Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

AUTOMOD

Van B. Norman

AutoSimulations
655 Medical Drive
Bountiful, Utah 84011, U.S.A.

Kenneth D. Farnsworth

AutoSimulations
655 Medical Drive
Bountiful, Utah 84011, U.S.A.

ABSTRACT

AutoMod is an industrial simulation system that combines CAD-like drawing tools with a spreadsheet interface and a powerful, engineering-oriented language to model management strategies, control systems, and material flow. Unlike most other simulation languages, AutoMod's powerful graphical interface accurately captures the physical constraints of distance, size, and space, resulting in accurate three-dimensional detail.

1 INTRODUCTION

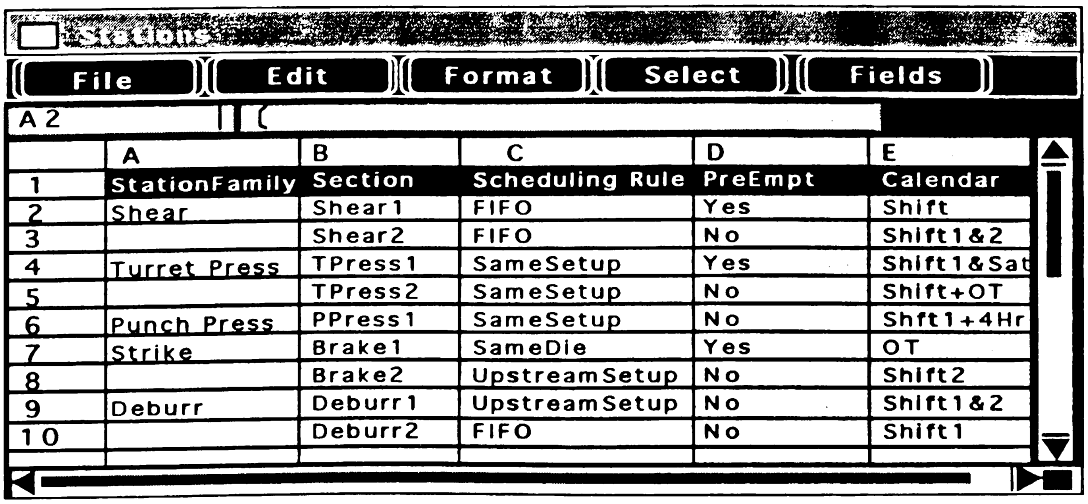
AutoMod contains a set of movement systems that have been developed from AutoSimulations Inc.'s (ASI) experience in industrial automation. Therefore, the underlying model logic is automatically generated from the geometry and the movement system input parameters. The 3-D animation is an automatic part of model development and exactly depicts the logic of the model.

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

AutoMod consists of two environments. The Model Development Environment is for defining the model's logic and graphics. The model is then compiled into an executable model and run in the Simulation Environment, where the simulation and animation run concurrently. The executable model is fully interactive; it can be stopped at any instant in simulated time to examine statistics and model status and to conduct interactive modeling experiments.

2 MODEL DEVELOPMENT ENVIRONMENT

An AutoMod model consists of one or more systems. All models require a logical system, which can be either a Process system or a Simulator system. A Process system is used to define control logic for material handling applications using AutoMod syntax. A Simulator system defines model logic for complicated process flows and routings using a spreadsheet interface. In addition to a logical system, a model may contain any number of movement systems.



	A	B	C	D	E
1	StationFamily	Section	Scheduling Rule	PreEmpt	Calendar
2	Shear	Shear1	FIFO	Yes	Shift
3		Shear2	FIFO	No	Shift1&2
4	Turret Press	TPress1	SameSetup	Yes	Shift1&Sat
5		TPress2	SameSetup	No	Shift+OT
6	Punch Press	PPress1	SameSetup	No	Shft1+4Hr
7	Strike	Brake1	SameDie	Yes	OT
8		Brake2	UpstreamSetup	No	Shift2
9	Deburr	Deburr1	UpstreamSetup	No	Shift1&2
10		Deburr2	FIFO	No	Shift1

Figure 1. AutoMod's Spreadsheet Interface

3 SIMULATOR SYSTEM

AutoMod's Simulator system is designed to model complicated routings of multiple products, showing the competition for equipment, personnel, tools, and raw materials. This competition for finite resources is modeled using the Simulator system's spreadsheet interface to define all factory resources, products, and routings.

3.1 Factory Resources

Factory resources represent standard elements of a facility, such as stations (places where work is performed), operators, and tools. These resources can be grouped into families based on their ability to perform interchangeable tasks. Parts travel from family to family to be processed. Operators can have additional characteristics, such as efficiencies and certifications. The user may also create custom graphics for both stations and operators.

All types of factory resources may have calendars attached to them, which can specify shifts, downtimes, preventative maintenance, holidays, and other calendar-based events and constraints.

3.2 Products

Products are both the parts that travel through the system and the routings that define that travel. Parts can be manufactured or purchased. The user can also represent a Bill of Material, in which subparts are consumed by other parts during processing.

A part's routing describes the steps required to process the part into a final product. The routing may define the necessary processing time, machine setup, batching criteria, and value the part has at each step in its process. You may also specify alternate routings, which is useful when modeling rework procedures.

3.3 Demand

The user can define all orders for parts, including parts that are in-process (WIP) at the simulation start time. All orders, including WIP lots, can have a start and a due date. In addition, WIP lots can have a specified starting value, representing their worth at the beginning of the simulation.

3.4 Rules and Task Selection

The two decision-making entities in the Simulator system are stations and operators. These two resources can decide what lot to work on next by using task selection rules. There are 16 default task selection rules, with six versions of each rule. These rules, which are based on criteria such as First In, First Out (FIFO), same setup, earliest due date, and many other factors, can help a station or operator select a lot to process. Rules can branch to other rules or can skip to another part of the same rule. Many standard rules check for preventative maintenance orders first, and can also check whether all necessary tools and resources are available before claiming a lot to process.

3.5 Customization

The Simulator's spreadsheets have over 200 standard fields that allow you to define your factory. In addition, there are over 90 custom fields and over 70 User Exits, which allow you unlimited flexibility when building a model. The User Exits are located at decision points throughout the simulation and allow users to tailor the default behavior of the Simulator to match the real world. These custom fields and User Exits help the Simulator to be easy-to-use, yet powerful.

4 PROCESS SYSTEM

The process system provides the general purpose simulation features required for simulation. Even if you use the Simulator system to model your control logic, there are entities in the Process system that you may use, such as resources, queues, variables, and order lists.

AutoMod's process system is both powerful and easy to use. Process logic is defined in process procedures. Process procedures can contain complex logic that controls the flow of manufacturing materials, contends for resources, or waits for user-specified times. AutoMod's process procedures use an easy-to-use, English-like language that contains:

- if-then-else logic
- while loops
- access to global and load-specific variables
- actions to use, take down, or bring up resources
- actions to multiply loads
- actions to choose processes, resources, or queues based on their state

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

Figure 2. Sample Procedure Logic

There are virtually no limits to the size and complexity of the logic that can be developed. And there is rarely a need to use a lower-level programming language, because AutoMod provides the flexibility required to simulate any task.

4.1 Loads

Loads are the active elements in AutoMod's Process system. They are created in three ways: from a creation specification using one of the standard statistical distributions, deterministically from reading data from a file, or based on user-defined distributions. Loads are named and may have user-defined attributes. These attributes are variables which are unique to the load. For example, a load that represents a car might have load attributes that indicate what color it is, what level of trim it has, or whether it receives air conditioning. These attributes change as a result of the model logic in the process procedures. Like most entities in AutoMod, loads can have 3-D graphics.

Loads that flow through the process logic have the ability to claim and release resources, enter and leave queues, be added 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.

4.2 Resources

A resource is a general and flexible entity that can be used to represent a machine, an operator, a fixture, a container, etc. Often several resources are used in a process in a similar fashion.

There are two levels in which the resource state is categorized. The first level is whether the resource is Busy or Idle (its state with respect to a load). The second level is the resource's availability, whether it is Up or Down.

Loads use resources for specified processing times, which are based on a standard time. Users can create variations of this standard time using either the random number distributions that are built-into AutoMod or using custom distributions. The processing times can be general for all loads or specific to each product type.

Resources can have downtimes. During the downtime period, the resource accumulates statistics in the down state. When the resource becomes available, it continues to work on the preempted load for the remaining processing time. Mean-Time-Between-Failure (MTBF) and Mean-Time-To-Repair (MTTR) are included and can be based on the same built-in statistical functions or on custom curves. The standard MTBF is based on model simulated time. AutoMod can easily accommodate MTBF based on machine run-time or machine cycles.

4.3 Queues and Order Lists

When loads are modeled, they must always reside in a physical space. AutoMod uses two types of physical space: movement systems and queues. Queues have capacities which can range from 1 to infinity. If a queue is full (it has reached its capacity), the next load must wait until there is room. Loads within queues can be sorted and sequenced using order lists. The queue contents can be shown dynamically during the animation.

Loads may be sorted and delayed at a process or queue until they are explicitly ordered to leave. A load can be directed to place itself on an order list. An order list is not a physical element; rather, it is a way of sorting loads that are delayed for some reason. Once on an order list, the load still remains in the process and physical territory it was in prior to being put on the order list.

The load can be ordered from that list by another load or from an order action in another part of the model. The load that has been ordered may continue on its way, or it can be sent to another process by the ordering action.

Order lists are not attached to specific processes. Many processes may place loads on the same order list. Likewise, when a load is ordered to a process, that process has no control of where it is coming from.

4.4 Variables and Counters

AutoMod provides a number of ways of storing values during the simulation period. Variables are data structures that can change as a result of a load executing the appropriate statement in a process procedure. Variables can be used in calculations or can be logically compared to other variables. In addition to integer and real numeric values, variables can also represent:

- Process Names
- Queue Names
- Resource Names
- Order List Names
- Counter Names
- Load Names

By storing the name of an AutoMod entity into a variable, extensive if-then-else logic can be avoided. One example of this would be sending a load to a variable that has the process name stored into it.

Counters are similar to variables, but are positive integers only and have a maximum capacity. A load trying to increment a counter already at its maximum capacity will be stopped until it can successfully perform the increment. Statistics are kept on counters throughout the simulated period.

4.5 Blocks and Traffic Limits

AutoMod is capable of controlling the number of loads that are either in processes or within physical space in the facility. Traffic limits prevent too many loads from being in a process, while blocks provide the same utility for physical space.

Blocks are like counters in that they use a set limit—they cannot be incremented beyond the limit. Normally the limit is one, so only one entity can occupy the physical space at a time. Blocks are commonly used at AGV intersections to prevent vehicles from colliding.

AGVs automatically increment/decrement blocks when entering/leaving blocks. Blocks can also be incremented and decremented from process procedures. This is useful, for example, to prevent having a bridge crane and an AGV collide.

4.6 Run Control

By using run control features, various experimental runs can be compared. Run control defines the length of the simulation (in simulated time units), when reports are printed to a file, and when statistics are reset. Resetting the statistics allows the model to run for an initialization, or "priming", period prior to running the model in a "steady state" period.

AutoMod's statistical package, AutoStat, helps you determine warmup periods, allowing "warm-up statistics" to be disregarded and "steady state" statistics to be used. AutoStat also allows you to make multiple runs using different data factors, and to compare those runs to find the most significant factors. This helps make model analysis easier.

5 PICTURE CONSTRUCTION IN 3-D

Both dynamic and static objects can be displayed in the model. Dynamic objects are things such as 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.

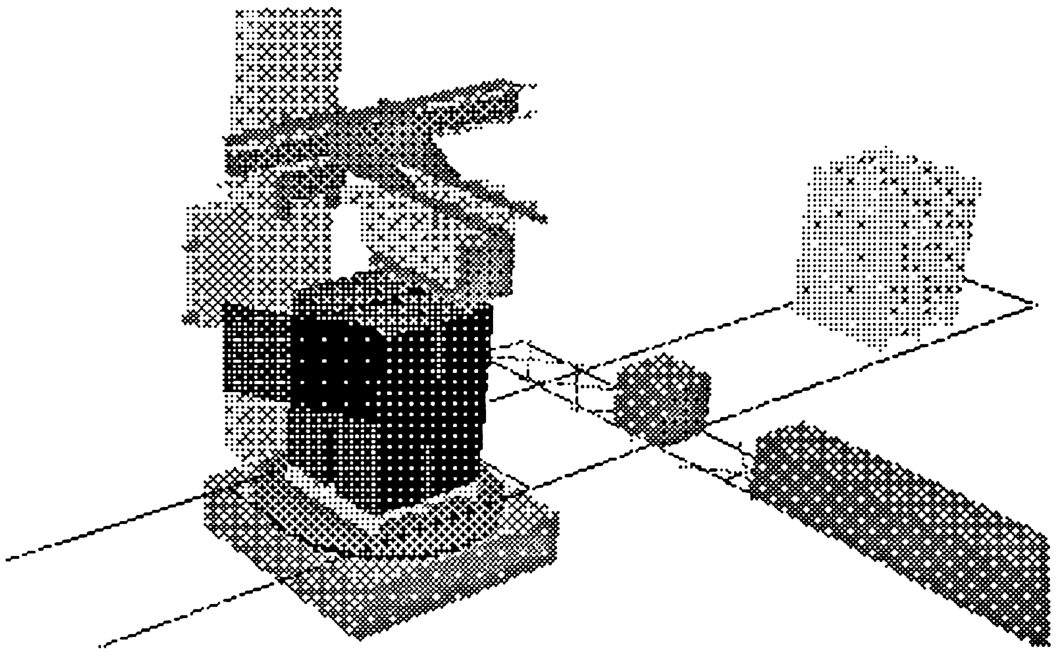


Figure 3. An AutoMod Model at Runtime

There are several ways to create a layout of the system that is being modeled. AutoMod comes with a three-dimensional graphics editor that allows the user to construct objects from standard graphics primitives such as Cone, Box, Hemisphere, etc. These primitives can be selected, placed, and scaled to create any entity.

AutoMod also has an optional utility called IGES/Sim. The acronym "IGES" stands for the Initial Graphics Exchange Standard. IGES is an industry standard exchange format for translating the graphic data from one CAD system to another. Any IGES file of a plant layout that was created from a CAD system can be easily imported into AutoMod using IGES/Sim. IGES/Sim can also export AutoMod graphics files to the IGES format.

6 SIMULATION ENVIRONMENT

As in AutoMod's Model Development Environment, the user has complete control of the model in the Simulation Environment. AutoMod uses concurrent animation—the simulation progresses as the animation picture is being updated. The model can be viewed with the animation on, or run with the animation off. With animation off, the simulation doesn't draw the picture but still performs all simulation calculations. The user can suspend the simulation at any instant and review statistics through pop-up windows, take resources down, set break points or alarms, and 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. Figure 2 shows the View Control window. This window provides the ability to position the graphics of the model in any orientation. The View Control allows rotation, translation, and scale to be dynamically changed with respect to all three axes (X, Y, and Z). The animation picture can be shown in solid mode with all the correct Z sorting, so that hidden lines and surfaces are accurately represented. The animation picture can be shown in either perspective mode or orthographic mode.

All view control actions have keyboard equivalents, and there is a friction toggle option which allows the user to spin or translate the model picture continuously. These are helpful features when the model's animation is being video taped or filmed. AutoMod has a filming package, AutoView, that allows the user to create a script to perform "walk-throughs" of the model, panning, zooming, and following a load or vehicle through a process. AutoView can make presenting simulation results easier and more effective than if the model was simply shown from start to finish.

6.2 Debugging

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

6.3 Model Output

AutoMod provides comprehensive reports. The reports can be displayed upon request at any time during the animation. Printed versions of the reports can also be specified.

AutoMod automatically keeps track of many statistics, and the reports are linked to specific entity types, such as:

- movement systems
- processes
- queues
- resources
- order lists, etc.

Reports can be sorted alphabetically or numerically for easier analysis. The user can also develop and generate custom reports from within process procedures. User-defined Business Graphs can be checked and updated as the simulation progresses.

If model logic is created with the Simulator system, there are additional reports and Gantt Charts available while running the model. These Gantt Charts and reports are useful for both debugging the model and for comparing results from multiple runs.

AutoStat provides enhanced statistical analysis capability for AutoMod. AutoStat can determine the model warmup and compute confidence intervals for model statistics. Its Design of Experiments feature reduces the number of experiments required to determine which values, from a set of options, provide the best system performance.

7. SUMMARY

AutoMod is a industrial-oriented simulation system that allows the user to define the physical elements of a system using CAD-like graphics and to define the logical portion of the system using a spreadsheet interface and a powerful procedural language. The accuracy and degree of detail with respect to movement systems is unapproached.

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 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, so all entities are shown in the correct relation to one another on the screen.

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

- AutoSched - Provides a powerful, fully featured finite-capacity planning and scheduling system.
- AutoView - AutoMod's post-processed 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, synchronized to the simulation clock.
- AutoStat - Provides enhanced analysis of the statis-

tics generated by AutoMod by calculating minimums, maximums, confidence intervals, and steady state information.

- IGES/Sim - A CAD-transfer utility that allows graphics to be brought into AutoMod from a CAD package, and to be written from AutoMod to a CAD package.

REFERENCES

AutoSimulations, Inc. 1993, *AutoMod User's Manual.*
AutoSimulations, Inc. 1993, *AutoSched User's Manual.*

AUTHOR BIOGRAPHY

VAN B. NORMAN, President of AutoSimulations, Inc., received a B.S. in mathematics at the University of Utah in 1969. He spent 6 years at Eaton-Kenway, where he implemented the first simulation animator. In 1982, using his experience in factory automation and simulation, he co-founded AutoSimulations, Inc., where he co-authored AutoMod, ASI's first graphic simulation software. He has authored papers on the application and the future of simulation in manufacturing. His interests are in world-class manufacturing operations and simulation research.

KENNETH D. FARNSWORTH, Software Development Manager of AutoSimulations, Inc., has been with the company for eleven years. He has been involved with the development of many products, including AutoMod, AutoMod II, and AutoView. He is currently managing a special project team that is developing AutoSimulations' next generation of simulation development tools. Farnsworth holds a B.S. in Physics from Brigham Young University.