AUTASIM: A SYSTEM FOR COMPUTERRED ASSEMBLY OF SIMULATION MODELS

Dr. Robert T. Burger

General Research Corporation

ABSTRACT

The AUTASIM System (Automated Assembly of Simulation Models) is described. The system is designed to rapidly assemble discrete event, stochastic simulation models of node network systems. Systems of this type are distribution systems, plant operations, systems for the provision of services, transportation systems, and combinations of these. Models can be used for comparative analysis of new concepts and proposed systems with other systems.

The main AUTASIM components are a Module Library of computer programs, a program called the Model Assembler and a Model Description Language. The Library contains functional and simulation service modules which are the "building blocks" for the simulation models created. The Model Assembler program reads a coded model description, selects the required modules from the Library, and creates the necessary linkage routines for a complete model program.

The Model Description Language, AUTASCRIPT, is defined. The operation of the FORTRAN coded Model Assembler and linkage control programs is described, as is the GASP based model simulation control.

INTRODUCTION

Most operations research analysts have, at one time or another, wanted to use a simulation model for the analysis of a problem being studied. Research of available models usually reveals that the formulation needed hasn't been developed, or an existing model doesn't match the problem. Faced with this situation the analyst could modify the problem to match the available model (usually an unwise alternative), modify the available model, or develop a model to match the problem. The formulation and implementation of complex simulation models until recently has required a lot of talent, a lot of effort and a lot of time. Modifying existing models can be as costly as developing new models. Many studies that could have benefited from the application of simulation models bypassed that route because of lack of time or resources.

This paper will describe a system that has recently been developed by the General Research Corporation which provides the capability to automate the assembly of simulation models. The system was developed under a contract with the US Army which required a model of the Army logistic system. The study was given the acronym "MAWLOGS" (Model of the US Army Worldwide Logistic System) and had the objective of developing a simulation model to be used to "compare proposed systems with each other and with the current system to determine the relative merits of each system."

The vast and complex nature of Army logistics was appreciated by the analysts assigned to the study. The initial research revealed that the logistic problems that would need to be addressed by a model ranged from studying a single function within a small segment of the system, to those of a worldwide nature, usually multifunctional in scope, and multi-item in detail. Thus, flexibility appeared to be an essential characteristic in the choice of modeling approaches, because most of the problems identified focused on less than the total system, involved two or more interacting functions (e.g., supply, maintenance and transportation), and often varied with respect to level of detail. In addition, there appeared to be the need to treat one function, say supply, at one level of detail while treating other interacting functions at different levels. A single model of the Army worldwide logistic system with these characteristics appeared inefficient and indeed infeasible, considering available computers. Consequently, a modular modeling system seemed a reasonable approach, and the task became one of designing a system for the rapid assembly of simulation models of particular scope and level of detail designed to focus on the particular problem to be studied. Stated another way, the objective was to rapidly assemble custom-made simulation models of logistic systems. Such a system has been developed and is currently operational at GRC in McLean, Virginia.

The development of the AUTASIM system was accomplished as a component of the MAWLOGS system (Model of the Army Worldwide Logistic System) under contract to the Office of the Deputy Chief of Staff for Logistics, Department of the Army.

GENERAL DESCRIPTION

AUTASIM stands for Automated Assembly of Simulation Models. The purpose of the AUTASIM system is to create discrete event simulation models of systems which can be described as node networks. The system consists of three elements: a Module Library, a Model Description Language, and a Model Assembler program. The Module Library is a collection of modular computer routines, each simulating a specific activity, and service routines that provide the conventional elements of a simulation model. The Model Description Language is a methodology for describing the node network structure of a model and the activities which are to be simulated at the nodes in the model. The Model Assembler is a computer program which, given the description of a system to be modeled, will retrieve from the Module Library the required modules, link them together as prescribed in the description, and produce a computer program of the system model.

The AUTASIM system provides the user with a powerful model building capability. A modular "building block" concept enables models to be assembled for a very wide range of systems. The systems which can be modeled are those that can be described as one or more activity centers or nodes connected by links over which materiel or information flows. The modules on the library are designed to represent the activities or subactivities which are present in the system. Within each system node an activity network of nodes and links can be defined. Each activity node can likewise be defined as a node network of subactivity modules. Figure 1 shows how a system can be represented as different levels of module networks. This modularity facilitates the inclusion in a model of those features, and only those, which are required to accomplish the purpose of the model with no extraneous logic and waste of core storage.

Many systems can be described as node networks in this manner. Notable among them are distribution systems, shop or plant operations, systems for the provision of services, transportation systems, communication systems, and various combinations thereof. The system can be small--two nodes and one link--or large. The limit on the size and scope of the system modeled is controlled by the core storage capacity of the computer on which the model program is to be run.

The AUTASIM system is fully automated. Given the description of a model to be assembled, the Model Assembler program can assemble a model in a very brief period of time--five to ten minutes. This rapid model assembly capability also facilitates rapid changes to the activity content or system structure of a model, a characteristic often needed when comparing alternative concepts.

The computer programs in the AUTASIM system, the Model Assembler and the modules on the library, are written to the maximum extent feasible in USA Standard FORTRAN code. The model programs generated by the Model Assembler are also written in FORTRAN. This language was used so that the system could be easily transferred to other computers. All programs are currently operational on the CDC 6400 computer system at General Research Corporation in McLean, Virginia.

SYSTEM PRODUCTS

The AUTASIM system is not a simulation model or even a group of models built for a particular user, but the product of the system is a model which is custom tailored to the user's needs. Any model assembled by the AUTASIM system can be described as a discrete event, dynamic simulation

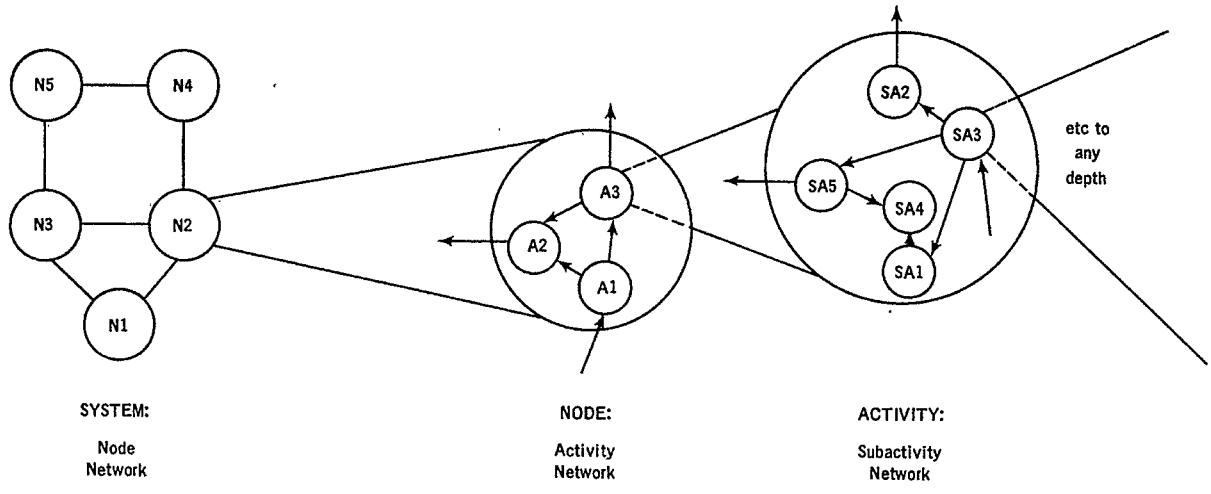

Fig. 1—Overview of AUTASIM Modular Approach

model. Most AUTASIM models are also stochastic; however, all elements of uncertainty in a model could be omitted by the model designer, which would yield a deterministic model.

Each model developed by the AUTASIM system can have the following set of features which are generally desired in a simulation model. Some of the features are included at the user's option but are strongly recommended. Every model has a complete statistics collection capability automatically included. Whether statistics are collected and in what form is specified in the input data deck at model execution time. A generalized output capability is included in each model. The user is free to specify at model assembly time what reports he desires from the model. Statistics may also be collected in detail on a tape file for postprocessing. A warmup capability is also included, allowing a model to be run from a "cold start" position to properly initialize model conditions before statistics collection is begun. A model restart capability is optionally included. This will allow the user to save the status of a model at any point in simulation time for future model restart from that point. A complete trace capability is also included which enables the user to follow the operation of the model.

The size of the system or node network which can be modeled using AUTASIM has a very broad range. Networks with two nodes and a single link up to multi-node, multi-echelon networks can be handled. Since the Model Assembler program develops a model program one node at a time, there is no practical limitation on the number of nodes which can be described in the model description and therefore represented in the model. However, the maximum number of nodes which can be simulated in a given AUTASIM model depends upon a number of factors including the size of the computer available, the functions to be represented, the level of simulation detail at each node, and the level of activity to be simulated. The size of the computer on which the model program is to be executed is the overriding limitation. The Model Assembler program, which can be run on a relatively small computer, can create a model program which could not be loaded into the core storage of some of the largest computers.

The functions and activities which can be represented in a model are determined by the contents of the Module Library. The library is considered to be the dynamic component of the AUTASIM system. As new areas of application or new activities within the current functional scope are desired, new modules will be developed and added to the library. This development process is relatively simple since the modules are programmed independently and the procedure for writing new modules is well defined. A large portion of the Module Library is independent of the activities which can be simulated. Included in this portion are programs to control the simulation, perform the necessary simulation bookkeeping, provide the statistics collection capability, and access the various data structures in a model.

Since the first application of the AUTASIM system was to develop Army logistic system models under the MAWLOGS contract, the functional modules currently on the library represent logistics activities. A short overview of the modules is given to show the reader how a particular application area is represented. These modules are cataloged by "family". Each family represents a separate logistic function--supply, maintenance, transportation, and communications, and each module therein a related activity. The supply and maintenance families simulate the activities associated with the support of fleets of equipment to include the generation of supply and maintenance demands, the repair and rebuild of end items and repairable components, and the supply of end items, components, and repair parts. The transportation family of modules permits the simulation of multimode, multicarrier operations, movement control, and terminal operations. Communications modules are available for simulating message dispatch, routing, and delivery. These modules operate at various levels of detail, thereby providing the user the flexibility needed to produce a variety of models.

MODEL DESCRIPTION LANGUAGE

The model description language, called AUTASCRIPT, is a special purpose code which is accepted by the Model Assembler program. AUTASCRIPT is used to define the content and structure desired in a simulation model. The code consists of a "vocabulary" of module names, a set of nine delimiters, and a set of rules for fitting module names and delimiters together to precisely describe a model structure. This code facilitates the description of very complex node networks, a capability which is a powerful tool by itself. Multifunctional networks so complex as to be almost impossible to draw in the form of node-network diagrams can be described in the model description language.

A demonstration of the use of AUTASCRIPT will be given after certain terms which have special meaning in the context of the AUTASIM system are defined. The definitions follow:

System - a network of nodes connected by links

Node - a special block of programming logic which can be referenced in a model. An activity center

Verb - any block of programming logic which can be included in a model by stating its name in the model description

Simple verb - a block of FORTRAN code. The logic for simulating an activity

Nonsimple verb - a structured assemblage of simple verbs into a larger block of logic

Module - any block of logic which is contained in the Module Library; the set of modules includes verbs, service routines, and common data structure decks.

Parameter slot - a point in the logic of a verb at which control may be transferred to logic outside the verb

The verbs on the Module Library are the vocabulary of the AUTASCRIPT language. The content of a model is the set of simple verbs or blocks of programming logic which are specified in the model description and therefore included in the model program. The structure of a model is the manner in which the set of content blocks is interconnected. The network of content blocks and the path of control flow through the content blocks is conveyed to the Model Assembler program in the model description.

The names of verbs through which control is to flow in the model are listed in a sequence and separated by commas in a model description. For example, given content blocks named "VERB1", "VERB2", and "VERB3", the sequence "VERB1, VERB2, VERB3" defines one flow path while "VERB1, VERB2, VERB1, VERB3, VERB1" defines another in which the same content blocks are referred to in a different order and more than once. In the first example, the content of "VERB1" would be given control of the execution, then the content of "VERB2", then of "VERB3". In the second example, control would flow through VERB1 three times during execution.

However, such sequences of content block references are inadequate by themselves since they have no beginning or ending delimiters and represent only one single sequence of execution. Therefore, provision is made for giving a name to a sequence of verb names by which the sequence can be referenced. This name is called a node name and a sequence preceded by a node name and a period and followed by a dollar sign forms a complete entity called a node. Examples of nodes formed from the above named content blocks are:

```
"NODE1.  VERB1, VERB2 $",
"NODE2.  VERB3, VERB2 $", and
"AAA.  VERB3, VERB1, VERB2 $".
```

Thus, the capability exists to define independent, identifiable logical flows in a model description. To indicate a transfer of control between nodes, a node may be referenced from within another node by including the node name preceded by an asterisk in the sequence of verb names. The above mentioned node "AAA" could be written as

```
"AAB.  VERB3, *NODE1 $"
```

and create the same logical flow. That is, the flow of control would be through "VERB3" and then to the verbs referenced as "NODE1", i.e., "VERB1" and then "VERB2".

The conventions for node names and for referencing node names allows the specification of more complicated structures but they do not permit specifying possible logical branches or "forks" in the flow of control. This capability is provided by parameter slots associated with a verb as described below.

Verbs are programmed to be highly modular so that they interface in a flexible manner with other members of the set. The logic of most computer programs consists of steps or parts, with some of the parts contained in subprograms that are called from within the program. In the case of a verb, all of the parts need not be included in the program or even referenced by name since a general external reference can be made. Parameter slots provide the capability of external references that need not be prespecified. A parameter slot can be inserted in the verb program at a point where logic outside the main verb logic may be applied to complete the function of the verb. The transfer of control is structured by the Model Assembler so that the particular logic to which control is given during execution of the verb is that specified in the model description. A verb that deals with reordering stock, for example, could determine the reorder quantity by any of several policies. When the reorder verb program is written, the specific reorder policy need not be known and a parameter slot can be inserted at the place in the program where such a policy would be implemented. The person describing the model can, therefore, specify in the model description the policy he wants implemented by filling the parameter slot with a reference to a verb that simulates the desired policy. This modular form of programming provides great flexibility and minimizes the size of verb programs.

To demonstrate how AUTASCRIPT is used to describe a model, the simple multinode system shown in Figure 2 will be used. The diagram shows a five node system. Each node is given a name or

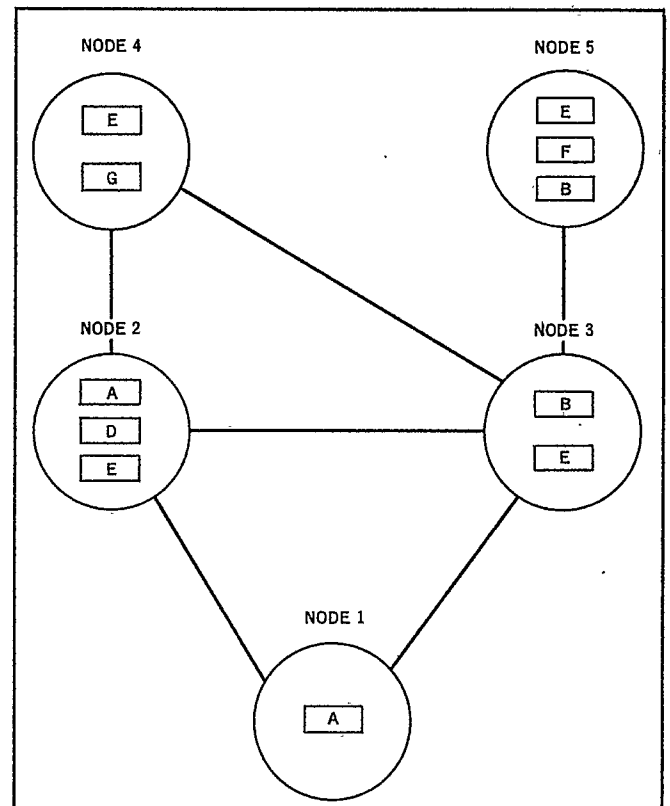

Fig. 2—Illustrative Node Network System

number of from one to five characters. Recall that each node in a model is an activity center where one or more activities are simulated. The module in NODE1 labeled A represents the main activity to be simulated there. Similarly, the modules labeled A, D, and E represent the activities to be simulated at NODE2 and so on. It is assumed for this example that the logic to simulate activity A is contained in the verb named VERBA, the logic for B is contained in VERBB, etc. The precise activities to be simulated at NODE1 and their connections with NODE2 and NODE3 are described in the model description language as shown below. Subactivities at NODE1 are represented by the logic in VERBB and VERBC.

```
NODE1.  VERBA (1 = VERBB $
              2 = VERBC (1 = VERBB) $
              3 = DELAY (P = 3), *NODE3),
        DELAY (P = 5), *NODE2 $
```

This description of NODE1 states that it contains the activity represented by VERBA; that at the point in the execution of VERBA where Parameter Slot 1 is encountered, control is transferred to VERBB; that after the execution of VERBB, control is returned to VERBA; that at the point in the execution of VERBA where Parameter Slot 2 is encountered, control is transferred to VERBC; that at the point in VERBC where Parameter Slot 1 is encountered, control is transferred to VERBB; that after the execution of VERBB, control is returned to VERBC; that after the remainder of the logic in VERBC is executed, control is returned to VERBA; that at the point in VERBA where Parameter Slot 3 is encountered, control is transferred to verb DELAY; that after verb DELAY is executed, control is transferred to NODE2; and that after the execution of the final logic of VERBA, control is transferred to verb DELAY and NODE3 in turn. Schematically, this description would appear as shown in Fig. 3. This schematic shows that VERBA contains an internal transfer in its logic which can branch around Parameter Slot 3. Thus, under certain conditions, control is transferred to NODE3 when the execution of VERBA is completed. Once control is transferred to another node, it does not return to the current node. Different delay times for these two transfers are specified through the parameters (i.e., P = 3, P = 5) of the verb DELAY which refer to two different probability distributions.

The significance of this illustration is that it demonstrates how one can describe a model including interrelations among blocks of logic with time interdependencies in relatively simple form. It also demonstrates the flexibility in varying logical procedures available to the person describing a system to be modeled. Admittedly, he must be very familiar with the system to be modeled, which is essential for any analyst designing a model, and quite intimate with the contents of the module library.

To simplify model descriptions, commonly used combinations of simple verbs can be formed into nonsimple verbs and stored on the Module Library. A nonsimple verb is a free form string of symbols

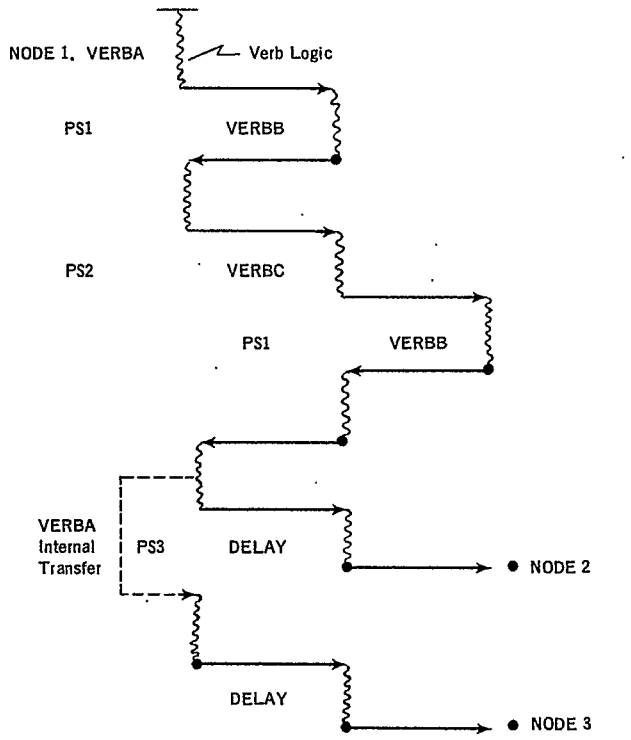

Fig. 3—Schematic Representation of Flow Control

and verb names which fixes a particular substruc- of other verbs. The earlier example of verb VERBA can be constructed as a nonsimple verb NSVA1 and can be defined as shown below. The double asterisk defines a parameter slot location in a nonsimple verb.

```
NSVA1:  VERBA (1 = VERBB $
               2 = VERBC (1 = VERBB)$
               3 = **1),
        **2
```

The key point is that NSVA1 has been defined as a particular pattern of simple verbs VERBA, VERBB, and VERBC. The description of NODE1 shown earlier can now be written more easily as follows:

```
NODE1.  NSVA1 (1 = DELAY (P = 3), *NODE2 $
               2 = DELAY (P = 5), *NODE3)
```

The reference to nonsimple verb NSVA1 causes the Model Assembler to include its structure in the model. One or more verbs used to define a nonsimple verb may themselves be nonsimple, to any depth. But every nonsimple verb must ultimately be expandable into only simple verbs, since among verbs only simple verbs may contain program statements that are executed.

MODEL ASSEMBLER PROGRAM

The Model Assembler is a computer program that, given the description of a system in AUTASCRIPT and access to the Module Library, will retrieve from the library the necessary modules,

generate linkage routines which will link them together according to the system description, and output a complete computer program of the model of the system. An additional input to the Model Assembler is a set of dimension values which are used to set the dimensions of the data storage arrays in the model program. An additional output of the Model Assembler is a list of modules and a list of data requirements and input formats for the model generated. The latter is very important, because the inputs depend on the verbs included in the model. For this reason each model may require a unique set of inputs. Listing these inputs in the required sequence and the required card formats is a great help to the model user in preparing the inputs for model execution.

The model assembly process is shown in Fig. 4. The Model Assembler scans the model description one node at a time, building a list of the verbs required. The nonsimple verbs referenced in the node are then expanded. Modules which are referenced within the verbs on the list are then added to the list. Linkage routines that provide the flow of control specified in the node description are then created. This process is repeated for each node in the description. When the complete model description has been scanned, all modules required in the model are selected from the Module Library file. A list of common data structure decks required by these modules is built and the necessary decks are retrieved from the library and their dimension values are fixed. Finally, a listing of the data requirements for the model is output.

The basic output of the model assembler is a complete simulation model program that includes (a) the functional modules referenced by name in the model description, (b) the modules which are called internally by the functional modules, (c) a set of service modules which control the simulation mechanics, and (d) a set of linkage routines which connect the functional modules in the manner designated in the model description. The modules in groups (a), (b), and (c) are retrieved from the Module Library as complete subprograms. The linkage routines in group (d) are generated by the Model Assembler in direct response to the coded model description. These are FORTRAN programs which transfer control between verb routines. In a standard model program, a specific routine would be called by name, but the AUTASIM linkage structure removes this rigidity. In the coding of a verb, a parameter slot is activated by a call to a general routine named LINK. For each reference to a verb with parameter slots specified in the model description, the Model Assembler creates a linkage routine which calls the verbs named in the parameter slots.

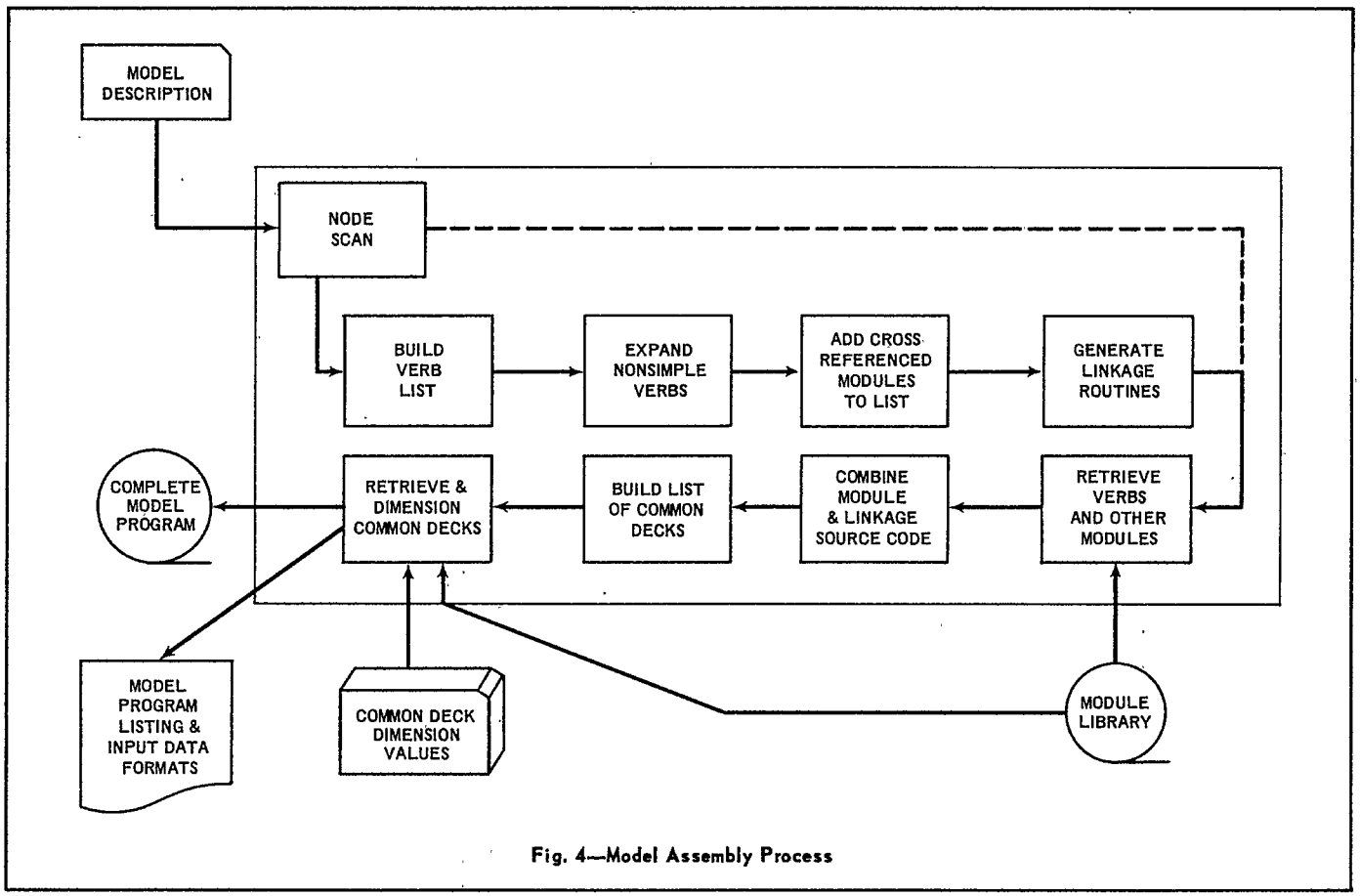

Fig. 4—Model Assembly Process

Figure 5 shows the routines created for two occurrences of the verb VERBA in a model description.

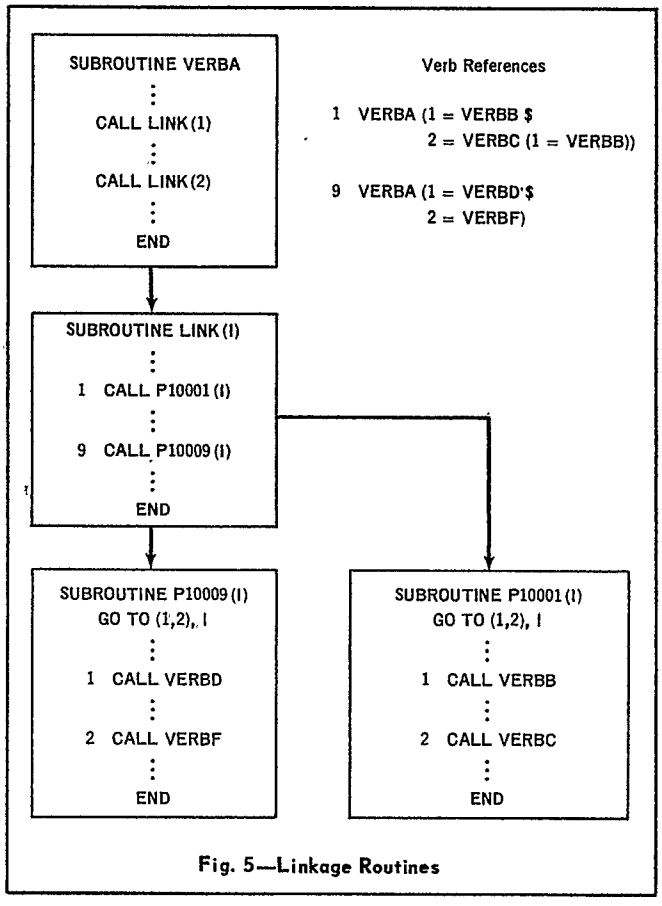

Fig. 5—Linkage Routines

These are the first and ninth occurrences of a verb with parameter slots specified in the description, and the routines are accordingly named P10001 and P10009. The logic for VERBA appears only once in the model program. When it is executed, it calls LINK which determines what linkage (i.e., P10000) routine is controlling the parameter slot contents for this reference to VERBA. The appropriate routine then transfers control to the proper verbs. A similar linkage routine is generated for each node in the model and handles the transfer of control between main verbs in the node.

MODEL PROGRAM OPERATION

The simulation mechanism of models built by the AUTASIM system is based on the GASP II (General Activity Simulation Program) structure. However, to utilize the full flexibility offered by the AUTASIM linkage system, there was a need to circumvent the fixed event code system and the FORTRAN restriction that a subroutine cannot be called recursively or be reenterable. This was achieved by an unconventional use of the assigned GO TO statement and a system of push down stacks. In AUTASIM models, verb routines are made recursively callable by setting the return address associated with a particular call with an ASSIGN statement. This address is then saved in a push down stack for use by a special return sequence which contains an assigned GO TO statement instead of the standard FORTRAN RETURN statement. The event selection in an AUTASIM model is controlled by saving data in the push down stack which identifies the node and parameter slot routines being used in the current flow path.

The data structure in AUTASIM models is a pseudo dynamic storage allocation system which circumvents many of the problems created by the fixed FORTRAN array structure. The functional data areas are divided by node in the model. Particular sets of data can then be associated with each node so that the same data retrieval operation will retrieve potentially different information depending upon which node the data retrieval was executed from. Thus the same module can be referenced in two different nodes in the model description and react differently. Even though the logic for the module appears only once in the model program, it will access different data elements based on its use in the model description.

RANGE OF APPLICATION

As noted previously, the activities which can be simulated in a model assembled by the AUTASIM system are those for which verbs are available in the library. However, in terms of the range of application of the system and basic importance to the system, the current verb families are the least significant aspect of the AUTASIM system. Figure 6 shows the AUTASIM elements in perspective over the possible range of application. The most basic part

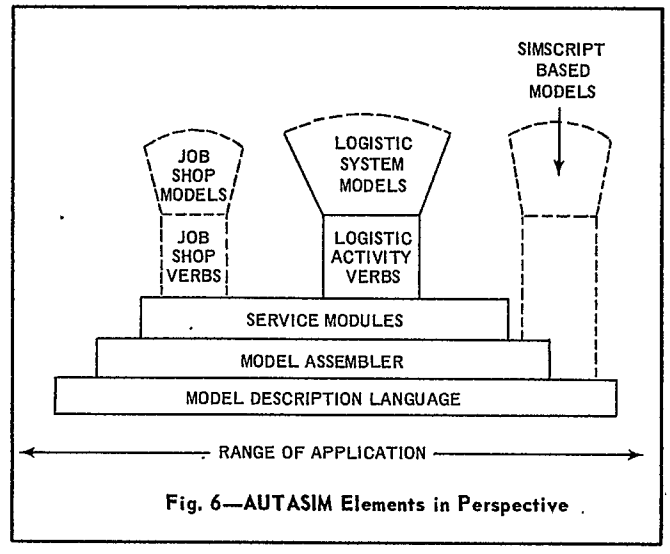

Fig. 6—AUTASIM Elements in Perspective

of the system and the one with the broadest application is the model description language. This language is an efficient procedure for describing large and extremely complex node/link structures. As such it could be adapted to entirely different types of models--such as PERT networks and certain types of LP--as well as other forms of simulation models. Similarly, the Model Assembler is more powerful than is indicated by the use made of it in the current AUTASIM system. For example, with an appropriate group of modules and a few relatively minor additions to the Assembler, it could be used to assemble SIMSCRIPT based models as indicated by the dotted box on the right side of the figure. The existing GASP based service modules provide an adequate foundation for adding new functional module groups which would enable a user

to build other varieties of simulation models, for example, job shop models or general distribution system models.

SUMMARY

The AUTASIM system is in operation and has been used to build several multifunctional, multiechelon models of Army logistic systems. The capability to rapidly create simulation model programs is attained from three components, the Model Assembler program, the Model Description Language, and the Module Library. The first two components provide the flexibility necessary to describe and assemble simulation models. The contents of the Module Library determine the range of systems which can be modeled.

ACKNOWLEDGMENTS

Two colleagues at GRC, Mr. Thomas B. Roelofs and Mr. Howard A. Markham, have made valuable contributions to this paper. I have been fortunate to work with them during the development of the AUTASIM system over the past few years. Special recognition is given to Mr. Thomas M. Lisi, who invented the model description language and programmed the initial formulation of the model assembler.