*Proceedings of the 1998 Winter Simulation Conference*
*D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.*

# AUTOMOD™ BY AUTOSIMULATIONS

Tyler Phillips

AutoSimulations, Inc.
655 Medical Drive
Bountiful, UT  84010, U.S.A.

## ABSTRACT

AutoMod differs significantly from other simulation systems because of its ability to deal with the physical elements of a system in physical (graphical) terms and the logical elements of a system in logical terms.

AutoMod also offers advanced features that allow users to simulate complex movement (kinematics and velocity) of equipment such as robots, machine tools, transfer lines, and special machinery. All graphics are represented in 3-D space with unlimited viewing control including: translation, rotation, scale, light-sourced solids, perspective, and continuous motion viewing.

AutoMod's CAD features are used to define the physical layout of manufacturing, material handling, and distribution systems. Unlike most other simulation languages, AutoMod's powerful graphical interface accurately captures the physical constraints of distance, size, and space in 3-D. (See Figure 1)

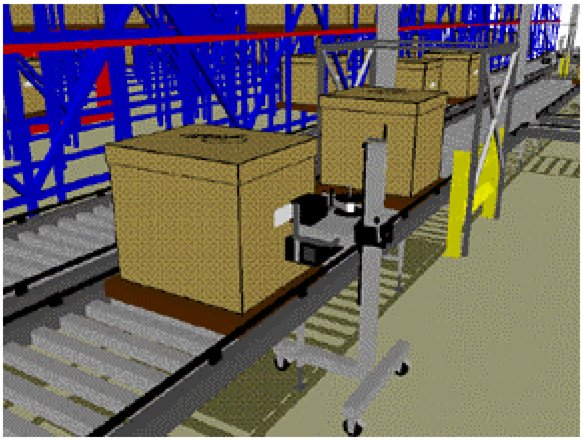

Figure 1:  AutoMod Model Graphics

AutoMod is really two programs. The build portion is for the physical and logical model definition  The resulting executable model is fully interactive, and can be stopped at any instant in simulated time to view statistics and model status.

Significant enhancements to conveyor movement systems and resources have been added in the newest version of the software, AutoMod v8.6.  These enhancements solidify AutoMod’s position as a world class simulation tool for accurate and precise, true-to-scale modeling of material handling, manufacturing, and distribution systems.

## 1 INTRODUCTION

*AutoMod’s* strength is that it combines the ease of use of a “simulator” with the power and flexibility of a simulation language.  Simulators usually address specific real-world problems with predefined constructs, making model definition quick and easy.  *AutoMod’s* movement systems aid users in defining the movement of material either manually or by automated equipment.  Material movement systems include:

- Path Mover (path/vehicle systems such as fork trucks, AGVS, and human movers)
- Conveyors (including belt and roller types)
- Automated Storage and Retrieval Systems (ASRS)
- Robots
- Bridge Cranes
- Power and Free Chain Conveyors

To define material movement systems, the user simply creates geometric entities such as paths and stations, then inputs the operating parameters such as velocity and acceleration.  *AutoMod* then automatically creates the corresponding model logic.  3-D animation is created automatically as well, providing a realistic picture of how a facility will operate.  Model animation can be viewed from any angle or perspective, providing visualization capabilities unmatched in other simulation systems.

A big difference between *AutoMod* and other simulation software is that users deal with geometry in a graphical way and create logic using a fourth-generation simulation language. This approach is intuitive to engineers who understand how their facilities operate but who do not have experience with simulation tools. Model output includes 3-D animation, automatically collected statistics, and business graphs. *AutoMod* provides users with a rich environment that facilitates thorough understanding and analysis of manufacturing systems.

The animation provided in *AutoMod* models is concurrent, meaning that the graphic pictures are running real-time with the simulation model. The model execution environment is very interactive, allowing the user to stop and start the simulation, run without animation in an accelerated time scale, and select objects from the animation screen to gather more detailed information. 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 and provide a forum for communication about system performance among project team members.

## 2 AUTOMOD INTERFACE

An *AutoMod* model consists of one or more systems. A system can either be a Process system, in which control logic is defined, or a movement system. Each model must contain one Process system and may contain any number of movement systems. Processes can contain complex logic to control the flow of either manufacturing materials or control messages, to contend for resources, or to wait for user-specified times. Loads can move between processes with or without using movement systems.

Loads that flow through the process logic have the ability to claim and release resources; enter and leave queues; be added to and removed from Order Lists; change the value of variables, counters, and load attributes; create a new load or kill an existing load; read from and write to external files; and determine the next process. All interarrival and event times can be represented by deterministic values or be derived randomly from one of several statistical distributions. *AutoMod*'s interface is window-oriented, utilizing pop-up and pull-down menus, dialog boxes, selection lists, and a mouse-based editor for developing process logic.

## 3 AUTOMOD'S WORLD VIEW

An *AutoMod* model is made up of a collection of systems. Any number of movement systems can be defined, and a process system connects the movement systems to the logical flow of products. In the process system, loads (products, parts, etc.) move between 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

- Use machines/operators
- Move into queues
- Clone new loads
- Change load types
- Wait on user-defined delay lists (order lists)
- Increment/decrement counters
- Set variable values
- Read from data files
- Send to other processes
- Make conditional tests

### 3.1 Processes

The process system is the backbone of an *AutoMod* model, providing the general-purpose simulation features required to model real-world problems. While material movement is important, it is not critical in manufacturing. No value is added to product while it is begin moved around a manufacturing facility. Value-added operations are performed by machines, processes, and humans that exist in the facility. The *AutoMod* process system is where the value-added operations and control logic are defined.

*AutoMod's* process system includes a simulation language based on action statements. Action statements combine the power of a structured language with the ease of use of 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 is easy to model almost any real-world situation. Figure 2 is an example of a process procedure.

### 3.2 Loads

Loads are the active entities in *AutoMod* and can be created in many ways, including deterministic or probabilistic generation. Their interarrival rate can be read from an ASCII data file or attached to a statistical distribution. *AutoMod's* predefined random distributions 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 makespan. Attributes can be accessed and modified in

*AutoMod's* action statements. Loads have 3-D shapes and dimensions like most other entities in *AutoMod*.

```
begin Pfixit arriving procedure
  move into FixQ
  if FixedYet = 1 then send to die
               else begin
               choosea resource from among
                                Picker(1), Picker(2)
              whose current loads is                    minimum
                 save choice as RecIdle
               use RecIdle for uniform 10, 5 min
               set FixedYet to
               send to SizeWeigh
  end
end
```

Figure 2: Sample Procedure Logic

### 3.3 Resources

Resources in *AutoMod* are used to represent machines, operators, fixtures, containers, and any other finite capacity objects. There are two default categories of resource state. The first is the working category (busy or idle), and the second is the availability category (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 are either part specific or are applied to all parts using a particular resource. Resources can also be preempted if a higher priority load needs immediate attention.

Resources can have downtimes, which are defined by MTBF (mean time between failures), and MTTR (mean time to repair). These times can use *AutoMod's* statistical distributions to represent random failures. Though MTBF is calculated by model-simulated time by default, it can be based on parts processed, as well as 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 PM's occur for a resource. Procedural actions can also be called at certain events within a resource cycle

### 3.4 States

States other than the default states can be defined by the user. These states might be used to represent conditions such as blocked, starved, PM, offline, etc. The state of a resource can then be changed in *AutoMod*'s procedural code so that statistics can be tracked for each state.

In addition, state monitors can be defined and used to track states for entities other then resources, such as vehicles, conveyors, or particular areas of a facility.

### 3.5 Queues and Order Lists

Because an *AutoMod* model is graphical as well as logical, loads must be located somewhere in the physical space of the model. *AutoMod* has two types of physical space: movement systems and queues. Queues in *AutoMod* are both physical and logical, and they can have capacities ranging from one to infinity. When a queue has reached its capacity, the next load trying to enter that queue must wait until there is space available. Loads are sorted in queues by using order lists. Queue contents can be shown dynamically in the animation, and loads can be "stacked" in any direction.

Loads that are at a queue or process may be sorted or 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 not a physical entity like a queue, but a logical element that provides a way to sort loads that have been delayed for any reason. To remove a load from an order list, another load must execute 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 process logic. Order lists can be sorted by load priority or other load attributes, either in ascending or descending order.

### 3.6 Variables and Counters

Data may be stored in an *AutoMod* model using variables. Variable values are changed using the "set" action as follows:

```
set Vtime to 123.456
```

Variables can be used in calculations or can be logically compared to other variables. In addition to integer, real, and string types, variables can also be used to store:

- processes
- queues
- resources

- order lists
- counters
- loads
- locations

By setting a variable to another *AutoMod* type, such as a process, elegant approaches can be used to construct flexible and concise model logic. *AutoMod* also supports the notion of arrayed entities, making it easier to model real-world systems that have some repetition.

Counters are similar to variables, except that they can only be positive integer values. Counters have a maximum capacity, making them useful for global traffic control. When a load tries to increment a counter that is filled to capacity, the load will be delayed until another load decrements the counter. Statistics for counters are collected automatically.

## 3.7 Types

Types in *AutoMod* provide users with a powerful tool for creating, among other things, lists of entities such as resources or stations, and then making complex decisions based on the contents of the lists. Some of the default types are integer, real, resource, location, etc. But with user defined types, it is possible to create, for example, a variable of type ResourceList, whose data can then be manipulated in any number of ways to facilitate complex modeling constructs and decisions.

## 3.8 Traffic Limits and Blocks

*AutoMod* has a powerful means of controlling the number of entities that can either be in processes or occupying physical space. Process traffic limits control the number of loads that are either on their way to or are in a process. Blocks control the number of entities occupying a physical space, making them very useful for controlling transporter vehicle collisions.

Blocks may have any capacity, but the capacity is generally set to one. Transporter vehicles (AGVs, fork trucks, etc.) 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 cylinders, cubes, cones, or frustrums.

## 3.9 Text Labels

*AutoMod* provides the ability to use text labels to enhance model communication. Labels may be added to any location in the model's physical space, and they can either be static or dynamic during the simulation. Labels can rotate when the animation view changes or can be attached to a fixed position on the screen.

## 3.10 Tables

Tables in *AutoMod* supply the user with the means to collect statistics on any model entity and to classify those statistics for better understanding of their distribution. Tables automatically provide average, standard deviation, maximum, and minimum for all values entered in the table. Users can define the number of "bins" or categories, and *AutoMod* adds values to the appropriate bin when directed by the user in the tabulate action statement.

## 3.11 Graphs

Graphs in *AutoMod* are easy to define and update real time with the animation. 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, including transporter vehicle velocity, number of loads on a conveyor section, or average utilization of a machine. Graphs can be printed or plotted to a variety of supported output devices, and graph displays can be controlled using *AutoMod* language.

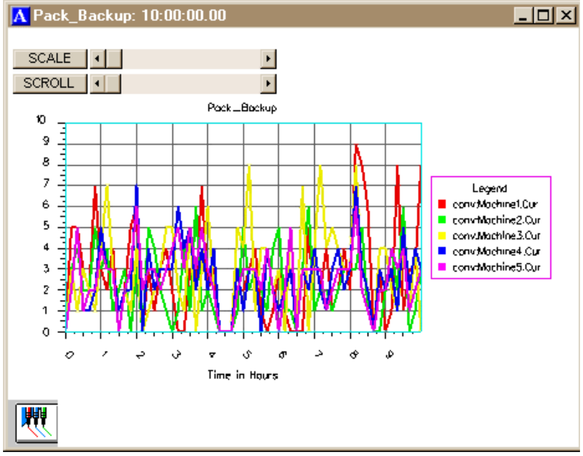

Figure 3: AutoMod Timeline Business Graph

## 3.12 Run Control

Run control in *AutoMod* allows users to define the warm-up and steady-state periods for the model by resetting time-persistent statistics. Reports can be printed for any run control period, or "snap." Business graph output can be automatically created, as well as post-processed animation records.

## 4 AUTOVIEW

AutoSimulations' post-processed animation module, can be used to view the animation records created while running an *AutoMod* model. Run control also provides an entity tracing capability that gives the model builder an event-by-event account of the model run. This information is useful in verifying and validating a model.

## 5 GRAPHICS IN 3-D

Both dynamic and static objects can be displayed during model execution. Figure 4 shows a screen shot of a typical AutoMod model during a model run. Dynamic objects represent loads, resources, queues, and statistics. The static layout is the background graphics of the plant. It may contain column lines, aisle markings, and walls. Labels can identify specific areas in the facility.

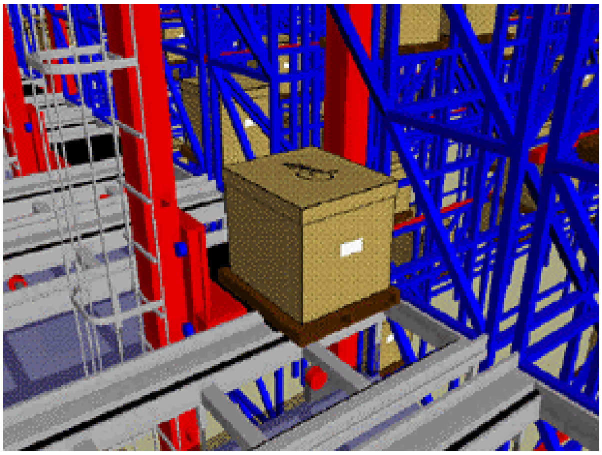

Figure 4: AutoMod 3-D Graphics

There are several ways to create a layout of the system to be modeled. *AutoMod* comes with a three-dimensional graphics editor that allows the user to construct objects from standard graphics primitives. Cone, Box, Hemisphere, Trapezoid, Frustum, Cylinder, Arc, Vector (list), Set, Text, and Triad are primitives that can be selected, placed, and scaled to create any static entity in the facility.

*AutoMod* also has a utility that allows users to import IGES format files directly into the *AutoMod* model building environment.

## 6 RUN-TIME ENVIRONMENT

In keeping with *AutoMod's* interactive features, the user has complete control of the model in the run-time environment. The model can be viewed with the animation on or run with the animation off. *AutoMod* uses concurrent animation; the simulation progresses as the animation picture is being updated. With animation off, the simulation doesn't draw the picture, but it still performs all simulation calculations. The user can suspend the simulation at any instant to review statistics through pop-up windows, to take resources down, to set break points or alarms, or to control the view of the animation without constraint.

### 6.1 View Control

*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, *AutoMod* provides the ability to run in batch mode without animation. Whether the need is for a highly interactive mode or a batch mode, *AutoMod* lets you do it.

### 6.2 User Interaction

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

*AutoMod* also provides comprehensive reports. The reports can be displayed on request at any time during the animation. Printed versions of the reports can also be specified during the Model Development Environment.

*AutoMod* automatically keeps track of many statistics. These automatic reports are linked to specific entity types such as:

- Movement systems,
- Processes,
- Queues,
- Resources,
- Order Lists, etc.

Vehicle states are tracked during the entire model run, and reports are generated automatically.

Reports can be sorted alphabetically or numerically for easier analysis. The user can also develop and generate custom reports from within process procedures.

## 7 SUMMARY

*AutoMod* is an industrial-oriented simulation system that provides the ability to define the physical elements of a system using CAD-like graphics and the logical portion of the system using a powerful procedural language. The results are that a typical user can be three to ten times more productive using *AutoMod* in comparison to using any other simulation language. The accuracy and degree of detail with respect to movement systems is unequalled

*AutoMod* allows the construction of very large, complex models. In fact, *AutoMod's* structured language has proven that the larger the project, the more benefits *AutoMod* has over alternative approaches.

*AutoMod* provides real, three-dimensional graphic animation. There are no limits to the views or the size of the picture to be shown. The degree of animation realism is also unmatched, as *AutoMod* provides light-sourced solid graphics with Z-depth sorting, showing all entities in correct relation to one another on the screen.

Enhancing *AutoMod's* already robust capabilities are the following extensions and utilities:

- *Simulator* - Provides a powerful, manufacturing template for capacity planning.

- *AutoView* - *AutoMod's* post-process animation package that allows you to create a directed "walk-through of the model by panning, zooming, and moving back and forth in time and space. The animation view can be attached to a moving model entity like a load or vehicle, providing a realistic view from the "inside" of a model.

- *AutoStat* - Provides enhanced analysis of the statistics generated by *AutoMod* by calculating minimums, maximums, confidence intervals, Design of Experiments (DOE), and warm-up determination to determine model steady state.

## ACKNOWLEDGEMENTS

This paper was adapted from a paper by Matthew W. Rohrer originally published in the *Proceedings of the 1994 Winter Simulation Conference*.

## REFERENCES

AutoSimulations, Inc. 1998. *AutoMod User's Manual*

AutoSimulations, Inc. 1998. *AutoMod Beginning Class Notes.*

## AUTHOR BIOGRAPHY

**TYLER PHILLIPS**, Western Regional Account Manager, joined AutoSimulations, Inc. in 1994. Over the past 4 years, Mr. Phillips has trained hundreds of users at all levels in most of AutoSimulations' products, including *AutoMod*, *AutoSched*, and *ASAP*. He also spends several days every year lecturing on Customer Service. His main interest is in providing excellence in training and customer support for all AutoSimulations' customers. Mr. Phillips holds a BA from Brigham Young University.