

another agent will cause agents to alter their execution. Agents are executed with a defined frequency of alarm. The alarm frequency dictates the speed to receive and send signals. For example, if the alarm frequency is set to slow (perhaps every second), the agent might miss signals from other agents because it only acts upon the signal it last reads after the second. An alarm frequency set too fast (perhaps one millisecond) will slow down the entire environment because it attempts to take exclusive control of the computer controlling the VE.

CONCLUSIONS AND FUTURE WORK

The development of an agent-based VE has been described, with emphasis on designing a virtual factory floor. The design of the virtual factory floor allows access to various aspects of the manufacturing system. These include the visual system description, agent behavior, and behavioral analysis techniques. In addition, the design of agents within the VE is explained. Significant characteristics of the VE will include: (i) system visualization, (ii) a framework for the integration of decision support capabilities, (iii) use of formal modeling tools and methodologies within the VE with active simulation, (iv) promotion of a modular agent-based approach to developing a prototype, and, (v) provisions for different visual perspectives of the virtual environment and a highly interactive interface.

Presently, the virtual factory consists of robot agents and setup agents. The communication between agents will undoubtedly change with the new implementation in the new version of DIVE (version 3.0). ISIS is no longer used in the newer version; a distributed computing package, SICS Multicast Distribution Package (SID), is used instead. Also, the new version will have a Tcl and World Wide Web (WWW) Interface. Therefore, the virtual factory will have the capability to support the interactions of multiple users with the facility of connection that the WWW provides (see Figure 9). Hopefully the development of the virtual manufacturing prototype will be an evolutionary one involving the input of many internet users.

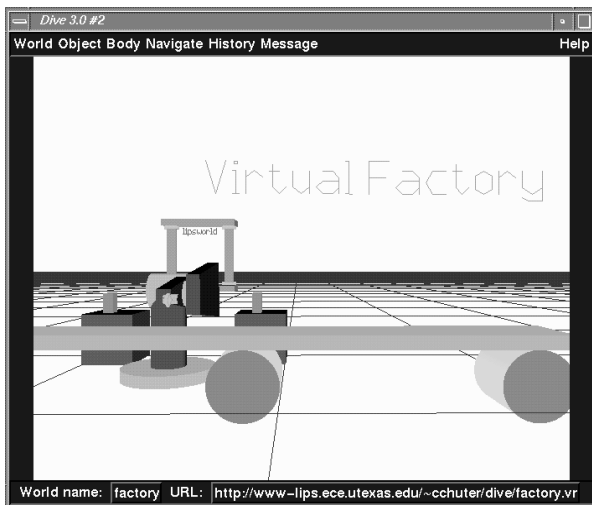


FIGURE 9. VIRTUAL FACTORY IN THE WWW

REFERENCES

- [Car93] C. Carlsson, O. Hagsand, "DIVE - A Multiuser Virtual Reality System", *Proc. of VRAIS*, Seattle, Sept. 1993.
- [And94] Andersson, C. Carlsson, O. Hagsand, O. Stahl, "DIVE - The Distributed Interactive Virtual Environment Technical Reference Manual", Swedish Institute of Computer Science, March 1994.
- [Bir94] K. P. Birman, T. Clark, "Performance of the ISIS Distributed Computing Toolkit", Cornell University TR 94-1432, June 1994.
- [Res94] S. Ressler, "Applying Virtual Environments to Manufacturing Technology", National Institute of Standards and Technology, NISTIR, June 1994.
- [VM93] ----, "Virtual Manufacturing: A Methodology for Manufacturing in a Computer" Airforce Mantech Perspective, Oct. 1993.
- [Ford95] ----, "User Profile: Ford Motor Company", *VR News*. London. Vol. 4, Issue 1. Jan/Feb 1995. pp. 26-29.
- [Bry92] S. Bryson, C. Levit, "The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows", RNR-92-013, NASA Ames Research Center, April 1992.
- [Lof94] R. B. Loftin, P. J. Kenney, "Virtual Environments in Training: NASA's Hubble Space Telescope Mission", *Submitted to 16th Interservice / Industry Training Systems and Education Conference*, <http://www.jsc.nasa.gov/cssb/vr/Hubble/shortpaper>, Nov. 1994.
- [Coh93] S. S. Cohen, R. C. Sharable, The Object-Oriented Brewery: A Comparison of Two-Object Oriented Development Methods", *ACM SIGSOFT Notes*, Vol 18, No. 2, April 1993, pp. 60-73.
- [Wie90] L. Wiener, R. Wirfs-Brok, B. Wilkerson, "Designing Object-Oriented Software", Prentice-Hall, 1990.
R. S. Pressman, "Software Engineering: A Practitioner's Approach", Third Edition, Mc-Graw Hill, 1992.
- [Har87] D. Harel, "StateCharts: A Visual Formalism for Complex Systems", *Sci. Computer Programming*, Vol. 8, No. 2, June 1987. pp. 231-274.
- [Mur89] A. Murata, "Petri Nets: Properties, Analysis and Applications", *Proceedings of the IEEE*, Vol. 77, No. 4, April 1989.
- [Bar94] B. T. Barcio, "State Machines for Object-Oriented, Concurrent, Hierarchical Engineering Specifications", M. S. Thesis, University of Texas at Austin, Dec. 1994.
- [Ram96] S. Ramaswamy, K. P. Valavanis, "Hierarchical Time-Extended Petri Nets Based Error Identification and Recovery for Multilevel Systems", *to appear in the IEEE Transactions on Systems, Man and Cybernetics*, Feb. 1996.
- [Sta92] O. Ståhl, "MDraw - A Tool for Cooperative Work in the Multig Telepresence Environment", Swedish Institute of Computer Science, SICS T 92:05, May 1992.
- [Lee83] C. S. G. Lee, "Robot Arm Kinematics", In C. S. G. Lee. R. C. Gonzalez, K. S. Fu, Editors, *A Tutorial on Robotics*, IEEE Computer Society Press, 1983.
- [Spo89] M. W. Spong, M. Vidyasagar, "Robot Dynamics and Control", John Wiley & Sons, 1989.

simulations include:

1. Multiple users evaluate/control the robot: Users activate and view the robot within the same VE using the same tools.
2. Three-dimensional movement/control is facilitated: Users simply 'walk around' to view different perspectives of robot movement.
3. Immediate solutions/problems visually observed by a group of users: Users viewing backlog or idleness can deactivate robots to make simple scheduling changes.

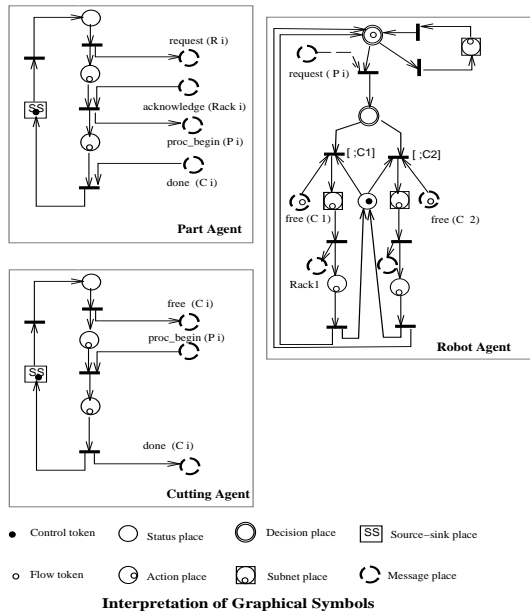


FIGURE 6. PETRI NET SYSTEM MODEL OF INDIVIDUAL SYSTEM AGENTS

Simulation. A surface modeled robot is created as shown in Figure 7. The robot arm link coordinate system as well as the derived equations giving the forward and inverse kinematics are taken from [Lee83][Spo89]. The base of the robot defines the starting coordinate axes, a right-handed coordinate system with the z-axis pointing upward through the center of joint 1. Each following joint rotates around a z-axis transformed from the base z-axis. For example, rotating around the base z-axis by 90 degrees and then rotating around the x axis -90 degrees, both according to the right hand rule, will result in a new z-axis controlling the rotation of the second joint.

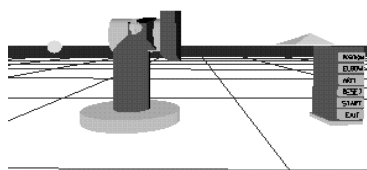


FIGURE 7. PUMA ROBOT WITHIN DIVE

This design greatly facilitates the programming of the robotic simulator. Each process of rotation within the simulator is enacted only on one joint. Other simultaneous movement of joints are dependent on the position of the original joint. Thus, rotation of the first three joints is hierarchical and solely around the z-axis of that joint object.

The robot is constructed with the base of the robot being defined as the top object and the joints being defined hierarchically as sub-objects. Each subsequent joint is defined as a sub-object of the object before it. The design is chosen to facilitate graphical consistency and movement. The data file description of the robot joints is illustrated in Figure 8. In addition, changes in the robot arm link coordinate parameters can be easily implemented in the data file by changing parameter variables. By changing appropriate values, users will be able to test future designs of robots or designs of robots not in their possession.

```

%converted to meters
#define A1 0.0
#define A2 2.4318
#define D1 0.0
#define D2 .14909
%converted to radians
#define THETA1 1.57
#define THETA2 0.0
#define ALPHA1 -1.57
#define ALPHA2 0.0
...
%base of robot
object {
  view 0 {
    cylinder .4 .4 1 % .4m radius, .1m height
  }
  subobject { %joint 1
    view 0 {
      cylinder .15 .15 .6604
      % .15m radius, .6604 height
    }
    subobject { %joint2
      translation v A1 0 D1
      eulerxyz v 0 0 THETA1
      eulerxyz v ALPHA1 0 0
      view 0 {
        cylinder .15 .15 .44909 %(D2 + 3)
        %D2 + diameter of joint1
      }
    }
    subobject { %joint 3
      translation v A2 0 D2
      eulerxyz v 0 0 THETA2
      eulerxyz v ALPHA2 0 0
      view 0 {
        cylinder .1 .1 .1
      }
    }
  }
}

```

FIGURE 8. DIVE DATA FILE FORMAT

Communication. Robot agents are introduced into the VE with the help of a setup controller agent as shown in Figure 3. Communication between the controller agent and the robot agent, although visually transparent to the user, is directly controlled by buttons on the console of the setup controller. For example, to test if communication lines are open between the controller and the robot, a user presses the test button. If the agent receives the test signal from the controller agent, it will send back an acknowledge signal to the controller agent. Each agent executes independently, but a signal from

tion about the source or destination for the particular event, that is, it is of the form:

$$E: 'type', 'source', 'destination'$$

i_s and f_s represent initial and final state specifications or petri net markings. G represents the goal state. All goal states must be specified in terms of f_s states, but f_s states can also be states that are not recognized goals, i.e. error states. As an example, a robot agent will be represented by the six-tuple, $A_{robot} = (R, E_{ri}, E_{ro}, i_{rs}, f_{rs}, G_r)$. The operational states of the robot are shown in Figure 4. The robot moves from the *Idle* state to the *Active* state by picking up an object and returns to an *Idle* state after successfully moving the object to its destination, represented as 'put down' in Figure 4. Any error will cause the robot agent to transition to its *Down* state. Agents can either receive (input) events or send (output) events while in the *Active* and *Idle* states. In the *Down* state an agent can only send (output) events, thereby announcing to other agents that it is not operational. Therefore, a *Down* state consists of only f_s states, whereas an *Idle* state can contain both i_s and f_s states.

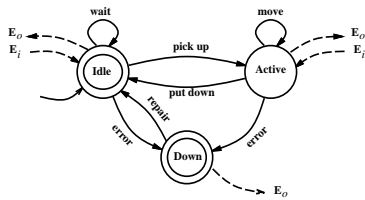


FIGURE 4. STATES OF ROBOT AGENT

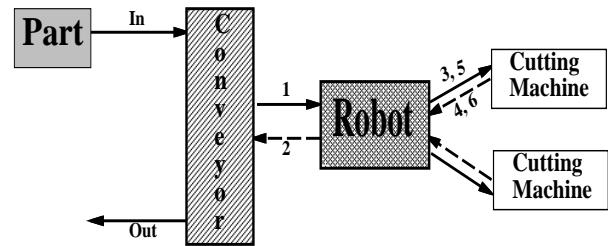
Agent Communication

Communication within the virtual manufacturing system is between agents. Communication among objects is in large part defined as internal to an agent. Of course, communication among objects is often reflected in communication protocols among agents. For example, if a moving joint of a robot reaches a singularity or joint limit in the *Active* state, this may be seen as an error message from the robot agent. Every agent can generate input events for other agents and each agent can become active upon receiving an input event.

As an example, consider a virtual factory floor consisting of a conveyor, one robot, two cutting machines, and a part to be manipulated by these machines, as shown in Figure 5. Parts arriving on the conveyor, are picked up by the robot and moved to one of the two cutting machines. The numbers in Figure 5 signify event pathways. Dashed arrows in Figure 5 refer to event acknowledgments. The Petri net model of the various agents in the system is shown in Figure 6.

As soon as a part arrives on the conveyor, the part agent sends an outgoing event, E_{po1} , (*request* in Figure 6) to the agent able to perform some operation on the input part (in this case, a pick-up event for a robot, represented as an incoming event, E_{ri1} , (*request* in Figure 6) to the robot agent). The robot remains in an idle state until the part is in its range. Then, the robot picks up the part and moves it to one of the cutting machines. The communication between the robot and the cutting agents determine whether one of the cutting machines is free to perform a cut on the incoming part. If a machine is free the part is moved to the appropriate cutting machine (*free* represents the available cutting machine in Figure 6). Otherwise, the

robot agent does not pick up the part. This will be indicated by the part reaching an error state of operation. If one machine is in its *Idle* state, it will send an appropriate event to the robot allowing the robot to move the part to that machine. When both machines are *Active*, the robot waits until one becomes *Idle*. Consequently, if the robot cannot reach the cutting machines or reaches a singularity from which it cannot return, it will transition to its *Down* state. (This is not represented in Figure 6). When the part receives an acknowledgement from the robot after being picked up successfully, it sends another event, E_{po2} , instructing the cutting agent to perform the cut operation. Finally, the cutting agent will receive the cut instruction, E_{ci1} , (*proc_begin*) from the part agent and perform the cutting task. The event pathways are summarized in Figure 5.



Event	Part	Robot	Cutting Machine
E_o	1 - Pick up	3- put down	4 - Ack. put down
	5- Cut	2 - Ack. pick up	6 - Ack. cut
E_i	2 - Ack. pick up	1 - Pick up	3 - Put down
	6 - Ack Cut	4 - Ack. put down	5 - Cut

Event Communication Table

FIGURE 5. VIRTUAL FACTORY FLOOR

IMPLEMENTATION

Users can enter a VE and interact in real time with each other and the VE, either by pointing devices, such as a mouse, or aural vehicles that relay voice messages. Gateways within the virtual world act as bridges between other worlds or computers. In addition to users, various application processes can be simultaneously executing within the VE. For example, one application allows multiple users to draw shapes and text on a white-board mdraw, while another application displays a clock. There can be multiple instantiations of each application. Also, each application can be located anywhere in the VE.

Using standard internet protocols and the ISIS distributed computing package, DIVE allows the creation of a dynamic VE for both users and applications. The VE rests upon a threaded shared database that several computers can access. An interface application, called the visualizer, supports the use of several I/O devices such as head-mounted displays (HMD) and datagloves.

Agent Example: Robot. A robot agent acts as a three-dimensional robotics simulator within the VE. The robot agent employs forward and inverse kinematics to accurately simulate a robot. Multiple robots can be active simultaneously in the virtual factory floor setting. Multiple users, connecting through a virtual gateway, can either view or interactively control these robots.

Advantages in using VE's over standard two-dimensional robotic

isfy each system responsibility. This stage also establishes a sequence of such responsibilities and a high level communication structure between the system resources based on the constraints involved and the necessary schedules.

2. *Agent Design*: This stage involves the definition of distinct groupings within the system, the various states of the individual agent, and events that trigger the transition between these states. The interactions with other agents in the system and the means of such interaction is also defined.
3. *Evaluation*: In this stage, the event based state description of each agent is modeled and analyzed using state based modeling and simulation tools, such as state charts, petri nets, or timed finite state automata.
4. *Refinement*: In some instances, especially during initial development and after the evaluation of certain behaviors, there may exist a need to define new system requirements or identify additional responsibilities. In such cases the VE development follows the *requirements tuning* cycle. However, once the responsibilities have been completely established, additional constraints, new behaviors, or methods may be added to agent definitions. In this case, the VE development follows the *behavior tuning* cycle. The agent-based VE prototyping model allows for new agents to be defined even at the final stages of development so that objects and agents can be redefined to decrease coupling and increase cohesion.

Two kinds of agents are distinguished: *active* and *passive* agents. Active agents are agents that have some assigned responsibility and hence are always executing towards accomplishing their stated responsibilities while passive agents are those that become active only in response to some service request. In the VE all agents are considered to be active agents. Each active agent operates in two different modes: (i) *Autonomous*: In this mode, the agent operates by itself, executing some functionality that is derived from its overall responsibilities. The agent does not interact with other agents, and, (ii) *Service*: In this mode, an agent's execution is dictated by requests from other agents that require some service. The agent's communication module is active in this mode.

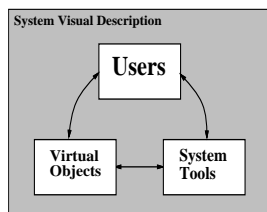


FIGURE 2. SYSTEM SPECIFICATION

Agent responsibilities are partitioned into distinct stages of execution. Each of these stages are executed in an autonomous mode. However, in between these stages, agents check for service requests and depending on the relative priority of the requests, an agent chooses to intertwine the service within its thread of execution. The communication module of an agent is always enabled to queue service requests.

The system development environment incorporates object behavior, visual descriptions, simulation, analysis, and user inputs, as il-

lustrated in Figure 2. Users have access to all aspects (objects and agents) of the manufacturing system. Single or collective manipulation and observation by single or multiple users affect the visual system description. In addition, users can affect each agent individually by generating certain events. Simulation and behavioral analysis tools such as state charts [Har87][Bar94] or petri nets [Mur89] [Ram96] will also be accessible to the VE users/designers.

Figure 3 shows a detailed representation of the various agents populating the virtual manufacturing system. Users and system tools, such as scheduling systems, process planners, and behavioral modeling tools, are agents that communicate with other agents defined within the VE. These tools aid in defining and analyzing specific system characteristics. Once users enter a VE they are considered to be higher level agents that can control the behavioral characteristics of other agents in the VE. Users can view the behavioral specifications of individual agents and can directly manipulate these agents. *Note that agent definitions may themselves reflect a nested hierarchical structure.*

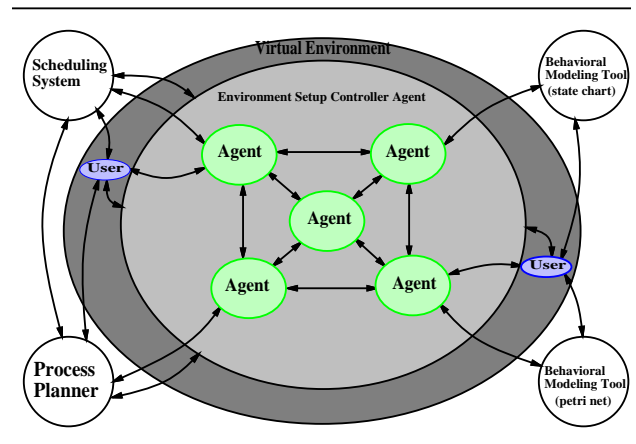


FIGURE 3. AGENT-BASED SYSTEM DESCRIPTION

The initial configuration of virtual objects within the virtual factory is controlled by an environment setup controller agent. Users as well as other system tools can communicate with this agent.

Agent Description

Objects that comprise an agent description significantly collaborate with each other. Specifically, each object is associated with an agent specification, and in each mode the agent has three macro-states of operation: *Idle*, *Active*, and *Down*. The first two states describe operational states; in the *Idle* state the object conducts a self-diagnosis and in the *Active* state it attempts to satisfy a goal. These states are contained in either mode of operation. In the *Down* state the object is disabled from operating in the virtual factory.

An agent is described by a six-tuple:

$$A_{behavior} = (D, E_i, E_o, i_{sys}, G)$$

where, D represents the behavior specification, either by state charts or petri nets; specifically, hierarchical extensions to these basic system behavior models are used [Bar94][Ram96]. E represents a list of known events, which are either input events, E_i , or output events, E_o . Furthermore, every event variable also contains informa-

provide astronauts with step by step instructions for repairing the hubble space telescope [Lof94].

Significant, successful results have been obtained by organizations using VE to aid manufacturing processes. Ross Operating Valves have incorporated VM techniques to reduce delivery time from concept to product time. The delivery time was reduced from six months to 24 hours. This reduced engineering and direct labor cost by 80%, and inventory by 35%. Pitney Bowes has utilized aspects of VM to reconfigure its factory while meeting the production schedule. This resulted in decreasing parts inventory by 60%, while production increased by 50% per factory floor space [VM93]. Other organizations have also produced promising results using VE in manufacturing areas. Caterpillar Inc., reported both cost and time benefits by using VE's to design and evaluate wheel loader and back-hoe loader designs [Res94].

The objectives of this paper are to: (i) provide a complete model for VE development, (ii) enumerate, in detail, the various aspects of the model and the various stages involved in the development of the VE, and, (iii) implement a simple case study to illustrate this approach. The proposed virtual factory will act as a testbed to visualize and test technology for manufacturing applications. For example, the virtual factory will allow the testing and integration of different assembly sequence or knowledge-based planners, scheduling algorithms, and behavioral modeling techniques. The virtual factory acts as an implementation of a manufacturing prototype.

The virtual factory is designed to be modular. Before proceeding further, it should be noted that terms such as objects and agents have been used interchangeably in this paper to identify a compact representation of the system resources and its critical operations. In general, agents refer to a grouping of a set of resources and their common operations, akin to a class definition in object-oriented programming languages, and, objects refer to class instantiations with the definition of specific methods/functions. Objects in the VE act independent of one another and communicate by pre-established protocols. Objects are further associated with agents depending on their behavior specifications. This design approach allows for the combinations of devices to create different virtual factories and facilitates easy changes in the factory floor system. For example, the internal workings of a robot agent controlling its kinematics can be adjusted for different types of robots. Individual joints or links of the robot could be derived as separate objects. These changes do not change the overall behavior of the agent and can be easily re-integrated into the virtual factory.

MANUFACTURING PROTOTYPE MODEL

A *virtual manufacturing prototyping environment* is an environment designed for the construction of an agent-based virtual manufacturing environment integrated with modeling and analytical tools that facilitate the specification and analysis of design and management decisions. The effects of such decisions can be studied directly on the manufacturing prototype. This approach not only gives an opportunity to feel and examine design decisions on a virtual factory floor, but can also serve as a communication medium between designers at various sites.

The VE development is carried out to assist in the design or redesign of the actual manufacturing floor. Information required for the construction of the VE include the following: (i) domain specific in-

formation, such as the purpose of the VE, important activities to be performed, resources involved in performing these activities, and the constraints involved, (ii) tools and resources required to construct the VE, and, (iii) tools and resources required for interaction with and manipulation within the VE.

The above factors dictate the VE system architecture. The architecture supports the identification of the resources involved, the responsibilities of individual resources and methods or procedures involved in establishing subtasks to accomplish some overall responsibility. These factors influence the definition of agents and agent behaviors. Figure 1 illustrates the various stages involved in the construction of the VE.

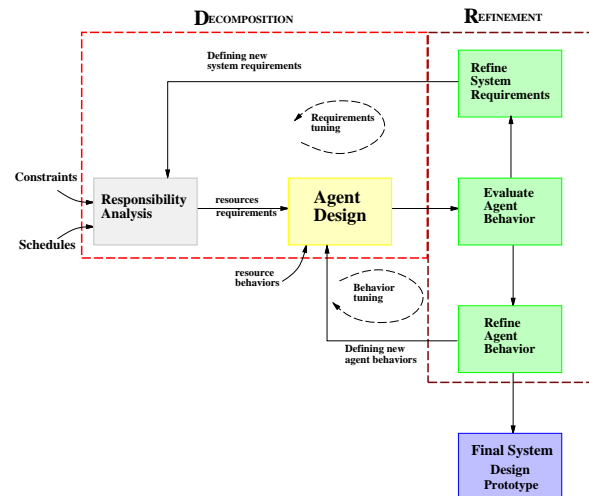


FIGURE 1. VIRTUAL ENVIRONMENT PROTOTYPE MODEL

A complete VE definition depends on accurate information related to resources, constraints, and schedules in the real world domain and an effective representation. Resources consist of hardware and software systems as well as personnel involved in the creation and manipulation of the system. From the initial system description, agents are defined and developed using a responsibility-based requirements analysis [Coh93]. The design of agents along with their communication and behavioral models, defines the final VM prototype. A complete VM prototype defines a testbed for the analysis of the system.

Agent-based VE definition is an iterative and interactive process. Successive iterations lead to a well designed system structure and refinement of agent behavioral requirements. The first few iterations lead to a well defined external system view while successive iterations lead to a coherent internal view. An external view is one that describes the agent in terms of its external activities while the internal view provides details of the agent's complex internal functionalities [Coh93]. The VE development cycle follows four distinct stages. These four stages, as illustrated in Figure 1, are:

1. *Responsibility Analysis*: This stage identifies the major tasks in the development of the virtual system and the overall system responsibilities. The output of this stage identifies various sub-responsibilities with certain system resources, necessary to sat-

A VIRTUAL ENVIRONMENT FOR CONSTRUCTION AND ANALYSIS OF MANUFACTURING PROTOTYPES*

Christopher J. Chuter, S. Ramaswamy, and K. S. Barber

Laboratory for Intelligent Processes and Systems
Department of Electrical and Computer Engineering
The University of Texas at Austin - Austin, Texas, 78712-1084

ABSTRACT

The building of a virtual environment for the construction, analysis, and design of a three dimensional virtual manufacturing prototype is discussed. An agent-based framework is used to specify the visual environment. The environment is built to be flexible enough to support research efforts in scheduling, planning, and behavior modeling. The implementation consists of a heterogeneous, distributed interactive virtual environment using the virtual environment tool, DIVE.

INTRODUCTION

Price and availability are some of the main drawbacks in restricting the widespread use of virtual environments (VE). Therefore, VE's have been used more in research than in practical applications. While virtual reality (VR) system hardware (a head-mounted display and dataglove with trackers, etc.) is costly, numerous VR software tools have been developed and are available free of cost, or, at a nominal licensing fee. Certain tools are essential to adequately create an immersive VE. An ideal multi-user virtual environment system consists of multiple workstations and a variety of application specific virtual reality devices. Interactive modes of operation are most benefited by the use of position sensitive glove devices. A position sensitive head-mounted display (HMD) can be used at all times to facilitate the observation of all manufacturing activities. Haptic and spatial audio devices aid in analysis and allow cues for evaluation and observation. In this research, we use the freely available, academically licensed software, DIVE (Distribut-

ed Interactive Virtual Environment), developed by the Swedish Institute of Computer Science [Car93][And94]. With the aid of the ISIS package [Bir94], the DIVE system provides a non-homogeneous, distributive computing environment.

VE's offer manufacturers the ability to evaluate manufacturing capabilities, schedules, process plans, production processes, and new technologies that affect manufacturing, without committing a vast amount of technical and financial resources. Several companies and organizations have investigated using VE's in manufacturing domains [Res94][VM93]. Virtual manufacturing (VM) will allow organizations to conduct rapid and thorough assessments of risks, affordability, producibility, and other impacts on manufacturing capabilities. In addition, VM simulation will provide a foundation for continual experimentation and process development, thereby aiding in product advancement.

The U.S. Air Force VM initiative attempts to develop technology to investigate the development of advanced VE manufacturing systems before actual deployment [VM93]. Some organizations have incorporated virtual technology for more specific aspects of manufacturing. Boeing and Ford Motor Company have both used virtual reality (termed Augmented Reality in Boeing's research) within the product design phase. Specifically, Boeing uses a HUDset (Heads-Up, see-through, head-mounted Display) to superimpose computer generated images over physical objects [Res94]. The Ford Motor Company's *Global Studio* is a networked computer environment which will enable the company's designers and engineers throughout the world to work together on three-dimensional models of new vehicles [Ford95].

NASA has pioneered virtual reality (VR) work within many fields, such as computer design and education. One such work is the virtual wind tunnel, a VE allowing a user to explore three-dimensional unsteady flows. Using a boom-mounted six degree of freedom head-position-sensitive stereo CRT system and a position sensitive glove, one can interactively view air flows moving around an object [Bry92]. More recently, NASA has developed a VE to

* This Research was supported in part by the Texas Higher Education Coordinating Board, Advanced Technology Program, under grants ATP-115 and ATPD-112.