferences. His main research interests lie in distributed and web-based simulation, virtual reality and engineering, and middleware technologies like the High Level Architecture, CORBA, and Web Services. His email address is <strassburger@iff.fraunhofer.de>.

THOMAS SCHULZE is an Associate Professor in the School of Computer Science at the Otto-von-Guericke-University, Magdeburg, Germany. He received the Ph.D. degree in civil engineering in 1979 and his habil. degree for computer science in 1991 from the University of Magdeburg. His research interests include modeling methodology, public systems modeling, manufacturing simulation, distributed simulation with HLA and online simulation. He is an active member in the ASIM, the German organization of simulation. His email and web address are <tom@iti.cs.uni-magdeburg.de> and <http://www-wi.cs.uni-magdeburg.de>.

MARCO LEMESSI is a member of the Simulation Group in the Industrial Engineering Department of Deere & Company, the worldwide corporate headquarters of John Deere. He received his Ph.D. (2002) in Traffic Engineering and M.S. (1998) in Civil Engineering from the University of Rome "La Sapienza", Italy. Before joining Deere & Company in 2002, he has been expert consultant in several EU transportation research projects, and has lectured as expert in the Department of Environmental Engineering of the University of Perugia and in the Transportation Department of the University of Rome "La Sapienza". His email address is <LemessiMarco@JohnDeere.com>.

GORDON D. REHN supervises the Simulation Group in the Industrial Engineering Dept. of Deere & Company. He received his B.S.M.E. from Iowa State University, and is a registered Professional Engineer in the State of Illinois. He has performed discrete event simulation analysis of manufacturing operations since 1976 for internal Deere projects, as well as consulted on simulation projects outside the Deere organization. He has presented at various technical conferences for IIE, ASME, and INFORMS, as well as speaking at academic symposiums sponsored by the University of Iowa, and the Monterrey Institute of Technology in Monterrey, Mexico, and the Otto-von-Guericke-University, Magdeburg, Germany. He is a member of IIE. His email address is <RehnGordonD@JohnDeere.com>.