Proceedings of the 2010 Winter Simulation Conference
B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.

LIVE MODERNIZATIONS OF AUTOMATED MATERIAL HANDLING SYSTEMS: BRIDGING THE GAP BETWEEN DESIGN AND STARTUP USING EMULATION

Nathan Koflanovich
Peter Hartman

Retrotech, Incorporated
610 Fishers Run
Victor, NY 14564, USA

ABSTRACT

Modernization projects on mission-critical automated material-handling systems are often performed while the system and the processes it supports remain operational. A successful live-system modernization depends on a rapid, predictable startup with no unmitigated risks. This is achieved through extensive testing prior to startup, which can not be performed while the system is in service. During a project, engineers overcame the live system's availability constraints by creating a software-based virtual system, emulating all control and feedback signals, operating speeds, constraints, and physical characteristics of the equipment. Twenty-one programmable controllers, a supervisory computer system, and four operator interface applications, all developed to operate the physical system, were connected to the model. All subsystems were brought online, and the system was run as a whole. All software, algorithm, control system, operator-interface, and factory-acceptance tests were performed in the model environment. The modernized system was brought online without disruption to any factory operations.

1 AUTOMATED MATERIAL HANDLING SYSTEMS IN INDUSTRIAL FACILITIES

During the past several decades, automation has been widely adopted as a key component to the material-handling operations within industrial facilities. In both manufacturing and distribution, automation is used to increase productivity, decrease building footprints, and reduce operating costs. Once installed, it can often become the foundation of a facility's operations, and productivity can not be achieved when it is inoperable.

1.1 The Live Modernization Project

As with all technology, the automation installed in industrial facilities has a finite viable lifespan. Over time, equipment becomes obsolete or difficult to support and maintain. Likewise, facilities sometimes outgrow their installed systems and need to expand or increase performance. When modernization is called for in an automated material-handling system that is supporting ongoing operations, the system-owner faces a challenging situation. Removing the system from service can be extremely costly. Outside storage may be required to replace internal capacity while work is performed. Temporary manual labor may be used to perform the functions of any disabled equipment. Attention must be given to perishable items residing in a disabled storage system. In cases where no reasonable alternate process exists to circumvent the automation, system downtime may even force the shutdown of upstream or downstream processes in the facility.

Out of this unappealing scenario was born the concept of the live-modernization. The concept is straightforward – perform all modernization work in such a way that the overall operation of the facility is

not disrupted. There may be localized disruptions within the system, but the overall productivity of the facility must remain intact. Production plants must not experience production stoppages, and distribution centers must not experience shipping or receiving outages.

1.2 Keys to Success

All modernization projects are challenging. The risk of unforeseen equipment problems, incorrect documentation, and misunderstood functionality within the existing system is much higher than with a greenfield installation. A live modernization of a mission-critical automated material-handling system requires that extensive planning be undertaken to understand not only the work to be performed, but the interaction of that work with the ongoing operation of the facility.

Quality must be guaranteed before going into production on the modernized system, as extended downtime can not be tolerated. To eliminate disruptions to other facility processes, equipment downtime must be minimal and isolated. Extensive, well coordinated functionality and performance testing, customer acceptance testing, and equipment testing are the keys to guaranteeing success.

In a live modernization, the time available on the real system for testing, debug, and startup is limited, because these efforts can not be performed while the system is in-service. This limitation creates a gap between the design and startup phases of a project that must be addressed to ensure success. This gap is bridged through modeling, simulation, and emulation.

2 THE ROLE OF SIMULATION AND EMULATION

To bridge the gaps between design, development, and startup in live modernizations of automated material-handling systems, a functional model of the entire system is created to serve as a test-bed. The scope of the modernization will determine whether the model will employ simulation, emulation, or a combination of the two. A system is generally divided into subsystems, and a choice between simulation and emulation is made for each of those subsystems depending on the depth of the modernization work to be performed.

2.1 Simulation

Simulation is the replication in the model of the functionality of the system (or a subsystem within that system) under study. Simulation does not require that the physical system be embodied in the model, although this is one possible approach. The key to successful simulation is that all external interface points to the system, both physical and logical, are implemented, and that the simulated system accepts stimuli and responds in exactly the same manner as the real system.

In a simple implementation, a simulator may be a standalone application developed to replicate a communication interface, and to provide additional interface capability to connect to other simulators and emulators where physical interactions take place between subsystems. This form of simulator pretends to perform the functions of the equipment being simulated through the use of logic, timing, data handling, and other processes, and provides feedback about its activity via the simulated interfaces. The processes within the simulator are abstractions of the real processes being simulated, and need not be identical to the real system, as long as the interfaces have identical functionality and performance when the simulator and the real system are viewed as black-boxes.

A more complex or detailed simulator may embody the physical system and all of its components, as well as the items to be handled, and contain the logic required to operate all components, devices, and processes within that physical system in the exact manner found in the real system. This type of simulation may be used to allow for more precise analysis of the timing, performance, and functionality of the system over a broad range of test scenarios.

2.2 Emulation

Emulation is the replication in the model of the functionality of individual devices within the system, down to the level of discrete control signals. Where simulation implements a system or subsystem with self-contained internal processes, emulation implements the discrete devices within that subsystem, but does not provide the control and processes that make it perform any system-level functionality. Emulation relies on an external control system to operate its devices. This allows the actual controllers and control software that will operate the real system to operate the emulated system.

In a simple implementation, an emulator may be a standalone application developed to replicate the discrete control signals coming from and going to all of the devices in the system, as well as the interactions between those devices as they are actuated, and to provide additional interface capability to connect to other simulators and emulators where physical interactions take place. This form of emulator pretends to respond to the actuation of discrete devices and provides feedback on any interdependent devices through the use of logic, timing, data handling, and other processes.

A more complex or detailed emulator may embody the physical system and all of its components in the exact manner found in the real system. This type of emulation may be used to allow for more precise analysis of the timing, performance, and functionality of the system over a broad range of test scenarios. A standalone emulator reacts to control signals in a predictable manner because it is preprogrammed to do so. An emulator that embodies the physical system reacts as the real system would in any operating scenario, and is not limited to preconceived operating patterns, providing a much more robust testing environment for unproven control software.

2.3 Selecting a Method

Simulation implements the subsystem as a whole, with all functionality included, while emulation implements the equipment within the subsystem, leaving functionality to an external control system.

When a system is to be modernized, there are generally certain subsystems that will not be modified in any way, but must still be integrated into the system, interacting with parallel subsystems as well as an overall control system. Simulation is an appropriate tool for modeling these subsystems, because no testing is required within the subsystem, but the interfaces and interactions with other subsystems, as well as the subsystem's role in the overall functionality of the system, must be tested. If the internal processes are already proven, and only the interaction with other processes and subsystems will be tested, simulation provides all of the required capability in the model.

When modernization efforts are performed on the internal workings of a subsystem, such as when a subsystem controller is replaced or its software is rewritten, emulation is a more appropriate choice. Emulation provides all of the features of simulation, but also allows the newly developed control software to operate the subsystem, allowing it to be debugged and tuned as though it were installed on the real subsystem.

3 CASE STUDY: AN INTEGRATED MODEL

Retrotech, Incorporated was contracted to modernize an automated material handling system in a food production facility. The system handled raw ingredients and supplied the facility's production lines. Production was completely dependant on the system, and no workaround existed to operate without it.

3.1 System Description

The system consisted of several subsystems that were tightly integrated and interdependent. First, a conveyor subsystem accepted raw materials in totes from a receiving dock, and conveyed them to a storage buffer. Automated cranes in the storage buffer carried the totes from pickup locations on the conveyor to locations in a storage rack. There were six cranes servicing six storage aisles, with a total capacity of 2,592 totes. When a production line needed an ingredient, a crane retrieved a tote from the rack and depo-

sited it on another conveyor. This conveyor presented the tote to a rail-guided vehicle subsystem, comprised of seven vehicles on an oval track. A vehicle would pick up the tote and carry it to one of three conveyor subsystems, servicing four production lines.

Two of these subsystems were dedicated to individual lines, while the third handled the remaining two lines. Once a tote was dropped off, it would be conveyed to an operator station where an operator would open and unpack the tote, and then release it to the production line. In the first two production line subsystems, the tote was conveyed to additional rail-guided vehicle subsystems, each comprised of a straight track with a single vehicle. The vehicle shuttled totes from the pickup station to any of eight hoppers that fed a production line. It also brought empty totes from the hoppers to another conveyor, which delivered them back to the rail-guided vehicle loop after an operator re-packed and released them.

On the third production conveyor subsystem, totes were conveyed directly to hoppers after being released by an operator. Once emptied, the operator would release them, and they would be conveyed back out to the rail-guided vehicle loop.

When a tote was released from any of the lines, a vehicle in the loop would pick it up and deliver it to an elevator that brought it to the ground level to be removed from the system if it was empty, or back to the storage buffer to be put away again if it still had material in it.

3.2 Modernization Scope

Retrotech was to provide a new Warehouse Control System to manage the overall workflow in the entire material handling system. In addition, Retrotech was to develop new conveyor control software, new control hardware and software to operate each of the cranes, new control hardware and software to operate each of the rail-guided vehicles, new control hardware and software to manage work within the three rail-guided vehicle subsystems and the storage buffer subsystem, operator interface applications to run the whole system, operator interface applications to manage and maintain the cranes, and an operator interface application to act as a backup in case one of the other operator interface systems failed.

This work was to be performed as a live modernization, and no production downtime could be tolerated. Brief testing periods were available every second weekend, but the system had to be fully operational by Sunday evening.

Adding to the complication was the fact that Retrotech had never undertaken control of a rail-guided vehicle loop, and the algorithms to manage traffic and direct work in that complicated subsystem needed to be designed from the ground up. Vehicles on a loop create a highly interdependent subsystem with complex internal interactions. Each vehicle's activity may influence or interfere with one or all of the other vehicles. When analyzing time-based performance of the subsystem, even something as fundamental as the transit time between two points on the loop becomes complex, because it is dependant on the precise scenario on the loop at any given time. Heavy traffic in front of a vehicle while approaching a curve will slow that vehicle down. A vehicle performing a pickup or deposit operation may block other vehicles until the transfer is complete. All of these interactions make it apparent that there are many ways to manage the work in the loop, and that the loop's overall performance is highly dependant on the intelligent design of the management algorithms. The design process leading to the loop's management strategy had to answer many questions. Is it better for vehicles to look for work, or for work to look for available vehicles? What criteria are used to match a vehicle to a task in the loop? Where should idle vehicles be parked to minimize interference and maximize performance? When should idle vehicles be moved out of the way of others, and how far should they move? Can work be assigned to an idle vehicle that is moving to get out of another vehicle's way? How should the loop respond if a vehicle overshoots a target position? If the overshoot is small, can vehicles move in reverse, or do they have to go forward around the loop again? How do exception conditions such as this effect the subsystem as a whole, and how does it recover? What conditions could lead to gridlock, and how should they be addressed? The answers to all of these questions determine the performance of the subsystem, but none of these questions have obvious answers. These intricacies called for extensive, iterative testing for functionality, reliability, and performance.

Because of the massive breadth of the modernization project, the new technology involved, the limited testing windows, and the critical nature of the system, emulation was used extensively, along with some simulation, to model the system completely before any installation operations took place on the real equipment.

3.3 Modeling Platform and Implementation

Emulate3D Controls Testing Edition software was chosen to implement all equipment in the model with the exception of the cranes. This application can be used to create a realistic 3D model of a material handling system, complete with physical properties such as friction and gravity. It provides an interface to allow for connections with external control systems, such as programmable logic controllers. Rather than simulating activity through time progression alone, this software calculates physical interactions between entities in the model, and along with gravity, these interactions drive all movement. If an entity is resting on a running conveyor, the friction between the conveyor and the entity will cause it to accelerate and move. When an entity comes in contact with another entity, the force vectors that result from the collision will be applied to the entities. Figure 1 illustrates Emulate3D's physics display mode, showing two totes colliding on a conveyor.

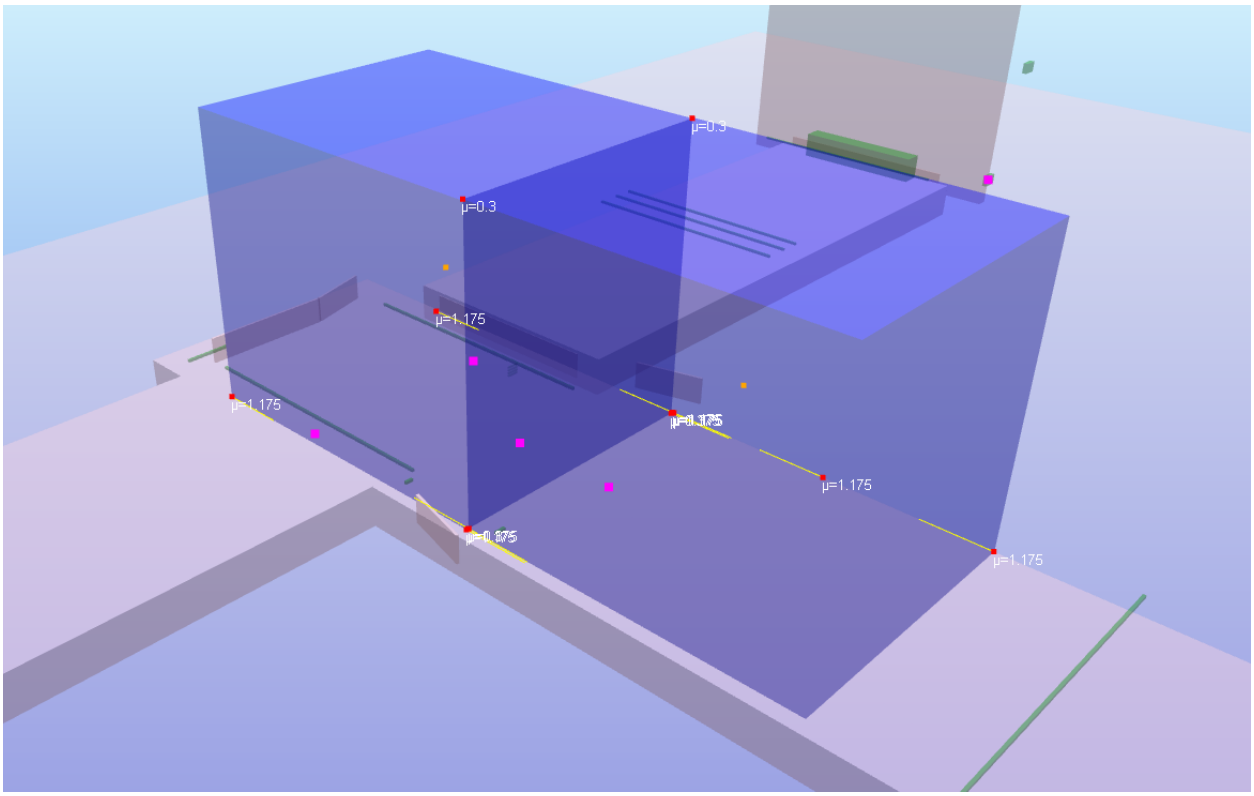

Figure 1: Emulate3D physics display of two totes colliding, showing interaction points

A model was created in Emulate3D containing all of the conveyor, elevators, rail-guided vehicle subsystems, operator interaction areas, and production lines. Totes were implemented with physical properties that matched the totes in the real system. Realistic speeds, accelerations, and manual intervention times were built into all of the model equipment. An image of the Emulate3D model can be found in Figure 2 below.

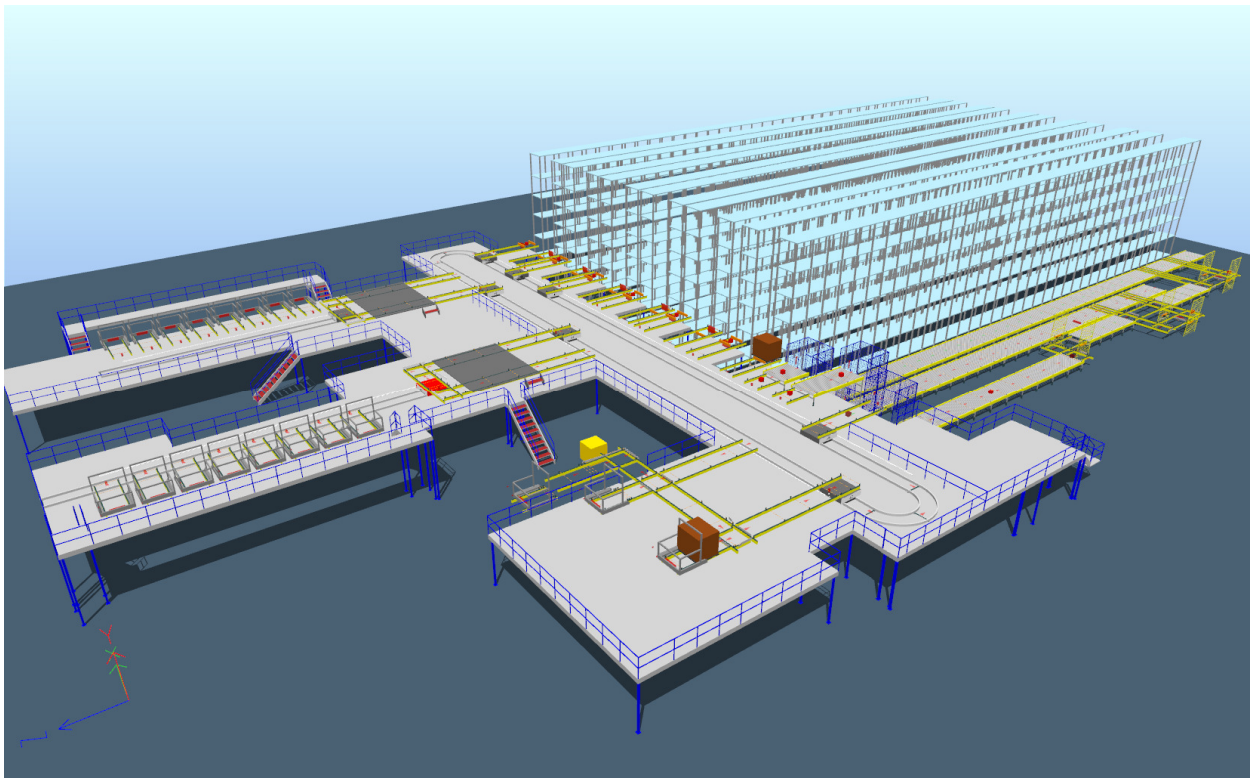

Figure 2: Emulate3D model of the system to be modernized

The model devices were designed to operate exactly as the real devices, with the same control signals, sensors, and actuators. Model signals were designed to match the electrical signals that were to be connected to the equipment controllers in the real system, and those signals were connected together via a communication interface.

Fifteen programmable logic controllers were connected to the model as though they were wired to the real equipment. There were five controllers to operate the conveyor, nine to operate the rail-guided vehicles, and one to manage the traffic in those subsystems.

The cranes were implemented using standalone emulators, and linked to the Emulate3D model to provide integrated functionality. The crane emulators were configured to match the physical properties of the cranes, and to mimic the real devices on the crane, with the same control signals, sensors, and actuators. Six programmable logic controllers were loaded with the actual crane control software alongside the emulator software, and the emulator signals were connected to the crane program in place of the electrical signal points, creating self-contained, model crane subsystems.

Operator interface terminals and applications, as well as the warehouse control system, were connected to these twenty-one programmable controllers in exactly the same way as they would be in the real system. The programmable controllers could not distinguish between the model and the real system. The operator interface applications and the warehouse control system were connected to the actual controllers, so no modifications were required, and no distinction could be made between the model and the real system.

A simulator was used to create and identify totes in the model to be inducted into the system. These totes became the inventory required to support the emulated production operations.

3.4 Testing

The model facilitated testing at several levels. The first and most fundamental level can be described as controls testing, in which the software that resides in the programmable logic controllers on each piece of

equipment is tested and debugged, and the equipment is run as a standalone unit. The next level is subsystem testing, in which multiple pieces of equipment interact to provide subsystem functionality. Next, during system testing, all of the subsystems work together to provide the functionality of the system as a whole. Finally, factory acceptance testing allows the customer to run the model system for approval and validation against the design specifications and requirements.

3.4.1 Controls Testing

Controls testing in the model can be seen as an abstraction of the initial machine startup process that is performed on the real equipment after it is modernized. The programmable logic controller, loaded with its control software, is connected to the model instead of the electrical systems on the real equipment, but otherwise, it is functionally equivalent.

Controls testing was performed with all of the programmable logic controllers that were to operate the conveyors and rail-guided vehicles. As each controller was connected to the model, all signals that were connected to emulated devices were checked for proper operation. All modes of operation were then tested using the same procedures that would be employed on the real equipment. Basic manual control was tested and debugged, followed by more advanced manual control functions. Once the device under test was fully operable in manual mode, automatic mode testing was performed.

The testing platform embodied by the model provided the developer with opportunities to perform more testing in less time, out of sight of the system owners. While forklifts, drivers, and manual labor would be needed to move totes in the real environment (much of which is not forklift-accessible), model totes could be created, removed, or relocated with the click of a mouse. This reduced the time required to set up and perform tests, facilitating more testing iterations than would have been feasible with actual totes of raw materials. Anomalous events, such as jams, collisions, unpredictable operation, and incorrect processing could happen in the model without any damage to the real equipment or the products being handled, and without any embarrassing visibility by the customer. A defect in the control system that allowed a tote to tip over or fall off of the conveyor did not result in any messy cleanup. The developer could simply delete the tote, correct the defect in the control program, and create a new tote for the next testing iteration. The real system would not be so forgiving.

The physics-based approach implemented in the model was designed to closely approximate the real system's responses to physical activity. A tote that was improperly conveyed could twist and jam, or even fall off of the conveyor. This allowed for detailed debugging of the control software that might not have been possible in a time-based simulator, where unforeseen interactions between entities within the simulation may not be programmed to have any effect within the system. It also provided a platform for highly accurate testing of the rail-guided vehicle software, which was completely new and came with many unforeseen challenges. Scenarios such as positioning-system deviation, overshoot, position misalignment, hunting methods, and position learning could all be tested in addition to the normal manual and automatic-mode operations.

Time spent performing controls testing against the emulator saved time and money on a grand scale. The testing was unavoidable, and had to be performed either on the model or in the real system. Performing it in the model allowed for quicker, easier setup of test scenarios, quicker recovery between tests, more testing iterations, no-risk testing of scenarios that could cause product or equipment damage, and testing of scenarios that would have been difficult or impossible to replicate on the real equipment. In addition, Retrotech was able to realize travel cost savings, a more rapid equipment startup on-site, and a boost in customer confidence due to the fact that the defects in the control programs were eradicated before ever setting foot in the customer's facility.

3.4.2 Subsystem Testing

Once the individual controllers were debugged and fully operational, subsystem testing was performed in the model. This allowed the interfaces and interactions between equipment within each subsystem to be tested and debugged. In this modernization, the subsystem that needed the most in-depth testing was the rail-guided vehicle loop.

The loop was brought online with all seven vehicles, and totes in need of transport were presented to it. The programmable logic controller responsible for traffic management was connected to the equipment to perform routing functions, commanding the individual vehicles to perform actions based on the activity in the loop as a subsystem. By having a complete and accurate model of this subsystem, Retrotech was able to test the functionality of the traffic manager, while further testing automatic operation of the rail-guided vehicles, without restriction or risk. Bugs that caused gridlocks, inefficient movement, delays, and improper operation were found and corrected before ever connecting to the customer's real equipment.

Subsystem testing validated management and communication within each of the subsystems, and set the stage for connection to the supervisory controller, the Warehouse Control System.

3.4.3 System Testing

With all equipment controllers and subsystems operational, the warehouse control system was connected to run the system as a whole. Raw material receipts were simulated, and totes were conveyed into the model system. The Warehouse Control System was able to interact with the equipment and make routing decisions, directing the activities within each subsystem to provide the required overall functionality. Raw materials were stored in the buffer, and retrieved to support the production activities. Totes were routed along conveyor, through the rail-guided vehicle subsystems, into and out of the production hoppers, and out of the system. The complete automated facility, with all processes, functions, and activities, was operational in the model, allowing for extensive testing of the Warehouse Control System and its integration with all of the other subsystems.

Testing on the model allowed Retrotech to ensure that the system was functional and in alignment with all specifications before the customer ever saw it. Ensuring quality prior to installation resulted in cost savings during installation, but it also instilled confidence in the customer. When the modernization was finally implemented on the real equipment, the customer saw smooth installation and integration efforts, followed by all subsystems coming online fully functional.

3.4.4 Factory-Acceptance Testing

With the realism and accuracy of the integrated model, the customer was able to perform all factory-acceptance testing in the emulated environment. This testing is conducted before equipment is delivered to the customer, and as such it is often only performed at the machine level. In this project, Retrotech was, in certain subsystems, not delivering any equipment at all, only new software to run existing equipment. A traditional factory acceptance test would not have proven any of the functionality of the integrated system, only bits and pieces of equipment functionality where new equipment was provided.

With the integrated model, the entire system, with all subsystems, including existing subsystems that were not part of the modernization effort, was brought online and made available for the customer to use and test. All operator interfaces were functional and could be validated for functionality and operability. All system functions were tested and proven, and the customer was left with a high degree of comfort that the integrated system, with so many newly developed pieces, was cohesive, functional, and in alignment with their requirements. The platform provided testing capability beyond what they expected, and they were able to sign off to allow the installation to proceed without hesitation.

3.5 Customer Training

Using the integrated model, sessions were held to train the system's operators before any installation activities began. The operators were able to run the system and try out all of its features, functions, and processes in the virtual environment. This accelerated the learning curve, shifting the operators' period of inefficiency away from the real system and into the model instead. Given the system's critical role in the overall operation of the facility, this mitigated the risk of lost productivity when the system went live. It also mitigated the risk of downtime due to mistakes made by novice operators.

4 CONCLUSIONS AND LEARNINGS

4.1 Physics-Based Modeling

It is worth reiterating the advantages of using a modeling platform that solves entity interactions as physical systems. Many simulators compromise realism for reasons of computational capacity or design simplicity. Simulators are often not built with the flexibility to handle unforeseen scenarios or unintended interactions. Any compromises in the simulated environment add uncertainty to the efficacy of the simulator as a testing and validation tool.

In a time-progression simulator, a tote on a conveyor generally accelerates and decelerates at the same rate as the conveyor. It may start and stop without concern for acceleration and deceleration. Using physics to drive motion, a tote may slip on a conveyor if accelerated too rapidly, causing it to skew, deviate from an intended position, jam, or perform any number of unintended actions. These possibilities are handled by the physics-based simulator, allowing a developer to see the consequences of interactions within the system before ever touching any real equipment.

During testing of the rail-guided vehicle loop, a vehicle went into an error state, reporting that its bumper had been tripped. The developers looked at the model, and found a tote on its side laying in the track. The tote had been inadvertently conveyed into the loop when no vehicle was present to pick it up, and a vehicle has proceeded to collide with it and go into an error state.

When totes were conveyed through right-angle transfers, timing problems in the control software could cause the tote to miss the transfer point slightly, and then twist, jam, or even be turned sideways as they were conveyed. A sideways tote was too wide to fit on a rail-guided vehicle. If a vehicle picked up a sideways or skewed tote, it would go into error indicating an overhanging object.

These testing scenarios may seem to be nuisances in simulation, but they have incredible value. They allow a developer to test error handling with unprecedented degrees of control, and also to test error recovery. The ease with which one of these scenarios can be created, whether intentionally or not, allows for testing that would be impractical with real equipment.

The physics-based model helps to separate the developer from the inner workings of the simulator, preventing preconceived ideas about how the equipment will perform and operate from influencing the capabilities of the model. The model is built on constants, facts, and physical properties, and when the model equipment is operated, anything can happen.

4.2 Emulation Processing Requirements

Simulation technology is constantly evolving, facilitated by inexpensive, readily available computing power. Processing physical calculations in a model is inherently more resource-intensive than processing by time-progression alone, and special care must be taken to avoid excessive complication in model objects, while retaining enough of the physical system to maintain realism. Oversimplification can lead to limitations similar to those present in a time-progression simulator.

An emulation platform must be able to run a model consistently without a speed multiplier. If model time is too fast or too slow, the timing by which the external controllers perform their functions will not be synchronized with activity in the model.

At this time, the authors have not determined any way to gauge computing requirements for a given emulation effort. It has been experiential, and benchmarks, as well as techniques, have been captured on every new implementation. When a computational capacity threshold is exceeded, a model must be analyzed to ensure that there is no wasted computational capacity, and optimized for performance, or more powerful hardware must be obtained. It is expected that further analysis could provide a more scientific formula to determine computational requirements for a model based on the equipment that is to be simulated or emulated.

4.3 Emulator Connectivity

In a large integrated model, emulation can require that a massive quantity of data points be exchanged with external devices. In the model described here, over 1,300 signals were exchanged with fifteen controllers in real-time. There are two fundamental approaches to handling this data exchange – polled or change-of-state.

This model used change-of-state data exchange. The model was connected to a data server that in turn communicated with the programmable logic controllers. Any time a signal changed in the model, it was communicated to the controller via the data server. Likewise, the data server monitored the programmable logic controllers for changing signals, and relayed them to the model. If there is limited activity in the model, this leads to the fastest possible data throughput because signals are exchanged on-demand any time they change, eliminating lag in their propagation.

There are, however, limitations in bandwidth on the communication channels between the data server and the programmable logic controller. While some data servers are more efficient than others, all have limitations. With hundreds of signals changing per second, the number of data packets that must be exchanged can grow unmanageable, causing delays in signal propagation. Hence, the performance of change-of-state data exchange is very high up to a threshold, and falls off sharply when the threshold is exceeded. In this model with 1,300 data points, the data server was able to perform adequately most of the time, but some signal loss was experienced when activity levels were high or network delays were present.

In a polled data exchange method, a routine, periodic data transfer is configured to exchange all or part of the data image between the model and the external devices. In this method, the exchange interval may be throttled to some predetermined rate, or allowed to communicate at the maximum natural free-running rate. Because a larger image is exchanged than with change-of-state, the response time can not generally match change-of-state data exchange when activity is low. When activity levels are high, the performance can vastly exceed change-of-state, because the data exchange is not linked to activity, and is routine and predictable over all activity levels. Exchanging the entire data image in a single transmission is much more efficient than exchanging hundreds of individual signals.

In selecting a method to be used to connect an emulator to an external controller, careful consideration must be given to the number of data points to be exchanged, the number of controllers, the level of activity in the system, and any devices that produce rapidly changing data.

The maximum data rates of the data server and the external controller must be compared against the expected data volume in order to choose a data exchange method and, if polled exchange is selected, a polling interval.

4.4 Diminishing Returns

With an exquisitely detailed model available for testing at any time, developers must remain focused on a test plan, or they may spend more time testing than intended. Testing in a model is convenient and inexpensive when compared to testing in a real system. Many challenges are avoided, such as travel, customer coordination, system downtime, manual labor for testing support, and the risk of equipment or product damage. Because of this, care must be taken to follow a test plan with defined tasks and a defined sche-

dule, or testing can go on endlessly. There will always be one more thing that a developer would like to see, but there is a point of diminishing return that must be managed.

4.5 Results

When the customer's project team first began to investigate the feasibility of performing this modernization without production stoppages, they were concerned that it may not be an achievable goal. because of the extensive use of emulation in its development, this project was implemented successfully and exceeded all plant expectations. All of the testing and training that was performed in the emulated environment led to a smooth transition from the old system to the new system. The changeover was performed over a weekend, and no unplanned downtime or production stoppages occurred. Plant personnel who were not directly involved commented that they did not even know anything had happened, which is a great indicator of success given the broad scope of work and all-encompassing changes in the material handling system.

Every hour spent testing in the emulated environment eliminated more than an hour of startup or live testing time on-site. On-site testing requires a minimum of two people: an engineer to perform testing of the equipment and software, and a support person to assist by moving totes around, actuating sensors, verifying device actuation, and observing activities in the system. Testing in the model environment was performed by the engineer with no support personnel, effecting a labor savings of at least 50%. Likewise, travel and living expenses that would have been incurred during on-site testing were eliminated when testing in the model environment.

Emulation was a driving factor in the success of this project. It should be considered when undertaking any industrial control project, and is especially valuable when strict time constraints or a high degree of complexity are present.

AUTHOR BIOGRAPHIES

NATHAN KOFLANOVICH is a Controls Engineer at Retrotech, Incorporated, an engineering firm that specializes in the design, installation, modernization, and support of automated material handling systems. He has worked in several roles for the company over the past 14 years, and is currently focused on systems engineering, simulation, and emulation. His e-mail address is <nkofl@retrotech.com>.

PETER HARTMAN is the President of Retrotech, Incorporated. He has a long history in the material handling industry, spanning three decades. He is interested in the evolution of simulation technology and its applicability to all phases of automated material-handling system projects, from pre-sales to startup. His e-mail address is <phart@retrotech.com>.