Proceedings of the 2011 Winter Simulation Conference
S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, eds.

USING HYBRID PROCESS SIMULATION TO EVALUATE MANUFACTURING SYSTEM COMPONENT CHOICES: INTEGRATING A VIRTUAL ROBOT WITH THE PHYSICAL SYSTEM

J. Viswanathan
W. S. Harrison
D. M. Tilbury

Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109-2125

Fangming Gu

Manufacturing Systems Research Lab
GM Global Research and Development
Warren, MI 48090

ABSTRACT

When using models and simulations in the design and reconfiguration of manufacturing systems, it is difficult to gage the fidelity of the model, especially if the system being modeled doesn't yet exist. The model cannot typically be validated until the system is in place. We propose the concept of Hybrid Process Simulation (HPS), an extension of traditional Hardware-in-Loop (HIL) technology, as a bridge between pure simulation and the final physical system. In this paper, we present a framework for swapping a virtual device with its real counterpart. The virtual model is developed using simulation software obtained from the vendor of the actual device, so that the model offers the same functionality as the actual device. Using the vendor simulation software, we were able to demonstrate the modularity of HPS, since the virtual and actual device can be more easily swapped with minimal changes to the rest of the existing system. The framework is presented as a proof-of-concept for HPS applications in the design and reconfiguration of a manufacturing line. A case study describing the implementation of the hybrid process simulation using a virtual and real robot on the Reconfigurable Factory Testbed at the University of Michigan is presented.

1 INTRODUCTION

Due to the increasing complexity of industrial processes, cyber-physical systems (CPS) have the potential to play a major role in the design and development of manufacturing systems, offering new capabilities that transcend current technology (Baheti and Gill 2011). There are however considerable challenges in the development of a cyber-physical environment with needs for "standardized abstractions" that allow the design of a CPS in a modular fashion, virtual models that can communicate with complex physical systems, and virtual and physical components that are reliable and reconfigurable (Baheti and Gill 2011). One way to address these challenges is through modeling and simulation.

In today's manufacturing environment, simulations analysis is used for a wide variety of applications such as throughput analysis and system validation. The accuracy of the simulation depends on the accuracy of the model itself. This raises various fidelity issues. How can a manufacturing system be simulated such that we can consider the output of the model to be a faithful representation of the real system? What if the real system does not yet exist? Modeling the components, layout and connectivity is difficult and expensive to carry out, with multiple simulations being required in many cases. In addition, it is hard to address issues such as simulations of data communication networks in manufacturing systems with pure virtual models of the system.

The development of manufacturing systems, thus, is usually carried out manually by field-testing, and the correctness of the design can only be validated in the implementation phase. This can present high rates of misunderstandings in the requirements and/ or mistakes in design, resulting in longer development time

and lower efficiency. In addition, any ideas for reconfiguration of the assembly line will have to follow a similar process of development with field-testing which becomes expensive and/ or time-consuming.

We present the idea of a Hybrid Process Simulation (HPS), a modular approach that uses Hardware-In-Loop (HIL) and Emulation-In-Loop (EIL) technology and blurs the line between simulation and field testing, such that neither is completely isolated from the other (Harrison and Tilbury 2008). Some existing HIL simulations use high-level data communication (e.g., OPC), instead of a system's native data communication method (e.g., DeviceNet, Profibus). We propose the HPS system consisting of virtual models and physical components, fused in a manner where it is difficult for a component to realize if it interacts with a physical or virtual component at a low-level. Thus HPS provides the benefits of a simulation with respect to easier system modification and scalability, and the benefits of a physical system. For instance, the data communication network does not need to be wholly simulated and can exist in its actual form. HPS is also modular which makes it possible to switch out individual virtual or physical components without affecting the rest of the system. This means that a HPS system is easier to reuse and can effectively shorten the development life cycle.

2 PROBLEM STATEMENT

A key aspect of the HPS approach is the communication between the virtual model and physical components of the system. Not only do the physical and virtual components need to be able to communicate with each other in real-time, but a framework that can support a smooth and easy transition between pieces of virtual environment and physical devices needs to be developed. In addition, as with most simulations, it is necessary to characterize the accuracy of the simulation. How close is the virtual model of a component to the actual device? Does the virtual model offer the same range of functionality as the actual device? We present a novel approach for this. By using simulation software provided by the vendor of the device, we can obtain the same or almost the same range of functionality on both the virtual component and the physical device. In some cases, it is also possible to directly download the control logic from the virtual to the actual device once it has been implemented, eliminating the time for re-writing the logic. The challenge presented in this approach is that vendor simulation software is not open source and it is difficult to communicate between the software and external devices.

In this paper, we present the methodology for integrating a single virtual robot in a hybrid process simulation using modeling software obtained from the vendor of the physical robot. A brief literature review summarizing the state-of-art for HPS is provided in Section 3, wherein a novel application of the proposed HPS methodology in a generic assembly plant is also discussed. Details on the integration of the virtual robot into an existing real system are discussed in Section 4. Section 5 provides a case study describing the implementation of the HPS in the Reconfigurable Factory Testbed at the University of Michigan, Ann Arbor (Moyne, Korsakas, and Tilbury 2004). Section 6 finally concludes the paper by describing current limitations and potential enhancements to the implemented HPS methodology.

3 BACKGROUND

3.1 State-of-the-Art

Several works have focused on building virtual models to simulate part or all of a manufacturing line. Pohit (2006) focuses on purely software to simulate a spur and helical gear design and manufacturing process, designed to give the user an idea of the actual gear generation process. Park (2005) develops a methodology for creating a virtual model for pure planning and predictive purposes without any real time communication or feedback from actual devices. The resemblance of the simulation to the actual system depends heavily on the accuracy of the simulation model in both cases. Ng et. al (2008) take this a step further by using I/O synchronization with a 3-D graphical simulation to model and study manufacturing resources (e.g., robots). It involves a manufacturing simulation of a machine service support system (MSSS) for remote press line monitoring and diagnostics using discrete event simulations (DES) and computer-aided

robotics (CAR). Other works such as (Wang, Orban, Cunningham, and Lang 2004) and (Chen, Bender, El-Wardany, and Renton 2002) focus on the emulation of a 3-axis CNC milling operation by integrating process simulation software and multimedia technology to realize a virtual environment for manufacturing process optimization and real time process monitoring.

Most of these systems that have been developed are just component simulations or models of actual systems, and cannot be considered as HPS systems, even though some involve some form of I/O synchronization with the simulated models. This is because HPS tries to address one of the grand challenges of modeling a complex manufacturing system: to obtain plug and play interoperability of independent component simulations (Fowler and Rose 2004). In many cases real-time I/O communication and connectivity in component simulations is not considered at a low-level (with respect to the actual system's native data communication network). However, with HPS in a manufacturing line, this can be done, such that virtual machines can be swapped out in a plug and play manner with their real counterpart, without making changes to the system as a whole.

We can think of the concept of HPS being closely related to model-driven software engineering, where the goal is to develop a framework for reuse of and interoperability between a variety of passive and active information models (Breton and Bezivin 2001). In model-driven software engineering approach, this is primarily done by starting off with a broad scope or an abstract decomposition of a system or a process, termed as meta-models, with focus on the organization of the meta-models and the relationship with other models (Breton and Bezivin 2001, Weston 1999). These generic meta models can then be used to develop more specific ones, thus making them more suitable for reuse. The design and development of meta-models are however based on the presumption that processes, products and services are pre-defined (Weston 1999).

Since one of the purposes of the HPS approach is to utilize the component simulations that are real-time and capable of communicating with a real system, there is a close overlap of the HPS approach with symbiotic simulations systems and online simulation systems. Although ambiguous in their definitions, both types of approaches usually refer to simulations that are initialized and driven by real-time sensor data from the physical system (Aydt, Turner, Cai, and Low 2008). The definition of HPS on the other hand does not necessitate this, as it possible to develop a hardwired simulation in which independent component simulations are driven by other component simulations over real-time networking and communication infrastructure. The accuracy or quality of simulation computations in the symbiotic or on-line approach is also heavily dependent on the model approximated to the real system (Aydt, Turner, Cai, and Low 2008, Peters, Smith, Curry, and LaJimodiere 1996). Less accurate models thus can have an adverse effect on the physical system. Though this may be true for HPS, since component models can be provided by the vendor of the actual device, the model is as close to the actual device as offered. Also, it is possible to minimize any unwanted, adverse effect on a physical system by using a hard-wired simulation with the model in question and the rest of the independent component models. Additionally, since the models are provided by the component vendor, we can assume that the model has the same characteristics and kinematics as the real device, which minimizes any consistency issues that could arise during replacement of the component models with the actual devices.

The difficulties associated with creating such a hybrid system are discussed in (Diogo, Vicari, Loures, and Busetti 2008), which presents a methodology for developing an environment in which pieces of the physical system can be incrementally added to a simulated model for testing the logic control. The limitation faced here was the integration of proprietary systems and industrial protocols that come with each device or manufacturer. Consequently, in a technologically heterogeneous system, establishment of the communication infrastructure between the simulated models and the actual devices with the industry standards is different. In this paper, we address this problem through the development of an interface program that allows communication with the simulated models using industry standards and protocols, such that communication with a real device and its virtual counterpart are the same.

3.2 Previous Work on HPS

Customarily in a manufacturing system, process simulations are performed first, validated and physical implementation is started. If any problems arose during the physical implementation of the system, one possible approach is to revisit and validate the process simulation and retry the physical implementation. The Hardware-in-Loop (HIL) approach was developed to address this type of an iterative process by allowing the creation of a test setup using simulations that are connected to real hardware/ software. There are however few drawbacks associated with this approach, including the lack of a widely accepted formalized methodology and typical applications being limited to only one region of simulation (Harrison 2011).

In contrast to traditional HIL, HPS can have multiple regions of real and virtual interacting (Harrison and Tilbury 2008) and the mathematical description developed does not rely on the accuracy of its individual simulated components for assessment. A HPS is a modular simulation approach where models of components (robots, machines, conveyors, controllers) can be replaced seamlessly by their actual (real and physical) counterparts (Harrison, Tilbury, and Yuan 2011). The simulation runs in real-time, and is used for validation analysis and design decisions. Using this approach, we can easily integrate simulation into the development process and throughout the product life cycle (Harrison, Tilbury, and Yuan 2011). The HPS methodology was developed and implemented using the Reconfigurable Factory Testbed (RFT) at the University of Michigan, consisting of two Fanuc M6iB robots, four CNCs and a conveyor (Moyne, Korsakas, and Tilbury 2004). For further details regarding HPS environment, refer to (Harrison and Tilbury 2008).

3.3 Development of a Manufacturing System Using HPS

We have so far discussed the benefits of HPS and the extent of its capabilities. But how can it be applied to an industrial setting such that it offers the maximum benefit to the user? For this be answered, let us consider a specific scenario in which a new assembly plant for XYZ Company is to be designed and launched. One possible approach would be to suggest that design start with a virtual model layout of the plant. Simulating the virtual model first will enable us to determine for instance what type of component to use (from a vast variety of vendors and processes), which configuration achieves the highest throughput and so on. This leads to the development of a tentative plant layout. The next step from this pure virtual model is to develop a hardwired simulation using vendor-supplied virtual components that emulate the real components and support the native data communication. A hard-wired simulation is an instance of HIL methodology. Here real controllers that communicate with the devices are now developed and implemented with the vendor supplied software, such that the all devices except the controllers are virtual models but the communication infrastructure between the devices and the controllers is real and not simulated. This avoids the issues mentioned earlier in modeling data communication networks. The final step involves incremental replacement of the virtual model for each device with the actual device of the same vendor. Once the communication infrastructure is established and it is ascertained the virtual model functions as desired, it can be disconnected from the hard-wired simulation and the actual device is connected in its place. This reduces ramp-up time to plant deployment. The complete process, as summarized in Figure 1, thus provides a platform for pre-launch validation of control code and system connectivity.

What now if XYZ considers replacing a device from Vendor A in plant with a similar device from Vendor B? The modular nature of HPS means that it can be reused, making it highly applicable to reconfiguration. Firstly, a virtual component of a candidate replacement device can be tested with the real system in the plant. Secondly, a real candidate replacement device can be validated with a virtual model of the system. Only once the results are satisfactory is the actual device installed in the physical system. In this manner, HPS can help with the replacement of malfunctioning devices and provides a platform for quick evaluation of multiple reconfiguration scenarios.

The above example illustrates a novel application of HPS in the design, launch and reconfiguration of a production line. We will outline in the following sections a methodology for connecting and swapping a virtual robot with a real robot, as a proof-of-concept for the methodology explained above.

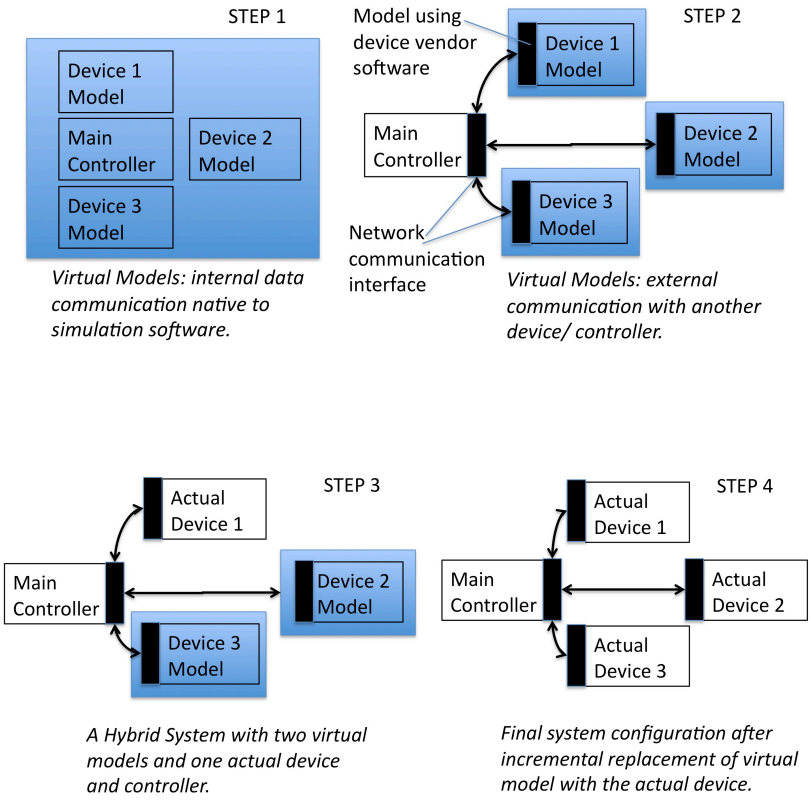

Figure 1: Example of HPS application in launching a production line, showing migration from pure simulation to a hard-wired simulation to incremental replacement of each virtual device until a fully functional plant is obtained.

4 METHODOLOGY FOR INTEGRATING A SINGLE VIRTUAL ROBOT IN A HPS

It is not always possible to obtain a virtual replica of a real component capable of native data communication from the vendor. Thus, we summarize the process of creating a virtual replica of a real device in three main steps: creation of the device's virtual counterpart, installation of component libraries for the virtual device obtained from vendor, and finally the creation of an interface program using the component libraries that connect the virtual device to another physical/virtual device. To illustrate this process, we created a virtual replica of a robot communicating over DeviceNet with a physical system. The details of such a process are discussed in Section 5.

There have been several works focusing on the creation of virtual models (see Section 3.1) and this paper will not provide a generic outline on their importance and development process. Section 5.1 of this paper discusses details specifically on our development of the virtual robot.

The second step is to install component libraries for the virtual device. The libraries provide control of a virtual device through the interface program existing outside the simulation software environment. They are usually obtained from the vendor of the device and simulation software. Since we require the virtual device to be able to send and receive I/O signals to other devices in the system in the same way its physical counterpart would, and since extended control of the virtual device from a point external to the software is desirable, it is preferable to use the device vendor software, since it is more likely to come with the necessary libraries. Component libraries are usually installed when the component is set up.

Finally an interface that allows communication between the device model and the rest of the devices is developed. The interface program exists as an intermediate program to facilitate communication between the virtual component and other devices in the system. The interface should be easy to reuse, such that switching the between virtual component and the actual device or simply disconnecting and connecting the virtual component is relatively simple. Our interface program sends/ receives I/O signals from the controller and, using the component libraries, sets the task or the state of the virtual component based on the I/O signals. Hence, at the high level, the interface program can be thought of as having two sides: one that communicates with the virtual component and records the I/O signals; and the other that communicates with the another device and keeps track of the I/O signals.

5 CASE STUDY

Figure 2 illustrates an overview of how the interface program is set up. The virtual robot represents a real M6-iB R-J3 Fanuc robot and was developed using Roboguide simulation software, also provided by Fanuc Robotics. Since the main purpose of Roboguide is to purely simulate Fanuc robots and give details on throughput, task completion rates and robot movements, an interface program is needed to actually be able to communicate using external I/O with the virtual robot. Fanuc's PCDK (PC Developer's Kit) software includes necessary libraries to facilitate the external I/O communication with the virtual robot. The external communication was done using DeviceNet to a main PC controller carrying a DeviceNet card and scanner.

5.1 Creation of a Virtual Robot in Roboguide

Since the real robot being simulated was a Fanuc R-J3 M-6iB type robot, the Fanuc Roboguide software was used to create the virtual robot that can then be programmed using the internal built-in teach pendant and run in test mode to verify the process. (Viswanathan 2008) gives details on how to create a virtual robot cell.

5.2 Installation of PCDK Library

The necessary robot libraries were available through Fanuc's PCDK (PC Developer's Kit) software. PCDK is the development environment that provides a COM based Robot Server for communication between a real robot and a PC. Robot Server is an Active-X executable program that provides the user PC access

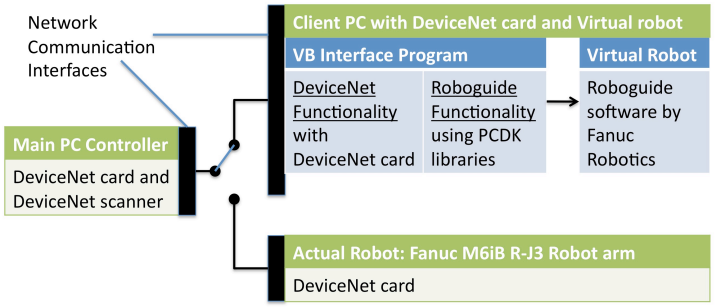

Figure 2: Illustration of the swapping process between the real and the virtual robots, and the structure of the interface program for the virtual robot that allows this swapping.

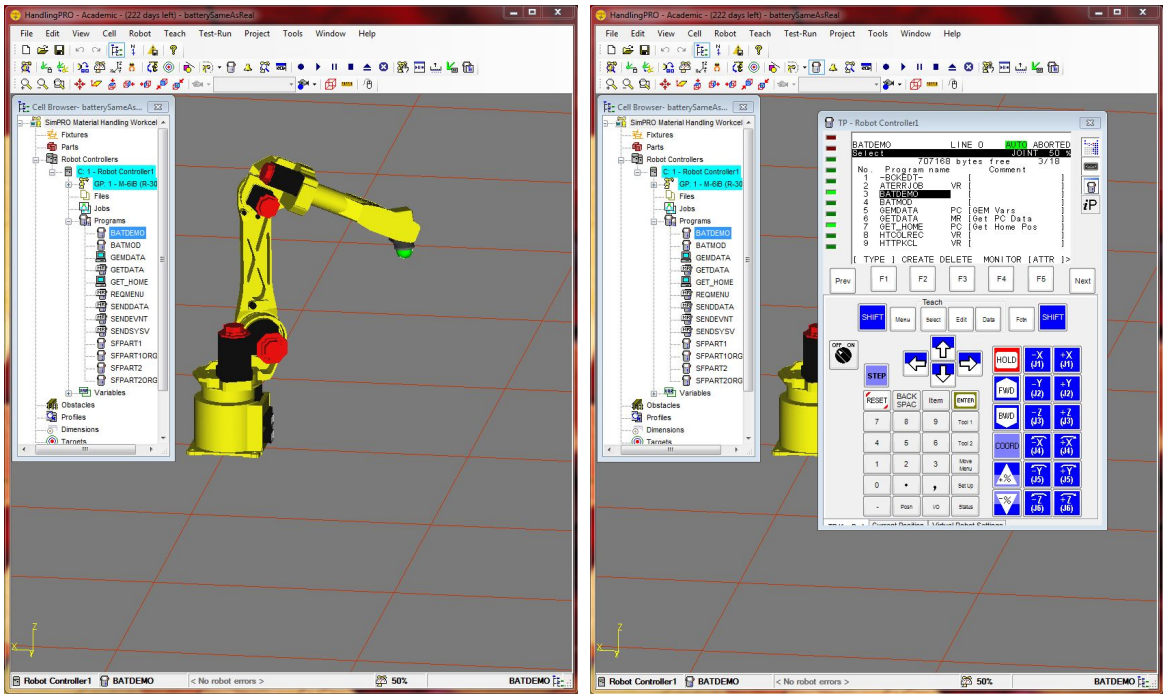

Figure 3: The virtual environment in Roboguide showing the Fanuc M6-iB RJ3 robot and the internal teach pendant used for programming the robot.

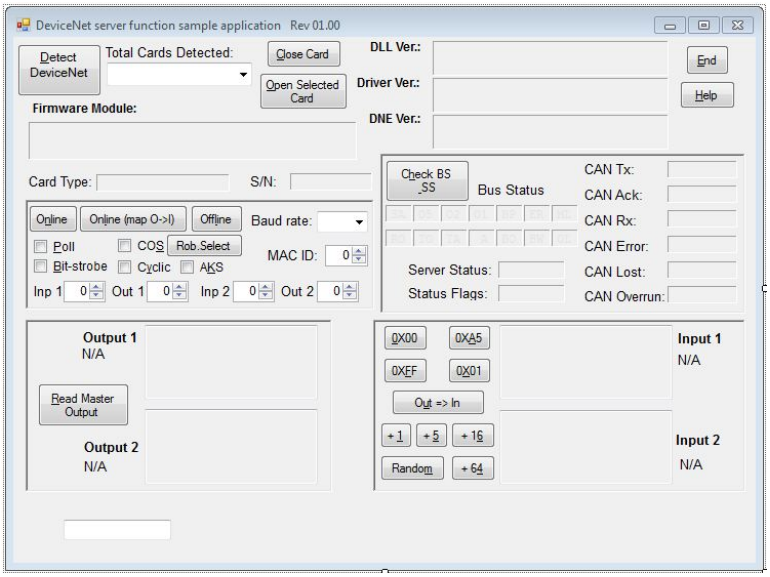

Figure 4: GUI for DeviceNet Communication Configuration.

to the real robot controller. (Fanuc 2008) provides details on Robot Servers and using PCDK. We used these 'libraries' of software codes and examples provided by the kit to write the interface program between the controller and the virtual robot. We can then use the program to control the virtual robot through a computer, which in turn can communicate I/O signals from the main controller. Details on the installation of PCDK can be found in (Viswanathan 2008).

5.3 Development of Interface Program in Visual Basic 2008

The program for controlling the robot is developed in Visual Basic Express. Desired libraries are imported from PCDK into Visual Basic and used as references to write the program to start/ stop the robot. As explained in Section 4, the interface program can be thought of as having two sides (see Figure 2): one that communicates with the main controller (carrying DeviceNet functions) and the other that communicates with the virtual robot (carrying Roboguide functions). The following sections describe the implementation of each of the sides.

5.3.1 DeviceNet Communication Using DeviceNet Card

The client computer needed to be able to send and receive DeviceNet I/O from the controller. The client has a SST-DN3 card for all DeviceNet communications. In order to establish DeviceNet communication, a number of classes, modules and functions for demonstrating the server functionality for the SST-DN3, SST-DN4 and 513 6-DNP family of cards (manufactured by Woodhead Industries) are written in Visual Basic (See Figure 4). The code contains functionality for performing operations such as loading the drivers, opening the card, bringing it online, checking the bus status and so on. For details, refer to (Viswanathan, Tilbury, Harrison, and Gu 2011).

5.3.2 Roboguide Communication Using PCDK Libraries

The virtual robot in Roboguide needs to be able to recognize the device I/O and run the appropriate assembly program. As with DeviceNet communication, connectivity to Roboguide was obtained using a number of classes, functions and structures written in Visual Basic (see Figure 5) to perform tasks such as handling a

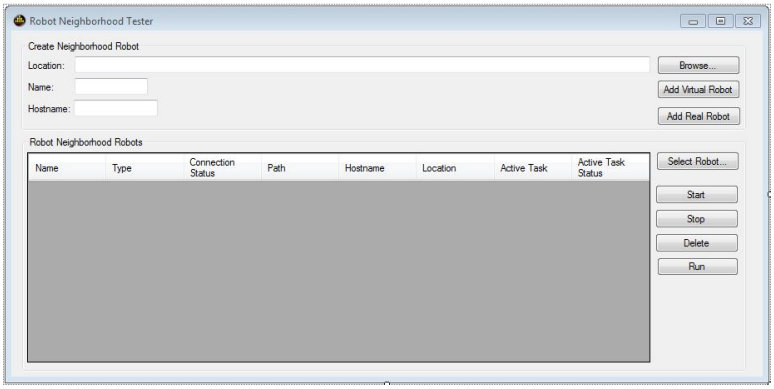

Figure 5: GUI for Interaction with Roboguide. Once connectivity to the virtual robot is established, the robot is in a wait state listening for messages from the controller. Note that no further configuration for the virtual robot is necessary, once it is created with the desired properties and programs from the vendor software (refer to Section 5.1).

Table 1: A list of I/O messages sent as bit signals between the controller and the virtual robot.

	Main Controller	Virtual Robot
I/O Messages Sent by:	Place part 1 in CNC 1	- Placing part 1 in CNC 1 - Part 1 in place
	Place part 2 in CNC 2	- Placing part 2 in CNC 2 - Part 2 in place
	Retrieve part 1 from CNC 1	- Retrieving part 1 from CNC 1 - Part 1 retrieved
	Retrieve part 2 from CNC 2	- Retrieving part 2 from CNC 2 - Part 1 retrieved

change in the connection status or robot task status and checking and returning robot properties. The code was developed in collaboration with Fanuc Robotics.

For every DeviceNet I/O signal sent from the controller, the virtual robot performs a specific task. While performing this task, a series of digital signals are written to the controller to signify the different stages of task completion (see Table 1 for a summary of the I/O messages exchanged for placing and retrieving two parts in two CNC machines). Once all the signals have been sent to the scanner, the controller recognizes that the virtual robot has completed the task and can send a new I/O signal.

5.4 Summary of Work Done

Once the interface to the virtual robot was developed, we demonstrated that we were able to run the virtual robot as a component of the RFT manufacturing system after switching it with its real counterpart Fanuc robot. All of the other pieces of the system (conveyor, controllers, CNC machines) are real. The virtual robot has the same address on DeviceNet as the real robot. Since the virtual robot sent and received the same signals as the real robot, over the same network (DeviceNet), and using the same address, none of the other components of the system had to change – the controllers operate in exactly the same way.

6 CONCLUSION

An interface for connecting a virtual robot to a physical automated production system via native I/O communication network was developed and it was demonstrated that the interface works for the setup depicted in Figure 2. Real time communication between the controller and the virtual robot was achieved using this process, and the virtual robot behaved in essentially the same way as the real robot. This demonstrates a proof-of-concept for the potential industrial application of the HPS in the following areas:

1. pre-launch validation of control code and system connectivity,
2. quick evaluation of multiple scenarios of reconfigurability.

We were able to demonstrate our approach of connecting individual device models through an interface program to other devices successfully and use it for troubleshooting and repair purposes anytime the actual device failed. However, the testing of interface applications was limited to a custom VB interface particularly written to link the I/O signal from the virtual robot in Roboguide to a DeviceNet PC card. Further research may be needed to find out how the interface changes if a different communication network was used (e.g., Profibus or Ethernet instead of DeviceNet) or if a robot from a different vendor and thus different vendor simulation software (e.g., ABB robots) were used.

One aspect of the HPS that has not been investigated in this paper is the part handling. Whereas the real robot can pick up a part off the conveyor and put it into a machine, the virtual robot cannot. Thus, when the HPS is operating with the virtual robot, the part must go from the real world into simulation (virtual robot) then into real (machine) then back to simulation (virtual robot) then back to real (conveyor). The theory of how these multiple transitions from virtual to real and back again is discussed in (Harrison, Tilbury, and Yuan 2011); in short, the part must be simulated through all of the transitions and a "part emulator" must be used to trigger the appropriate sensors (e.g., part presence).

Another aspect that has not been explored is how the virtual robot could be run in parallel to the real robot for monitoring purposes. Since the two robots share the same DeviceNet address, they cannot both run in parallel, since it is not possible for two nodes to occupy the same address on the network. When considering a future expansion in this direction, the synchronization between the monitoring environment (virtual) and the physical environment (real) must be addressed. Both the real and virtual devices can receive the same inputs at the same times, but if they both send outputs the controller would need to know which one to react to.

ACKNOWLEDGMENTS

We are grateful to Fanuc Robotics for their invaluable help provided in making the external I/O API available for Roboguide. Special thanks to DeviceNet vendor for providing us with code for a software that can communicate DeviceNet over Visual Basic. Finally, the authors of this paper would like to thank all colleagues and students who helped contribute to this project and to the Engineering Research Center (ERC) facility at University of Michigan for providing the funding for this project.

REFERENCES

Aydt, H., S. Turner, W. Cai, and M. Low. 2008. "Symbiotic Simulation Systems: An Extended Definition Motivated by Symbiosis in Biology". In *22nd Workshop on Principles of Advanced and Distributed Simulation*, 109–116.

Baheti, R., and H. Gill. 2011. "Cyber-physical Systems". In *The Impact of Control Technology*, edited by T. Samad and A. Annaswamy, Volume Part 3, 161–166.

Breton, E., and J. Bezivin. 2001. "Model Driven Process Engineering". In *25th Annual International Computer Software and Applications Conference, COMPSAC 2001*, 225–230.

Chen, L., P. Bender, P. El-Wardany, and T. Renton. 2002. "Integrated Virtual Manufacturing Systems for Process Optimization and Monitoring". *CIRP - Manufacturing Technology* 51:409–412.

Diogo, R., C. Vicari, E. Loures, and M. Busetti. 2008. "An Implementation Environment for Automated Manufacturing Systems". In *Proceedings of the 17th International Federation of Automatic Control World Congress*, 10552–10557.

Fanuc 2008. "Help Topics: PC developers Kit. V7.40 (Rev. A).". Technical report, Fanuc Robotics.

Fowler, J., and O. Rose. 2004. "Grand Challenge in Modeling and Simulation of Complex Manufacturing Systems". *Simulation* 80:469–476.

Harrison, W. 2011. *Virtual Fusion: The Integration and Analysis of Simulation and Real Processes for Manufacturing Process Deployment*. Ph. D. thesis, University of Michigan.

Harrison, W., and D. Tilbury. 2008. "Virtual Fusion: Hybrid Process Simulation and Emulation-in-the-Loop". In *Proceedings of the 9th Biennial ASME Conference on Engineering Systems Design and Analysis*, Volume 1, 263–270.

Harrison, W., D. Tilbury, and C. Yuan. 2011. "From Hardware-in-the-Loop to Hybrid Process Simulation: The Complete Integration of Simulated and Actual". *To Appear in IEEE Transactions on Automation Science and Engineering*.

Moyne, J., J. Korsakas, and D. Tilbury. 2004. "Reconfigurable Factory Testbed RFT: A Distributed Testbed for Reconfigurable Manufacturing Systems". In *Proceedings of the Japan-USA Symposium on Flexible Automation*.

Ng, A. H., J. Adolfsson, M. Sundberg, and L. J. D. Vin. 2008. "Virtual manufacturing for press line monitoring and diagnostics". *International Journal of Machine Tools and Manufacture* 48 (5): 565–575. Advances in Sheet Metal Forming Applications.

Park, S. C. 2005. "A methodology for creating a virtual model for a flexible manufacturing system". *Computers in Industry* 56 (7): 734–746.

Peters, B., J. Smith, J. Curry, and C. LaJimodiere. 1996, December. "Advanced Tutorial - Simulation-Based Scheduling and Control". In *Proceedings of the 1996 Winter Simulation Conference*, edited by J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, 194–198. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Pohit, G. 2006. "Application of virtual manufacturing in generation of gears". *International Journal of Advanced Manufacturing Technology* 31:85–91.

Viswanathan, J. 2008. "ME 490 Report: Hybrid Interfacing Between Cell Controller and Virtual Robot". Technical report, Mechanical Engineering, University of Michigan, Ann Arbor.

Viswanathan, J., D. Tilbury, W. Harrison, and F. Gu. 2011. "Interfacing a Virtual Robot with a Physical Cell Controller using the Native DeviceNet Communication". Technical report, Mechanical Engineering, University of Michigan, Ann Arbor.

Wang, L., P. Orban, A. Cunningham, and S. Lang. 2004. "Remote real-time CNC machining for web-based manufacturing". *Robotics and Computer-Integrated Manufacturing* 20:563–571.

Weston, R. 1999. "Model-driven Component Based Approach to Reconfiguring Manufacturing Software Systems". *International Journal of Operations and Production Management* 19 (8): 834–855.

AUTHOR BIOGRAPHIES

JANANI VISWANATHAN is a graduate student in Mechanical Engineering at the University of Michigan. Her specialization is in the area of controls and robotics, coupled with her research interests in virtual fusion and development and applications of hybrid process simulations. She obtained her B.Sc in Mechanical Engineering from University of Michigan in 2008. She belongs to *Pi Tau Sigma* mechanical engineering honor society. Her email address is baboo@umich.edu.

WILLIAM HARRISON obtained his PhD student in Mechanical Engineering at the University of Michigan. He received his Bachelors in Mechanical Engineering from the University of Michigan, and his

Masters in Materials Science Engineering from the University of Florida. His interests include virtual fusion, virtual reality, simulation integration, and game engine utilization. His e-mail is wsh@umich.edu.

DAWN TILBURY is a Professor of Mechanical Engineering at the University of Michigan. She received the B.S. degree in Electrical Engineering, *summa cum laude,* from the University of Minnesota in 1989, and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Sciences from the University of California, Berkeley, in 1992 and 1994, respectively. Her research interests include distributed control of mechanical systems with network communication, logic control of manufacturing systems, reliability of ground robotics, and dynamic systems modeling of physiological systems. She belongs to ASME and SWE, and is a Fellow of IEEE. Her email address is tilbury@umich.edu.

FANGMING GU is a Staff Researcher at the Manufacturing Systems Research Lab of General Motors Global Research and Development. He received his Ph.D. degree in Mechanical Engineering from the University of Illinois at Champaign-Urbana in 1994. His research interests include machining process simulation, wireless technology applications in manufacturing, controls validation of manufacturing automation system and sustainable manufacturing systems. His email address is fangming.gu@gm.com.