Proceedings of the 2007 Winter Simulation Conference
S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, eds.

ENABLING INDUSTRIAL SCALE SIMULATION / EMULATION MODELS

Michael Johnstone
Doug Creighton
Saeid Nahavandi

Intelligent Systems Research Lab
Deakin University
Pigdons Rd, Waurn Ponds
Victoria Australia

ABSTRACT

OLE Process Control (OPC) is an industry standard that facilitates the communication between PCs and Programmable Logic Controllers (PLC). This communication allows for the testing of control systems with an emulation model. When models require faster and higher volume communications, limitations within OPC prevent this. In this paper an interface is developed to allow high speed and high volume communications between a PC and PLC enabling the emulation of larger and more complex control systems and their models. By switching control of elements within the model between the model engine and the control system it is possible to use the model to validate the system design, test the real world control systems and visualise real world operation.

1 INTRODUCTION

A 3D model coupled with Discrete Event Simulation (DES) enables the creation of real world systems for visualisation and analysis. With such powerful tools complex systems can be modelled and a multitude of questions asked about the system, questions ranging from analysis of the system performance to "what if" scenarios. These models can be made more valuable with the realisation that they can be used to also test real world control systems. Emulation, testing a control system via a computer model, allows for the off line development of control programs and testing of changes and numerous benefits detailed in literature.

In a complex system it may be desirable to test only parts of the control system. Glinsky et al. (2004) in their modelling of a hardware-in-the-loop system incrementally moved elements from the model into the real world as they become available. This same concept can be applied to emulation models. Control of elements within the model can be passed back and forth between the external control system and the simulation engine, allowing testing of individual parts of a control system or the entire control system.

Now with a 3D model of the system being controlled by the real world control system a powerful visualisation tool is available that can be used once the system is live. Data from the control system can be used to drive the model to represent what is occurring in the real world to allow for monitoring of the system. The original model created for validation of design or analysis can be used for much more than just the original purpose.

The paper is laid out as follows. Section 2 gives a review of the previous work in emulation while section 3 defines hybrid environment and how this was achieved. Section 4 gives the results for testing carried out with the interface defined in section 3, while section 5 gives a summary of the work and suggests areas where this new environment is suitable.

2 REVIEW OF PREVIOUS WORK IN EMULATION

The objective of emulation, or soft-commissioning as it has also been referred to as (Schludermann et al. 2000;Versteegt et al. 2002), is to connect actual real world control systems to simulation models to test the operation of those control systems (Schludermann et al. 2000;Schiess 2001;McGregor 2002). Successful emulation implementations have been achieved in varying fields, for example, baggage handling systems (Rengelink et al. 2002) and material delivery systems (Lebaron et al. 1998).

Historically the testing of control systems was achieved by connecting individual test or mock-up devices to elements of the control system. This method of testing required considerable time and resources and failed to test the system as a whole (Whorter et al. 1997;Schludermann et al. 2000). Due to this lack of adequate testing options

(Rengelink et al. 2002) stated that the quality of control software was adversely affected, as was lead time to deliver a correctly functioning system. Auinger et al. (1999) describes four methods for testing control systems using a combination of simulation models and real world objects; traditional testing where control and hardware are real, emulation where control is real and hardware is simulated, reality in the loop where the control is simulated and hardware real and finally off-line testing where both control and hardware are simulated. Versteegt et al. (2002) took these methods further whereby the control logic in the emulation test was simulated. To avoid a "credibility gap" (McGregor 2002) with this approach, the actual software program used in the control simulation is the software program used to control the real hardware. This approach is similar to that of a PLC simulator as used in McGregor et al. (2001), and as it uses the actual control software, the problems identified by Rengelink et al. (2002) in regard to separate control programs for emulation and real world control are negated.

Emulation in this form has been used successfully with real benefits to both integrators and customers (Lebaron et al. 1998;Mueller 2001). Time and money can be saved with the use of emulation, debugging controller logic can found in the lab rather than on the shop floor, saving on on-site install costs and lost production (Schiess 2001). Re-implementing control logics is not required as you are using the actual controller system (Lebaron et al. 1998;Vedapudi 2001). Other benefits include complete control system testing, a training environment for staff, reduced installation risk (McGregor et al. 2001), increased product quality and reliability, reduced testing time and faster debug time (Whorter et al. 1997), and the ability to test without disturbing production (Jacobs et al. 2005).

Simulation and emulation models share a 3D representation of the system being modelled, are accurate and consist of realistic modelling objects. However the differences between the two define the role of emulation. Simulation models test different solutions to achieve a desired result at high speed while the aim of an emulation model is to test a control system in real time (McGregor 2002). Emulation is concerned with the control of a system and the interface to the controlling system whereas a simulation models the behaviour a system. In order to create the emulation model two things need to be decided on, an interface to the control system from the simulation model and a communication protocol. (McGregor 2002) used OPC (OLE Process Control) as the communication protocol and wrote an OPC client to embed into the simulation model. Since then (Jacobs et al. 2005) wrote directly to the PLC over a TCP/IP network in order to preserve the real world configuration of the control system. The time taken for communication between the model and control system will differ for each different communication methodology, (Lebaron et al. 1998) states the need to be careful with time difference between the two elements, ideally the faster the communication method the better, as this will allow for more data and larger models to be emulated. The aspects of an emulation model to take special interest in are the interface to the control system and the frequency and volume of data transferred.

Rengelink et al. (2002) modelled a baggage handling system, BHS using a PLC as the control system. A serial profibus connection was implemented between the PLC and the 2D model and up to 70 conveyors were being controlled at a time. The authors state that improvements need to be made to the model visualisation, multiple threads to share processor loads and the speed of data transfer, among others, to improve on the overall abilities of the emulation model.

In the work by Jacobs et al. (2005), the MODBUS protocol running over TCP/IP was used to connect the simulator and the control system, a PLC. This provides a medium for improved data transfer over serial profibus mentioned previously. A section of memory was used as a buffer between the two elements to reduce communication, but volume of data was not mentioned. Data transfer occurred within with a period 30 milliseconds and this period was logged to later verify it was met. This period was achieved with the use of multiple threads, communication, simulation and animation threads. This emulation model has the framework for improved data transfer, volume and speed, via the use of Ethernet between the model and PLC and makes use of threads to share processor load in order to meet time deadlines. The model graphics are once again 2D, therefore taking similar ideas to a different simulator it would be possible to have 3D model being controlled, via Ethernet, by a PLC. The size of this model and the period of data transfer would determine the size of the model able to be emulated.

Versteegt et al. (2002) developed an emulation model of AGVs in an automated material handling system. Communication in their model was based on a poll to the communication buffer every 10msec to check for any changes from the control system. The authors also found that by using asynchronous communication that they were able to improve the performance of the emulation model. However they are still unsure as to how well their particular implementation would scale for larger systems.

3 ENVIRONMENT OVERVIEW

In this section we will describe the various elements in the environment using a Baggage Handling System (BHS) as an example. In our environment we have the simulation model running on a PC, a PLC acting as the control system and the interface between the two. Time issues, sequences and control strategies are also described.

3.1 Simulation Environment

The simulation environment is made up of elements such as conveyors, automatic tag readers (ATR), explosives detection systems (EDS) machines, etc, see Figure 1. These elements are controlled via signals. To simulate a conveyors control you would typically use three signals, one input signal to control the motor and two output signals, a pin wheel to measure belt movement and a photo eye to detect bags moving along the belt. A run time parameter is used to define if the element is to be controlled by the simulation engine or external PLC. When the element is controlled by the PLC the simulation engine generates the output signals and responds to the input signals. Based on the output signals the PLC logics decide when to change the input signal. When the element is controlled by the simulation engine, the signal responses and generation are the same however there are additional logics used to respond to output signal changes to control the input signal, ie the function performed by the PLC is simulated.

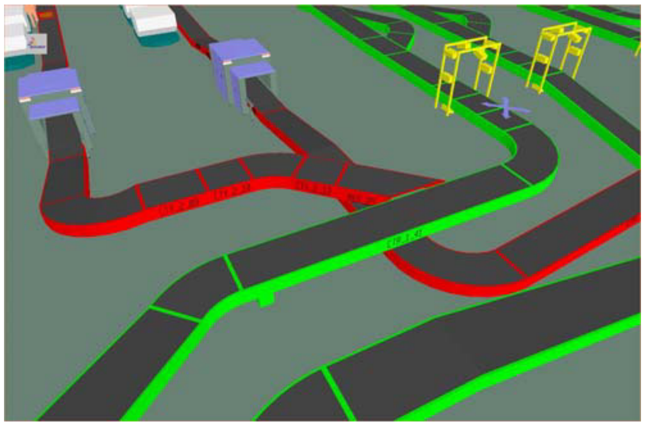
Figure 1: An image of a BHS showing EDS machines, ATRs and conveyors.

3.1.1 Threads

As it takes time to exchange data between the model and the PLC the model can operate in 2 distinct ways. Firstly by pausing and waiting for completion of the data exchange or secondly, allowing a separate process to handle the data exchange while the model continues processing. The former option does not provide the control delay normally seen in the real world, mentioned previously, and requires the simulation to run faster than real time to make up for the delay in waiting for the data exchange to complete. The latter allows the simulation to run at a constant rate and provides the control delay. Threads are used to make the data exchange asynchronous.

The simulation has a main application thread that uses worker threads to exchange date with the PLC. At every desired time interval a read or write is sent by the main thread to a worker and polls for the workers completion. The worker thread communicates with the PLC and upon completion provides the data to the main thread where the data is acted upon. Currently there is a limitation set in the simulation software that does not allow a worker thread to update elements within the model.

3.1.2 Time

The simulation model is required to run in real time as this is what the PLC is running in. So to slow down the simulation clock a PID controller was used. The controller adjusts simulation update rate, the rate at which graphics are updated. The smaller this value the slower this simulation runs.

3.2 Control Environment

A PLC forms the main component of a BHS control system. Usually more than one PLC is used for a variety of reasons such as redundancy and load. PLCs run ladder logic programs that modify outputs based on the inputs. Continuing with our example of a conveyor, the inputs would be signals from the photo eye and pin wheel, outputs would be the signal to the motor.

PLC execution is sequential. It runs programs, refreshed inputs and outputs and responds to external commands. This loop is executed as quickly as possible. One loop is called the cycle time. Cycle time defines the time it will take the PLC to respond to changed input conditions and also determines the time taken to reply to external commands. The cycle time must be lower than that of the cycle time required by the pin wheel, 25ms, or else the PLC will miss pin wheel events and not correctly track conveyor movement.

3.3 The Interface

The interface is depictured in Figure 2. Here a PC, connected via Ethernet to the control system, is running the model. The PLC is running a ladder logic program to control the elements in the real world that have been modelled on the PC. The model connects to the PLC to exchange data, eg writes photo eye and pin wheel information and reads motor information. The model updates the simulation according to the information read from the PLC. The PC and the PLC are connected via Ethernet. The first protocol tested between the two was OPC and subsequently another protocol was tested due to the performance of OPC.

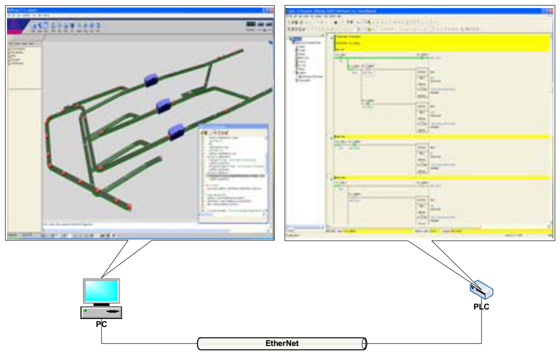

Figure 2: Model to PLC Interface

3.3.1 Communication Protocols

OPC is made up of an OPC Server and Client. The OPC server software was provided by the PLC manufacturer. The server communicates with the PLC over Ethernet and provides a software interface for clients to connect to in order to read or write data to or from the PLC. The server software can also provide access to a PLC simulator in place of a real PLC. The OPC client software was written and embedded into the model, see Figure 3. As the model ran, the client would send/receive data from the OPC server.

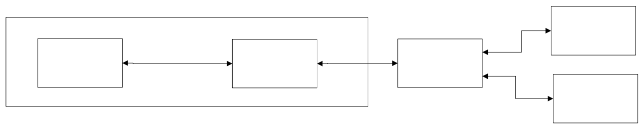

Figure 3: OPC communications overview

Initially a PLC Simulator was used with OPC. When a signal was changed by the PLC the OPC server would cause an event within the client, and the model would update as appropriate. To send data to the PLC the client initially used a synchronous write command. Here the simulation would pause as it waited for the write command to receive a write successful response from the OPC server.

When a simple model containing several elements was run, the model ran slower than real time due to this synchronous data exchange. The PLC simulator was replaced by a real PLC and the model run, again the model could not run at real time. The write command was changed to an asynchronous command and the simulation run. The simulation was able to run at real time however there was a delay between the PLC changing a signal and that change being reflected in the model. This delay was due to an update rate parameter built into OPC that defined the smallest time interval which a signal could be updated. This minimum value of the update rate was 100ms, therefore OPC could not accommodate a pin wheel signal that changes every 25ms.

A final test of OPC was to increase the number of elements in the model. As more elements were added to the model the OPC server started to lag. It was found that the OPC server would write data to the PLC one signal at a time. Therefore as the model increased the OPC server would queue the writing of signals to the PLC.

As the PLC had Ethernet capabilities, investigation into communicating directly with it was carried out via the use of a TCP or UDP socket, where we could hopefully write faster than OPCs 100ms and in greater volume. The protocol used to communicate over sockets by the PLC is FINS, which was the communication protocol used between the OPC server and the PLC. FINS allows for individual or bulk read/writes to/from the PLC, so where the OPC server would write to ten consecutive inputs ten individual times, with FINS it could be done with one write command if the five items to be written were in consecutive in memory locations., this is show in Figure 4.

The FINS protocol over a socket provides the ability to update many signals quickly if they are consecutive in the PLC memory, It was possible to read or write 1024 signals within 3.5ms.

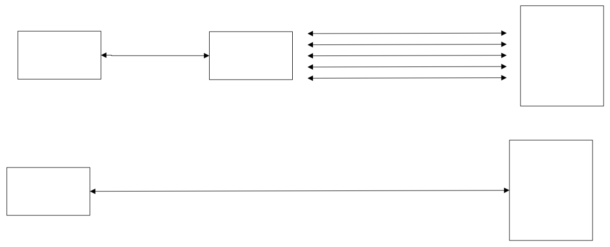

Figure 4**:** Pictorially showing the reduction in communication using FINS versus OPC.

3.3.2 Sequence of Events

As we are interested in the status of signals over time the model must be continually reading and writing data to and from the PLC. To achieve this the model loops through the following sequence. The loop has a period of 10ms to match PLC cycle time.

The model sequence:

1. Determine what data is to be sent
2. Use thread to send it
3. Use thread to read data from PLC
4. Sleep for a typical amount of time and then start to poll the read thread for completion.
5. Upon completion take action
6. Sleep for remainder of cycle time

3.4 Control

By varying the employed control strategy differing environments are achieved. When all control is handled by the simulation engine we have a simulation model where the normal analyses can be performed and "what if" questions answered. This is the first and often the only purpose given to a detailed model of a system.

When control is passed to the PLC we have an emulation model where the controller logic can be verified off-line using data input from the simulation model.

If the write commands are disabled in the simulation model and control is still with the PLC we have a visualisation model where we can determine actions to take based on data received from the PLC. In this instance additional steps have to be performed in the model like creating bags when the PLC detects them as opposed to using previous methods.

4 TESTING

The interface was tested by reading and writing from the simulation model to the PLC and capturing the packets using packet capturing software. The packet capture records the transmission times of the packets, enabling the response time of the PLC to be determined.

Conclusion

The benefits of using an industry standard for communication between a PC and a PLC are clear. OPC provides this standard, enabling the one PC software application to communicate with multiple PLC brands provided the PLC company has an OPC server to suit. However when attempting to emulate a higher number of connections to the PLC or emulate at high speed problems are encountered. By communicating from PC to PLC directly faster and higher volume communications can be achieved. The interface developed enables high speed communication allowing emulation of larger models.

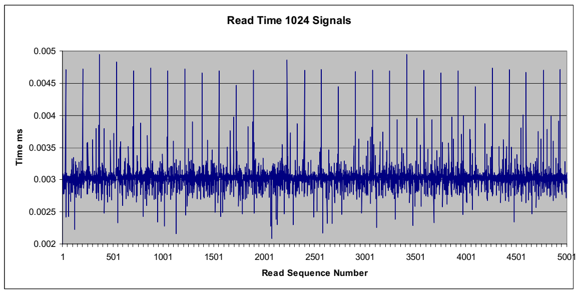

Figure 5 shows the time it took the PLC to respond to 5000 read commands that requested the status of 1024 signals. The average time was 3.03ms ± 5us. Similar times were recorded for writing to the PLC. These results compare favorably with past response times in literature, Jacobs et al. (2005) has a requirement for a 30ms cycle time, while Versteegt et al. (2002) were polling a communications buffer every 10ms.

The testing confirms that it possible to read from the control system enough information to run a large model. If we were to read the status of the 3 signals used in conveyor control as described in section 3.1 then we would be able to interrogate the status of 340 conveyors. The data requested from the PLC is required to be contiguous with the memory structure of the PLC so it can be quickly obtained with one read command, or not spaced at either ends of the PLC structure so that the entire memory structure is required to be read. In the case where the data required is not contiguous it is up to the PC program to isolate the required data from the response packet from the PLC.

5 CONCLUSION

The benefits of using an industry standard for communication between a PC and a PLC are clear. OPC provides this standard, enabling the one PC software application to communicate with multiple PLC brands provided the PLC company has an OPC server to suit. However when attempting to emulate a higher number of connections to the PLC or emulate at high speed problems are encountered. By communicating from PC to PLC directly faster and higher volume communications can be achieved. The interface developed enables high speed communication allowing emulation of larger models.

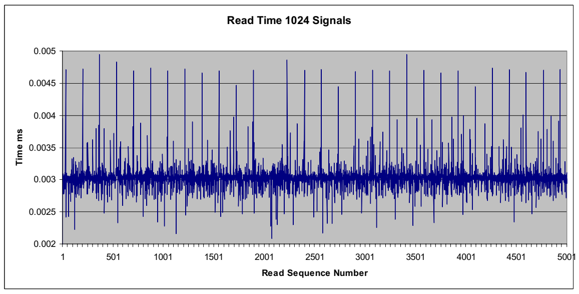

Figure 5 PLC time to respond to a read command requesting 1024 signals

With the ability to read data from a PLC at high speed it is possible to drive a model purely to visualise what is occurring in the real world. This application would enable a model that was used during the design process to prove ideas and answer "what if" questions and also used to test and develop control systems, to be used as monitoring tool during production.

Further work in this area would be to test against more PLC manufacturers to test that high speeds can be achieved with them also.

ACKNOWLEDGMENTS

This work was funded by the Australian Research Council.

REFERENCES

Auinger, F., M. Vorderwinkler and G. Buchtela 1999. Interface Driven Domain-Independent Modeling Architecture For "Soft-Commissioning" And "Reality in the Loop". In *Proceedings of the 1999 Winter Simulation Conference*. 798-805. Squaw Peak, Phoenix.

Glinsky, E. and G. Wainer 2004. Modeling and Simulation of Hardware/Software Systems with Cd++. In *Proceedings of the 2004 Winter Simulation Conference*. 198-205. Washington DC.

Jacobs, P. H. M., A. Verbraeck and W. Rengelink 2005. Emulation with Dsol. In *Proceedings of the 2005 Winter Simulation Conference*. 1453-1462. Orlando, FL.

Lebaron, H. T. and K. Thompson 1998. Emulation of a Material Delivery System. In *Proceedings of the 1998 Winter Simulation Conference*. 1055-1060. Washington DC.

McGregor, I. 2002. The Relationship between Simulation and Emulation. In *Proceedings of the 2002 Winter Simulation Conference*. 1683-1688. San Diego, CA.

McGregor, I. and R. A. J. Walters 2001. "Emulation Overview." Via http://www.automod.de/media/doc/Mcgregor.pdf [accessed June 8, 2007]

Mueller, G. 2001. Using Emulation to Reduce Commissioning Costs on a High Speed Bottling Line. In *Proceedings of the 2001 Winter Simulation Conference*. 1461-1462. Arlington, Va.

Rengelink, W. and Y. A. Saanen 2002. Improving the Quality of Controls and Reducing Costs for on-Site Adjustments with Emulation: An Example of Emulation in Baggage Handling. In *Proceedings of the 2002 Winter Simulation Conference*. 1689-1694. San Diego, CA.

Schiess, C. 2001. Emulation: Debug It in the Lab - Not on the Floor. In *Proceedings of the 2001 Winter Simulation Conference*. 1463-1465. Arlington, Va.

Schludermann, H., T. Kirchmair and M. Vorderwinkler 2000. Soft-Commissioning: Hardware-in-the-Loop-Based Verification of Controller Software. In *Proceedings of the 2000 Winter Simulation Conference*. 893-899. Orlando, FL

Vedapudi, S. 2001. "Using Mcm to Do Emulation of a Car Assembly Line." Via http://www.automod.de/media/doc/Vedapudi.pdf [accessed June 8, 2007]

Versteegt, C. and A. Verbraeck 2002. The Extended Use of Simulation in Evaluating Real-Time Control Systems of Avgs and Automated Material Handling Systems. In *Proceedings of the 2002 Winter Simulation Conference*. 1659-1666. San Diego, CA.

Whorter, S. M., B. Baker and G. Malan 1997. Simulation System for Control Software Validation. In *1997 SCS Simulation Multiconference*, Atlanta.

AUTHOR BIOGRAPHIES

MICHAEL P. JOHNSTONE is a PhD. Candidate at Deakin University. His research focuses on the use of discrete event simulation to analyse complex networks. He received a BE (Honours) in Engineering at Deakin University. Michael has several years working in IT and as a simulation consultant. His email address is <mpjoh@deakin.edu.au>.

DOUGLAS C. CREIGHTON is a researcher in the School of Engineering and Technology at Deakin University. The industrial focus of his work has been made possible through a collaborative research program between Deakin University and Ford Motor Company, called FAST. His research interests are discrete event simulation, intelligent agent technology, simulation optimisation techniques, and the design and modelling of manufacturing systems. He received a BE (Honours) in Systems Engineering and a BSc in Physics from the Australian National University, where he attended as a National Undergraduate Scholar. Doug Creighton also has several years of engineering and computing experience, working with the Australian Department of Defence and in Federal Parliament House, and more recently as a simulation consultant. His e-mail address is <dcreight@deakin.edu.au>.

SAEID NAHAVANDI received BSc (Hons), MSc and a PhD in Automation and control from Durham University (UK). Professor Nahavandi holds the title of Alfred Deakin Professor, Chair of Engineering and is the leader for the Intelligent Systems research Lab. at Deakin University (Australia). He won the title of Young Engineer of the Year for his novel intelligent robotic end effector in 1996 and has published over 300 peer reviewed papers in various International Journals and Conference and is the recipient of six international awards in Engineering. His research interests include modeling of complex systems, simulation based optimization, robotics, haptics and augmented reality. Professor Nahavandi is the Associate Editor - IEEE Systems Journal, Editorial Consultant Board member – International Journal of Advanced Robotic Systems, Editor (South Pacific region) - International Journal of Intelligent Automation and Soft Computing, Editorial Board member - International Journal of Computational Intelligence. He is a Fellow of Engineers Australia

(FIEAust) and IET (FIET) and Senior Member of IEEE (SMIEEE).