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.

THE SIMULATION POWER OF AUTOMOD

Todd LeBaron
Craig Jacobsen

Applied Materials
Manufacturing Automation Services
5245 Yeager Road, Suite 100
Salt Lake City, Utah 84116, U.S.A.

ABSTRACT

Decision making in industry continues to become more complicated. Customers are more demanding, competition is more fierce, and costs for labor and raw materials continue to rise. Managers need state-of-the-art tools to help in planning, design, and operations of their facilities. Simulation provides a virtual factory where ideas can be tested and performance improved. The AutoMod product suite from Applied Materials has been used on thousands of projects to help engineers and managers make the best decisions possible. AutoMod supports hierarchical model construction. This architecture allows users to reuse model objects in other models, decreasing the time required to build a model. In addition, recent enhancements to Automod's material handling template systems have increased modeling accuracy and ease-of-use. These latest advances have helped make AutoMod one of the most widely used simulation software packages.

1 INTRODUCTION

AutoMod is unique in the world of simulation tools. It allows users to construct models of any size and complexity that can be used for planning and design, for day-to-day operations analysis, and for controls development and testing. AutoMod combines concurrent 3-D graphics, shown in Figure 1, with a comprehensive set of templates and objects for modeling many different applications. Because of AutoMod's capacity for detail, it has become the tool of choice for serious simulation practitioners who need to maximize the return on their simulation investment.

The flexibility of AutoMod is noteworthy. The architecture allows for the reuse of model objects, thus reducing the time it takes to build a model. AutoMod can also be linked to other software through ActiveX to help create simulation tools tailored for plant managers, area supervisors, or operators on the floor. Simulation technology is being applied to more than just planning and design, and AutoMod meets that challenge with functionality to support a wide range of applications.

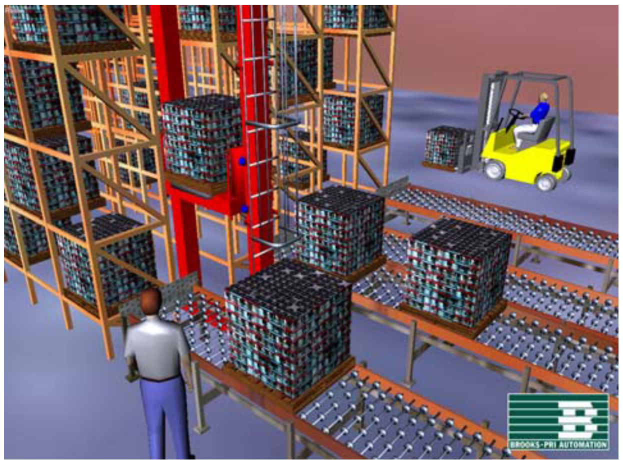

Figure 1: AutoMod's concurrent 3-D graphics

The AutoMod product suite includes:

- **AutoMod.** Model builder and execution environment.
- **AutoStat.** Statistical analysis and optimization tool.
- **AutoView.** Animation tool with AVI support. Produces dynamic walk-thoughs.
- **Model Communications Module (MCM).** Protocols for linking to third-party software, including OLE for Process Control (OPC), for control systems testing.

AutoMod contains templates for several types of material movement systems. AutoMod's movement systems allow users to model both manual material movement and movement by automated equipment. Movement systems include:

- Path movers (path-based vehicle systems, such as lift trucks, AGVs, and human movers)
- Conveyors (including belt and roller types)
- Automated storage and retrieval systems (ASRS)
- Robots (user-defined kinematics)
- Bridge cranes
- Power and free chain conveyors

To define movement systems, you simply define the system elements, such as paths and stations, and then fill in operating parameters like velocity and acceleration. AutoMod then automatically creates the corresponding control logic for the devices. Statistical performance reports and 3-D animation are created automatically, providing a realistic and statistically accurate view of a facility. Model animation can be viewed from any angle or perspective in real time, providing visualization capabilities unmatched in other simulation tools.

AutoMod's key strengths are as follows:

- Easy-to-configure material movement systems
- The power and flexibility to model complex systems accurately
- Unlimited model size
- A high-performance simulation engine
- Best-in-class statistical analysis features
- A graphic environment for geometry creation
- Hierarchical model construction
- True-to-scale graphical import from CAD tools

The animation provided in AutoMod models is concurrent. This means the graphics run real-time with the simulation model. The model execution environment is very interactive; you can stop and start the simulation, or you can run without animation in an accelerated time scale. You can also select objects from the animation screen and view detailed statistics about these objects and the system being modeled. Statistics can be viewed at any time during a simulation run. These model execution features make it easier to verify and validate models of complex systems. These features also provide a forum for communication about system performance among project team members and management.

2 AUTOMOD INTERFACE

An AutoMod model consists of one or more systems organized in one or more sub-models. A system can be either a process system, in which flow and control logic are defined, or a material movement system. Each model contains at least one process system and may contain any number of movement systems. Automod supports hierarchical model construction so users can integrate pre-tested sub-models into new models, reducing the time it takes to build a model. Processes contain logic to control the flow of manufacturing materials and to issue control messages, thereby meeting the exact needs of your operation. Model entities (loads) compete for resources or wait for user-specified times. Loads can move among processes either directly or by using movement systems.

All inter-arrival and event times can be either represented by deterministic values or derived randomly from one of many statistical distributions. AutoMod's interface is Windows-based, using pop-up and pull-down menus, dialog boxes, selection lists, and an editor for developing process logic.

3 AUTOMOD'S WORLDVIEW

Any number of movement systems can be defined in an AutoMod model, and process systems connect the movement systems to the logical flow of products. In a process system, loads (products, parts, and so forth) move among processes (locations) and compete for resources (equipment, operators, and queues). The load is the active entity, executing action statements that are connected to the processes. Typical action statements give users the ability to:

- Claim and use machines and operators
- Move into queues or onto vehicles
- Create new loads
- Change load types (graphical representation)
- Wait on user-defined delay lists
- Increment or decrement counters or variables
- Set variable values
- Read from data files, Excel workbooks, or databases
- Send to other processes
- Make conditional tests
- Call user defined subroutines and functions
- Perform mathematical calculations

Figure 2 shows the process system palette for AutoMod. The palette is organized in a top-down manner with the most commonly defined elements at the top.

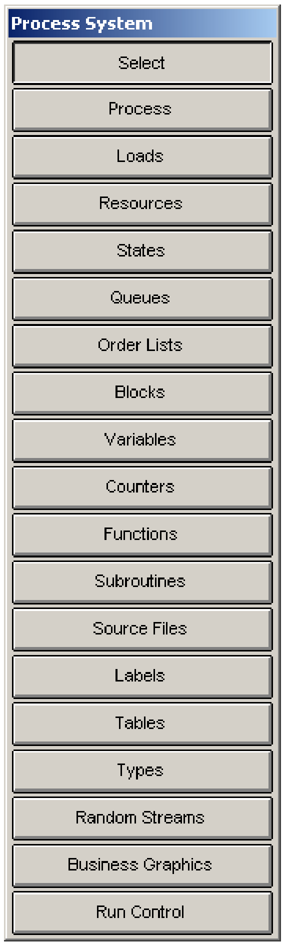

Figure 2: AutoMod's process system palette

In short, Automod provides the framework to accurately model any real-world system—including distribution centers, warehouses, assembly lines, airports, semiconductor fabs, special equipment, and electronics assembly plants.

3.1 Processes

The process system is the backbone of an AutoMod model, providing the general-purpose simulation features required for modeling a wide range of real-world problems. While material movement is important, it is not always critical in manufacturing. Simply moving a product around a manufacturing facility adds no value to the product. Machines and people perform operations that add value. The AutoMod process system is where these operations and control logic are defined.

AutoMod's process system uses a simulation language based on action statements. Action statements combine the power of a structured language with English-like, manufacturing-oriented syntax. AutoMod models are not limited in any way, so model logic can be of any size and complexity. The power and flexibility of the AutoMod language makes it easy to model almost any real-world situation.

AutoMod processes are where actions are performed or decisions are made. For example, an inspection operation could be represented as a process in AutoMod. From the final assembly process, parts would travel to the inspection process, where an inspector (resource) would inspect the parts. Processes can have a physical location, but this is not a requirement.

3.2 Loads

Loads are the active entities in AutoMod. They can be created in many ways, with either deterministic or probabilistic generation. The rate at which loads arrive at a process can be read from an external data file or attached to a statistical distribution. AutoMod has predefined random distributions that can be used to fit most real-world random events. Loads are given types, such as `RedCar` or `PartA`, and they can have attributes such as color, stock keeping unit (SKU), priority, and time in the system. Attributes can be accessed and modified through AutoMod's action statements.

Loads have 3-D shapes and dimensions, like most other entities in AutoMod. Load graphics can be changed on the fly with actions like:

```
set load type to RedCar
scale boxA by x 2 y 2 z 2
/*multiplies each dimension by 2*/
set this load color to blue
```

3.3 Resources

Resources represent machines, operators, fixtures, containers, and any other finite-capacity objects. There are two default categories for a resource's state: working (busy or idle) and available (up or down). During the animation, state colors indicate the status of each resource. Statistics are automatically collected for every resource in a model.

Loads use resources for specified processing times. These times can be deterministic or probabilistic, using AutoMod's built-in statistical distributions to simulate randomness. Processing times may vary by part, or they may apply to all parts using a particular resource. Resources can also be preempted if a high-priority load needs immediate attention.

Resources can have downtimes, which are defined by MTTF (mean time to fail) and MTTR (mean time to repair). These times can use AutoMod's statistical distributions to represent random failures. Though MTTF is by default based on simulation time, it can also be based on parts processed or machine running time.

Resource cycles can be created and attached to specific resources to indicate when or how often events such as random failures or preventive maintenance occur, or to model shift schedules.

3.4 States

The user can define states other than the default states. These states might be used to represent conditions such as blocked, starved, preventive maintenance, offline, and so forth. The state of a resource can then be changed using AutoMod actions so that statistics can be tracked for each state. In addition, state monitors can be defined and used to track states for entities other than resources, such as vehicles, conveyors, or particular areas of a facility.

3.5 Queues and Order Lists

Queues in AutoMod are both graphical and statistical elements. Queues have a user-defined capacity that can be set at build time, changed through code, or changed through the interface during the model run. When a queue reaches its capacity, the next load trying to enter that queue must wait until there is space available. Queue contents can be shown dynamically in the animation, and loads can be stacked in any direction in the queue. Containers can also be defined within a queue shape, for load stacking at specific locations. For example, Figure 3 illustrates a wafer cassette (modeled using a queue), that can hold up to 25 wafers (loads). The container defines a specific location within the queue for each load.

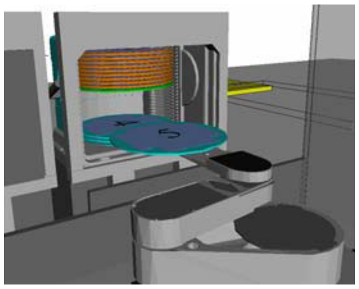

Figure 3: Queue used to represent a wafer cassette

Loads in a queue or process can be sorted, and they can by delayed until they are explicitly ordered to leave. To determine which action should take place next, the loads place themselves on order lists. An order list is a logical element that provides a way to sort loads that have been delayed. To remove a load from an order list, another load executes an `order` action. Loads can be ordered to move to another process or order list, or to simply continue where they left off in their processing. Order lists can be sorted by load priority or other load attributes, in ascending or descending order. Order lists are a very efficient and productive tool for tackling scheduling and sorting applications.

3.6 Blocks

Blocks control the number of entities occupying a physical space, making them very useful for controlling path mover vehicles. Blocks may have any capacity from one to infinity and may be set by default or modified during the model run. Both path mover vehicles (lift trucks, AGVs, electrified monorails, and so forth.) and loads increment blocks automatically when moving through the physical space defined by the block. Loads can also claim blocks in process logic, as directed by the user. Blocks can have any shape, including combinations of the AutoMod-supported shapes (discussed in Section 5).

3.7 Variables and Counters

Data values may be stored in an AutoMod model using variables. Variable values are changed using the `set`, `inc`, or `dec` actions as follows:

```
set Setup_time to 123.456
inc Vnuminsys by 1   /*Increments by 1*/
dec Vnuminsys by 1   /*Decrements by 1*/
```

Variables can be used in calculations or can be compared to other variables. In addition to integer, real, and string types, variables can also be used to store references to other process system entities, such as processes, queues, resources, order lists, counters, loads, and locations. Storing these references gives the user more power and flexibility to expand and extend models to match modifications of the actual system. AutoMod also supports arrayed entities, making it easier to model real-world systems that have some dimensionality (a warehouse with similar operations at multiple dock doors, for example).

Counters are similar to variables, but they must have positive integer values. Counters can have a maximum capacity, making them useful for traffic control. When a load tries to increment a counter that is at its capacity, the load is delayed until another load decrements the counter. Statistics for counters are collected automatically.

3.8 Functions, Subroutines, and Source Files

Functions and subroutines help you create modular models. This allows the models to be extended more easily. You can define your own functions in the AutoMod or C languages, and these functions can be called from anywhere in the AutoMod model. Subroutines help eliminate duplicate statements.

Source files contain the logic for the model. Any number of source files may be defined, and they are processed differently depending on their file extension. For example, .m files contain AutoMod logic and will be checked by AutoMod for correctness when edited. Files with a .c extension contain C language and will be compiled with the model. Users may also define other source files to contain input data or documentation for the model.

3.9 Labels

To make a model more understandable, AutoMod lets you place text labels anywhere in the model's physical space. Labels can be static, or they can change during the simulation run. Labels can rotate when the animation view changes or can be attached to a fixed position on the screen. In addition, at runtime you can set the label color, attach it to a queue or other entity, or scale it using AutoMod code.

3.10 Tables

Tables allow you to collect and classify statistics on any model entity. Tables automatically indicate averages and standard deviations, as well as maximum and minimum values. You can define the number of table "bins," or categories, and use the `tabulate` action to indicate what values should fill which bins.

3.11 Types and Random Streams

With types you can create lists of entities such as resources or stations, and then make complex decisions based on the lists. Default types include integer, real, resource, location, and so forth. You can also define your own types. For example, you could create a variable of type `ResourceList`, and then use that variable to facilitate complex decision-making.

Random streams may be defined to give each element of randomness independence from other elements. The number of different random streams used by AutoMod models is without limit. AutoMod uses the Combined Multiple Recursive Generator (L'Ecuyer and Touzin 2000).

3.12 Run Control

AutoMod's run control lets you define a model's warm-up and steady-state periods by resetting time-persistent statistics. Reports can be printed for any run control period, or "snap." In addition, business graph output can be automatically generated. Post-processed animation periods used by AutoView, an animation extension (discussed in Section 4), are also defined in the run control. Run control also provides an entity tracing capability that gives you an event-by-event account of the model run. This information helps verify and validate a model.

3.13 Business Graphics

AutoMod makes it easy to define and update graphs in real time. Graph types include bar charts, pie charts, and timelines. Figure 3 shows a typical timeline business graph. Any model entity can be attached to a graph, so that you could illustrate transporter vehicle velocity, the number of loads on a conveyor section, the average utilization of a machine, and so forth. Graphs can be printed or plotted to a variety of supported output devices. Graph displays can be controlled using the AutoMod language.

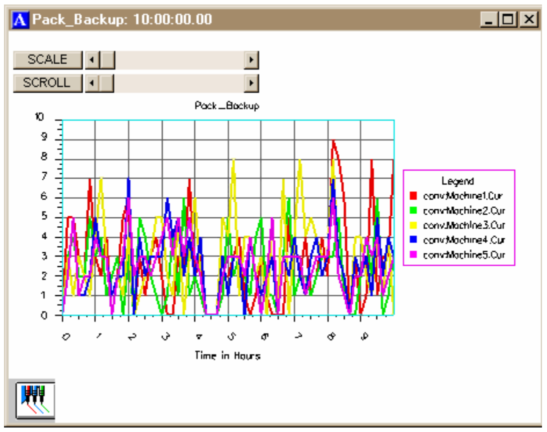

Figure 4: AutoMod timeline business graph

4 AUTOVIEW

AutoMod's post-processed [post-processor?] animation extension lets you save and view the animation records created by running a model. With the AutoView product you can pre-define all views and time periods in the model animation, and then play them back to generate presentation-quality animation files.

AutoView has single-frame capture capability to AVI- and MPEG-format files. This allows you to create animations that can be shared with others without the need for any additional software. Also, the single-frame capability helps smooth the animation on larger models where hardware constraints may affect animation performance.

5 GRAPHICS IN 3-D

Both dynamic and static objects can be displayed during model execution. Figure 1 and Figure 5 show screen shots of a typical AutoMod model during a model run. Dynamic objects represent loads, vehicles, resources, queues, and statistics. A static layout represents background graphics like columns, aisle markings, and walls. Labels can identify specific areas in the facility.

There are several ways to create a layout of the system to be modeled. AutoMod comes with a three-dimensional graphics editor for constructing objects from standard graphics primitives. Graphic Primitives include Cone, Box, Hemisphere, Trapezoid, Frustum, Cylinder, Arc, Vector (list), Set, Text, and Triad. Primitives can be selected, combined, placed, and scaled to create any static [or dynamic?] entity in the facility.

AutoMod supports the true-to-scale import of CAD format files under the IGES standard. Virtual Reality Modeling Language (VRML) and Open Inventor graphic format files can be imported directly. In addition, Automod supports DXF (.dxf), Stereo Lithography (.stl), trueSpace (.coa, .cob), Wavefront (.obj), 3D Studio (.3ds), and Lightwave (.lwo) formats.

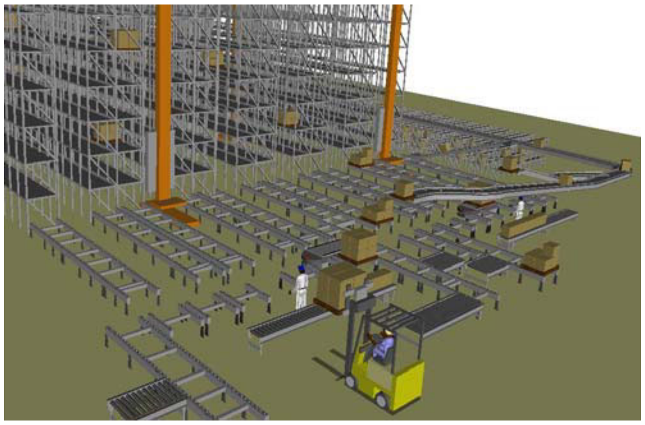
Figure 5: AutoMod’s 3-D graphics

6 RUNTIME ENVIRONMENT

In keeping with AutoMod’s interactive features, you have complete control of the model in the runtime environment. You can view the model with animation on, or run with animation turned off. AutoMod uses concurrent animation; the simulation progresses as the animation is being updated. With animation off, the simulation does not render the animation, but it performs all simulation calculations. The user can pause the simulation at any instant to view statistics in pop-up windows, to take resources down, to reset variable values for testing, to set break points or alarms, or to control the view of the animation without constraint.

6.1 User Interaction

AutoMod provides a comprehensive, flexible, and easy-to-use method of interacting with a model during model execution. If the simulation project is in the experimentation phase (where only parameter changes are made, and the model needs to be re-run several times), the AutoStat tool lets you run in batch mode without animation—automatically changing parameters, recording results, and generating reports and graphs.

6.2 Debugging, Reports, and Statistics

AutoMod also provides advanced debugging and trace facilities. A model can be single-stepped at any time during the animation. Also, with breakpoints and alarms, you can suspend the simulation when a certain event occurs or when a specific clock time is reached.

In addition, AutoMod provides comprehensive reports. The reports can be displayed on request at any time during the simulation. Printed versions of the reports can also be specified during model development. You can choose standard, full, or no report for any AutoMod entity.

AutoMod automatically generates statistics and reports for most entity types, including:

- Movement systems
- Processes
- Queues
- Resources
- Order lists

Vehicle states are tracked during the entire model run, and reports are generated automatically. Reports can be sorted alphabetically or numerically for easier analysis. You can also develop and generate custom reports from within process procedures.

7 AUTOSTAT

AutoStat is the statistical analysis tool in the AutoMod product family. AutoStat does extensive output analysis on an AutoMod model, including:

- Confidence intervals
- Warm-up determination
- Sensitivity analysis
- Factor-response analysis
- Design of experiments
- Optimization using evolution strategies.

Figure 5 shows an example graph from AutoStat.

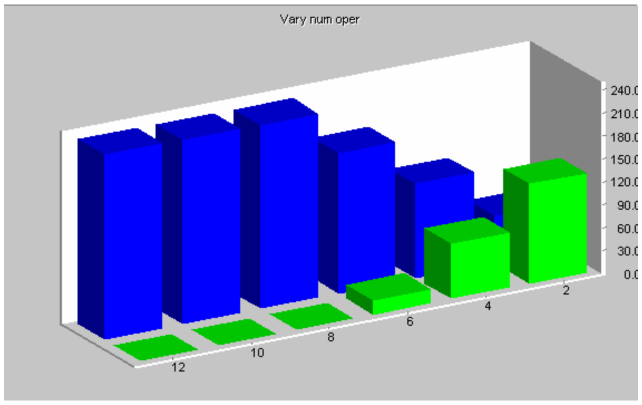

Figure 5: Example graph from AutoStat

AutoStat supports running scenarios across a network of machines. Support for multiple machines and CPUs lets you experiment with different simulation scenarios on multiple machines and compile the results to one machine, greatly reducing the time required for experimentation and analysis.

The evolution strategies algorithm used by AutoStat is well suited to finding the optimum solution without getting "trapped" at a local optimal value. Figure 6 shows the output from an optimization run. The X axis represents the number of generations required to find the optimum, and the Y axis gives the values for the user-defined fitness function. The fitness function can be defined as any combination of model inputs and outputs, with user-defined weighting factors applied to each value.

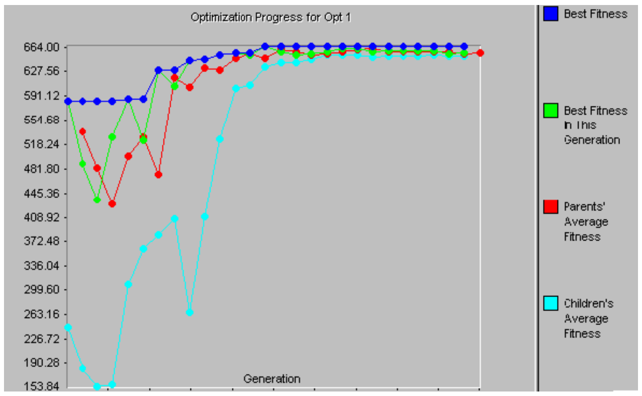

Figure 6: Optimization run graph

8 MODEL COMMUNICATION MODULE

The Model Communications Module (MCM) is an extension to AutoMod that allows simulation models to communicate with other programs either locally or on a network of machines. Some of the applications of MCM include communications between:

- Two or more AutoMod models
- Control systems and AutoMod models
- Other applications and AutoMod models.

Communication between two or more models allows for parallel and distributed simulation. MCM also includes Multi-Model Sync (MMS), which keeps the event lists of the linked models synchronized.

[Delete this sentence (see next paragraph)?] In order to facilitate communication, MCM enables sockets, Object Linking and Embedding for Process Control (OPC), and Dynamic Data Exchange (DDE) technologies. Connecting across different platforms is achieved using sockets technology, which allows communication through strings or C structures. Models may be easily linked to other applications using DDE. Examples of this include Excel spreadsheets, Access databases, ergonomics programs, and control programs.

8.1 Emulation with OPC

When MCM was added to the AutoMod family, it quickly became apparent that AutoMod users were applying the concept to a wide range of industrial applications. Although sockets technology is robust and works efficiently across different computing platforms, other technologies may provide better solutions for specific applications. One of these is OPC, or Object Linking and Embedding (OLE) for Process Control, a *de facto* industrial standard adhered to by all major control systems hardware manufacturers, including Rockwell Software, Siemens, Mitsubishi, GE, Schneider, Modicon, and many more. This well-defined standard makes it simple for users to link AutoMod emulation models to Programmable Logic Controllers (PLC) from almost any supplier. This enables the testing of real control programs using an AutoMod model to provide the same responses as the physical system. The benefits of this are considerable:

- Controls departments no longer have to wait for the system to be installed before beginning commissioning tests.
- Testing can be more complete and is much more convenient.
- Tests can be run in parallel.
- Operators can be trained offline on the real control system.
- Operators can see the whole system respond to their input.

The growing use of emulation is creating more opportunities for simulation departments to build models in collaboration with controls departments and increasing the area of application for simulation technology within materials handling, production, and automation companies.

8.2 How Emulation Differs from Simulation

Emulation is a powerful and flexible way to test the operation of industrial control programs before they are installed on-site. While a simulation model is used in the design and development phase of a project to produce a better solution, the role of an emulation model is much more precisely defined for the verification of the controls system. The emulation model is used to provide the control system with the same responses it would get from the physical system. A simulation model is used to generate results from different sets of operational parameters. Random number generators are used to closely match the uncertainties of the real system. Analysis programs are used to calculate trends and identify anomalies. Emulation models are used to step through a series of pre-defined sets of parameters that are unlikely to contain random elements; a "checklist" of situations is verified and the response of the control system is followed to detect any malfunctions.

8.3 The Advantages of Emulation

Figure 7 shows the emulation of a paint shop. The replacement of the real system by a virtual one offers several advantages:

- It can be ready for use before the real system.
- It can be duplicated at minimal cost.
- It does not disrupt existing production.
- It exists in a completely controlled environment.
- It can provide an overall view that is often impossible in the real situation.

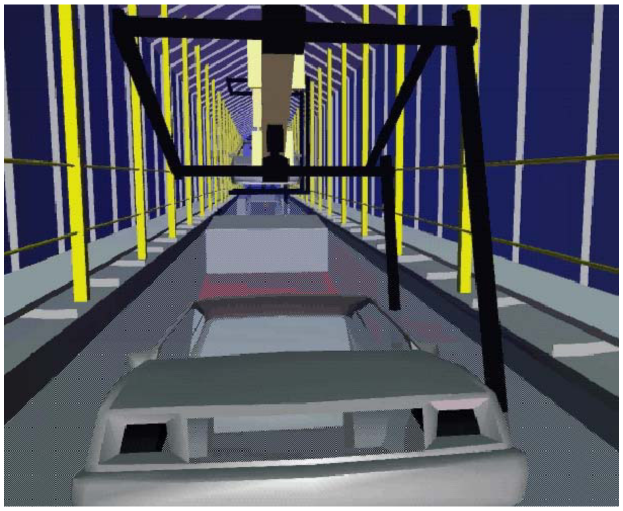

Figure 7: Emulation exhaustively tests control systems

An emulation model driven by a real control system provides a convenient and safe way to perform a series of standard operational tests that would otherwise have to be carried out on-site or at test sites.

8.4 Controlling the Commissioning Environment

As automated systems are designed to operate under loading conditions projected five or more years into the future, a common commissioning problem is a lack of suitable or sufficient loads with which to test the system. As a result, full testing on-site may be difficult, expensive, time-consuming, or even impossible. In addition, the commissioning and start-up phases of most automation projects are tightly scheduled, and start-up dates can easily slip. Commissioning problems may also be hard to trace and diagnose, because commissioning is often the first time the hardware system is combined with the control system; a missing signal, for example, could be the result of either a control system error or a misaligned detector.

As a result, emulation is most profitably applied during the commissioning phase. The use of emulation allows standard control system verification to be done outside of the commissioning window, leaving more time for other steps. Savings are generated as on-site costs are reduced. Also, verification can be carried out on several identical models simultaneously, rather than on the single real system. And emulation allows for more complete system tests under a wider range of conditions than would be possible during commissioning. The obvious result of more complete testing is fewer problems during ramp-up and normal operation, and this leads to further savings on maintenance, staff turnover, and unforeseen interventions.

9 TEACHING AUTOMOD

Teaching AutoMod has been greatly facilitated with the second edition of the text *Getting Started with AutoMod* (Banks 2004). Electronic files are available for all of the example models in the text. These are all downloadable from www.automod.com, as is the Student Version of AutoMod 11.1. Additionally, professors can receive a file with the solutions to all of the exercises in the text.

10 SUMMARY

AutoMod is a state-of-the-art simulation system that lets you define the physical elements of a system using true-to-scale, 3-D, CAD-like graphics and a logical portion of the system using a powerful procedural language. A typical user can be three to ten times more productive with AutoMod than with any other simulation tool. The accuracy and degree of detail with respect to model development is unequalled.

The AutoMod architecture allows construction of very large, complex models. In fact, the larger the project, the more benefits AutoMod provides over other simulation packages.

AutoMod provides realistic, 3-D visualization. There are no limits to the views or the size of the animation. The degree of animation realism is also unmatched, and AutoMod facilitates faster model building and better communication of model results.

The AutoMod product suite, including the AutoStat, AutoView, and MCM extensions, gives the simulation user a truly comprehensive solution. Through the use of Applied Materials technologies, models can become more than design tools. Models can be used to test controls, operate the facility, and plan for expansion.

REFERENCES

Banks, J. 2004. *Getting Started with AutoMod*. Bountiful, UT: Brooks-PRI Automation.

L'Ecuyer, P. and R. Touzin. 2000. Fast combined multiple recursive generators with multipliers of the form a = $\pm 2^q \pm 2^r$. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J.A. Joines, R.R. Barton, K Kang, and P.A. Fishwick, 683-689. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers.

AUTHOR BIOGRAPHIES

TODD LEBARON has worked for Applied Materials since 1990 and was recently made the Product Manager for Applieds Simulation Products. He has worked as a simulation analyst, conducting numerous simulation studies over the past fifteen years in a variety of applications. He has lead the development of the ToolSim™ Simulation Software, and teaches AutoMod training courses. Mr. LeBaron received a B.S. in Manufacturing Engineering from Brigham Young University in 1988. His email address is <todd_lebaron@amat.com>

CRAIG JACOBSEN is the project manager for simulation products at Applied Materials. Over the last 12 years he has worked as a support technician, manager of support and training and product manager for simulation products. Mr. Jacobsen received a B.S.E.E.T from Weber State College in 1977, a B.S.C.S from Weber State University in 1992, and an M.B.A. from the University of Utah in 2002. His email address is <craig_jacobsen@amat.com>