Proceedings of the 1990 Winter Simulation Conference
Osman Balci, Randall P. Sadowski, Richard E. Nance (eds.)

INTRODUCTION TO THE GENERAL SIMULATION SYSTEM GSS AND ITS APPLICATION TO COMMUNICATIONS SYSTEMS

Naeem Malik

PSI Simulation Systems Corporation
200 Atlantic Avenue
Manasquan, New Jersey 08736

ABSTRACT

The General Simulation System (GSS) offers a totally new approach to simulation. It provides a complete environment for modeling discrete event simulation, and analysis of complex dynamic systems. This system has been used for the last five years to build models, run simulations, and analyze both radio and switched communications systems for the U.S. Government and commercial communications companies. These tools provide significant time and cost savings when building large-scale models and simulations. Features include tying in hundreds of data sets containing thousands of sample points for scenario input files and data collection output files. Also included are automated facilities for parametric and sensitivity analysis, and optimizations. Additional capabilities include the multi-computer option to run a simulation on more than one computer and the real-time option to run a simulation in real-time. Graphics facilities provide for model development and documentation, run-time animation of the network and mechanisms of interest, and graphic representation of statistical results. The GSS Simulation Environment and its applications are discussed in detail in this tutorial.

1. INTRODUCTION

The General Simulation System (GSS) developed by Prediction Systems, Inc. provides a complete facility for quickly building models and simulations of general dynamic systems. Figure 1 depicts the overall structure of the GSS, showing how one can build models, run simulations and get results. GSS uses a unique approach to building and storing large quantities of reusable models. This includes a very high-level modeling language which enables the user to easily define models of the application to be simulated and to vary these models in subsequent simulation runs with minimum effort.

GSS is specifically designed to handle the complexities of very large scale real-time simulations. Applications include secure, mobile communications systems, real-time command and control systems, decision support systems, manufacturing systems, banking systems, etc. GSS has the ability to model complete physical systems, including models of the environments in which they are embedded. For example, GSS allows for specification of vehicle motion, dynamic communications links, and environmental factors, such as terrain, foliage, and radio signal interference.

The flexibility of GSS enables a wide range of applications to be simulated, from simple queuing models with stochastic inputs to large networks containing both fixed and mobile communications systems in dynamic interference environments, and simulation of human decision processes. In each case the user is assisted in building models by a sequence of prompts and simple responses. Input and output data can be stored and manipulated easily using the data management capabilities of the General Stochastic Analysis (GSA) system which supports GSS. Among other features, GSA provides for special nonstationary statistical analysis capabilities as well as sophisticated plotting and reporting facilities.

Refer to Figure 1 for an overview of GSS. To assist in building simulations, the user interacts with the **GSS Development Support System**, via a terminal, to create elements in two GSS libraries. These are the GSS **Model Library** and the GSS **Simulation Control Specification Library**. These elements may be reviewed and updated at any time using the Development Support System.

Each simulation in the Simulation Control Specification Library may draw together any number of models from the model library. This gives the system great flexibility as alternative simulations may be explored by simply changing the models used in the simulation control specification.

The Development Support System may be used to create input data files for the simulation. Alternatively, input may be created by an external system using PSI's GSA system as an option or by a previous GSS run.

A simulation is completely specified by passing a specification name to the **GSS Run-Time Support System.** Results from the simulation are displayed on the terminal or written on data files which may be analyzed by using the **GSS Post-Simulation Support System.** At all points the user is directly in control of the GSS system and may easily revise models and simulation control specifications based on the results of earlier runs.

2. BUILDING MODELS AND RUNNING SIMULATIONS USING GSS

Certain concepts are central to the GSS approach to building models of physical systems. These concepts have grown from the need to constantly change and control large scale simulations of very complex systems. The principal concept is to model all entities along physical lines, so that the boundaries are easily perceived by a newcomer to the modeling team. This is best accomplished by modeling all entities, e.g. physical systems, subsystems, equipment, staff personnel, etc., as interacting in shared environments. These environments can be electromagnetic, optical, atmospheric, acoustic, etc.

Implementing the concept of modeling along physical lines has required a distinct departure from languages normally used to support software development (computer programming languages). In fact, GSS stems from PSI's extensive experience in computer-aided design (CAD), and the systems it has developed to support electronic network design, using graphics. This has resulted in the development of a number of facilities which are used to build hierarchical structures which use these models, provide data inputs, collect data outputs, and allow the user to run multiple simulations with various changes easily.

Figure 2 depicts the basic structure of GSS modeling and simulation development facilities. Key to this structure is the ability to describe the behavior of an entity based on its own attributes, the rules that entity follows, and the attributes and rules of the environments it shares. Using GSS, hierarchical attribute structures are created using GSS Resources. Hierarchical rule structures are created using GSS Processes. These are grouped hierarchically to form models, which can contain other "submodels". To model a particular entity or environment, the modeler describes the attributes to be modeled in a set of GSS Resources. He describes the corresponding rules using GSS Processes. These processes determine how the resources change depending on the

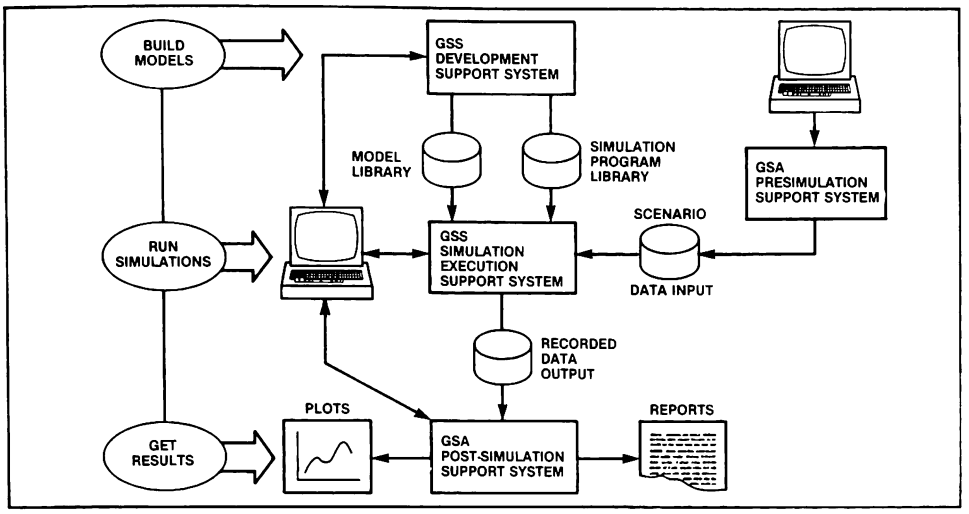

Figure 1. GSS Simulation Environment

current state of both internal and shared resources.

As an example, consider two pieces of communications equipment which interact through the electromagnetic environment. They could be sensors, radios, interference generators, etc. Each will react independently based on what each perceives from its electromagnetic environment, as well as other direct inputs to the equipments themselves. Each piece of equipment can change the environment at any time. The modeler must describe the rules which are used to put changes into the environment based upon perceived changes from one piece of equipment to another. GSS Resources and Processes are designed to easily describe the attributes and rules in an English-like language which is readable by any analyst.

To build simulations in GSS, resources and processes are created to describe the entities and environments in the system being studied. These are then linked together to form a **Model**. Using GSS, complex systems may be described using several models, each representing one part of the system. The models representing the whole system are then drawn together by a **Simulation Control Specification**. Figure 2 depicts how these various elements combine to build simulations in GSS.

This structure allows many large-scale simulation control specifications to be constructed which draw on a large number of shared models. Some simulations may differ only in the level of resolution of some of their models, or in the variations in types of entities used in the system, e.g., different radios or terminals. Figure 3 illustrates the flexibility provided by this structure.

The GSS library facilities provide a highly structured, organized approach to building and storing models which can be shared across many simulation control specifications. Separate identification of resources and processes, which can be grouped into a hierarchy of models, allows the user to work with independent entities at a high level, once the details are deemed satisfactory. Figure 4 illustrates how GSS users can build models symbolically. This example is incomplete, but is included here to demonstrate the GSS language and how it is used.

Figures 5, 6, 7, and 8 show how the GSS different language components are used to construct resource attributes, process rules, models, and simulation control specifications. These are taken as examples from a model of an aircraft flying reconnaissance missions and transmitting radio messages to mobile receivers on the ground. First the resource AIRCRAFT is described in terms of its attributes

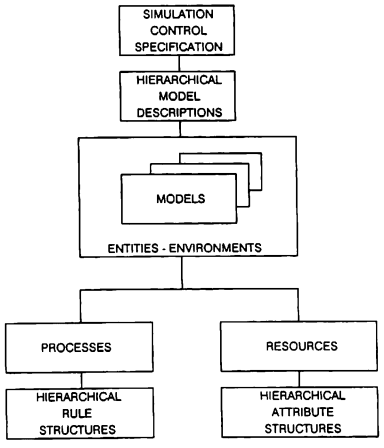

Figure 2. Modeling and Simulation Using GSS

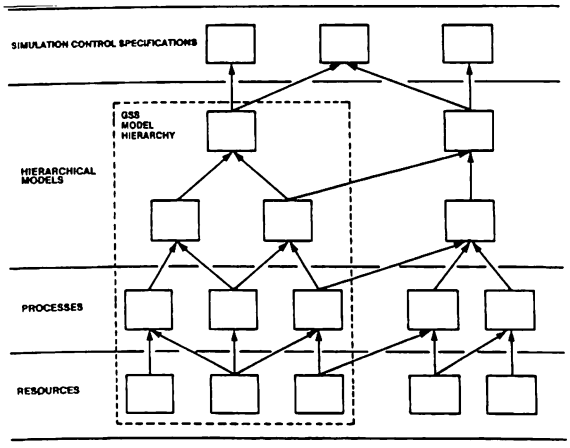

Figure 3. GSS Model Hierarchy

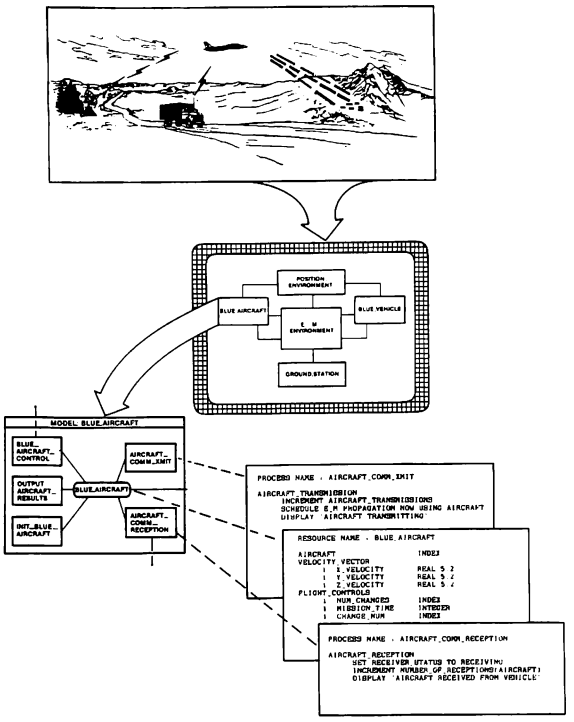

Figure 4. Example of GSS Symbolic Modeling

```
RESOURCE NAME:  AIRCRAFT

AIRCRAFT_LOCATION
     1  AIRCRAFT_X_POS                    REAL
     1  AIRCRAFT_Y_POS                    REAL
     1  ALTITUDE                          REAL
VELOCITY_VECTOR
     1  X_VELOCITY                        REAL
     1  Y_VELOCITY                        REAL
     1  Z_VELOCITY                        REAL
FLIGHT_CONTROLS
     1  NUM_CHANGES                       INDEX
     1  MISSION_TIME                      INTEGER
     1  AIRCRAFT_TRANSCEIVER_NUM          INDEX
     1  CHANGE_NUM                        INDEX
```

Figure 5. Example of a GSS Resource

```
PROCESS NAME:  BLUE_AIRCRAFT_CONTROL
RESOURCES:  AIRCRAFT
            TRANSCEIVER

FLY_BLUE_AIRCRAFT
     IF CLOCK_TIME IS GREATER THAN ZERO
          EXECUTE POSITION_UPDATE.
     IF CLOCK_TIME IS NOT GREATER THAN MISSION_TIME
          EXECUTE AIRCRAFT_TRANSMISSION
          SCHEDULE FLY_BLUE_AIRCRAFT IN 1 MINUTE.

AIRCRAFT_TRANSMISSION
     IF TRANSCEIVER(AIRCRAFT_TRANSCEIVER_NUM)
               STATUS IS IDLE
          TRANSCEIVER LOCATION(AIRCRAFT_TRANSCEIVER_NUM) =
               AIRCRAFT_LOCATION
          SCHEDULE TRANSMISSION NOW USING AIRCRAFT_
               TRANSCEIVER_NUM.

POSITION_UPDATE
     INCREMENT AIRCRAFT_X_POS BY X_VELOCITY
     INCREMENT AIRCRAFT_Y_POS BY Y_VELOCITY
     INCREMENT ALTITUDE BY Z_VELOCITY
```

Figure 6. Example of a GSS Process

(Figure 5). The process BLUE_AIRCRAFT_CONTROL (Figure 6) describes, in the GSS language, how the aircraft moves and transmits messages. The model BLUE AIRCRAFT includes this process and four others which describe the behavior of the aircraft (Figure 7). The simulation control specification in Figure 8 uses this model and other resources/processes to define the complete model to be used in the simulation.

3. THE GSS DEVELOPMENT SUPPORT SYSTEM

Figure 9 illustrates the knowledge base which GSS uses to perform simulations. It consists of three types of entities:

Hierarchical models, created by the user and written in the GSS high-level model language, including processes and resources
Simulation control specifications, written by the user in the GSS simulation control specification format
Scenario data bases, created in the GSA system or provided separately to supply input information and collect output data during the simulation

4. DESCRIBING RESOURCES

Generally, the physical attributes of a system are modeled as resources (see examples in Section 2). Specifically, a resource in GSS is defined in terms of the attributes which can be used by a process to perform a task. In the example of an aircraft flying a reconnaissance mission, the process BLUE_AIRCRAFT_CONTROL, which controls how the aircraft behaves, uses two resources, AIRCRAFT and TRANSCEIVER. These represent, respectively, physical attributes of the aircraft, and the transceiver in the aircraft which is sending and receiving messages from the ground as the aircraft moves.

Before running simulations in GSS, each of the resources in the system must be specified using the GSS language. Each resource has a name which is unique, and identifies that resource completely to GSS. (An example of a resource was shown in Figure 5, Section 2.) A resource must be constructed in terms of a hierarchical structure of **group and elementary attributes**. A set of **level numbers** is used to denote the organization of these attributes into hierarchical levels.

```
MODEL:  BLUE_AIRCRAFT

SUBMODELS
    INIT_BLUE_AIRCRAFT
    BLUE_AIRCRAFT_CONTROL
    AIRCRAFT_COMM_XMIT
    AIRCRAFT_COMM_RECEPTION

    OUTPUT_AIRCRAFT_RESULTS
```

Figure 7. Example of a GSS Model

```
SPECIFICATION NAME:  AIRCRAFT_SIMULATION

*CONTROLS
     TITLE.SIMULATION OF MOBILE RADIO TRANSMISSION
     SIMULATE
*IDENTIFICATION SECTION
     INITIALIZE_TRANSCEIVERS
     INITIALIZE_FLIGHT_PLAN
     INITIALIZE_VEHICLE
     INIT_E_M_ENVIRONMENT
*MODEL SECTION
     BLUE_AIRCRAFT
     GROUND_STATION
     VEHICLE
*EVALUATION SECTION
     OUTPUT_RESULTS
*END
```

Figure 8. Example of a GSS Simulation Control Specification

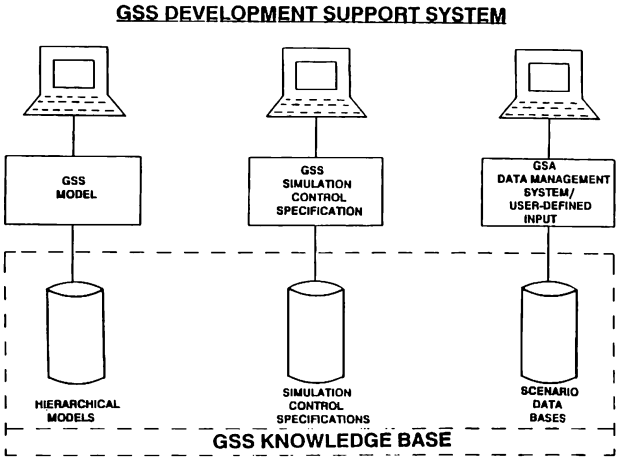

Figure 9. GSS Development Support System

```
                    GSS  RESOURCE

RESOURCE ATTRIBUTES CAN BE DEFINED IN TERMS OF SYSTEM STATES

   STORAGE:  RESOURCES CAN STORE DATA

        EXAMPLES  - MESSAGES

                  - INTELLIGENCE

                  - TARGET INFORMATION

   VALUE:  RESOURCES CAN TAKE ON VALUES

        EXAMPLES  - % UTILIZATION

                  - POSITION

                  - VELOCITY

   STATUS:  RESOURCES CAN CONTAIN ITEMS WHICH INDICATE STATUS

        EXAMPLES  - INPUT_QUEUE:  EMPTY, NOT EMPTY

                  - PRIORITY:  HIGH, MEDIUM, LOW
```

Figure 10. GSS Resource

Corresponding to each elementary level in the hierarchy is a **resource statement** describing an attribute of the resource. Resource statements first give the attribute name (which must be unique within the resource) followed by qualifying clauses which describe the attribute. Resource attributes can be described in three ways (see Figure 10) as either storing data, taking on values, or indicating status. The possible clause types for describing attributes are as follows:

STATUS clause - used to define each of the states which a status attribute may assume.
QUANTITY clause - used to define a table or vector-structured attribute which is indexed.
INDEX clause - used to define an index or pointer attribute.
INTEGER clause - used to define an integer-valued attribute.
REAL clause - used to define a real-valued attribute.
DREAL clause - used to define a double-precision real-valued attribute.
CHARACTER clause - used to define a character-oriented or alphanumeric attribute.
DECIMAL clause - used to define an attribute for use in formatted output.
ALIAS clause - allows a collective identifier to be used to represent a group of values or symbols.
REDEFINES clause - used to redefine an elementary or group attribute using a different hierarchical structure and different qualifying clauses.
INITIAL VALUE clause - allows values to be assigned to resource attributes at the beginning of a simulation run.

5. DESCRIBING PROCESSES

In any simulation there may be a number of entities which operate concurrently. In a complex simulation, it is desirable to model each of these entities independently. This is easily accomplished by describing the processes that occur in the models using special language commands and constructs. This section explains how to define the processes of a model for use in a GSS simulation. Processes must follow the rules of the GSS Process language.

5.1 Scheduling Processes

In order for models to operate concurrently during a simulation, processes can schedule and cancel other processes, or themselves, by using the GSS commands SCHEDULE and CANCEL. Processes may schedule other processes, either immediately, at some fixed time in the future, or after some time has elapsed. This "waiting period" may be specified deterministically or probabilistically (selected randomly from a specified statistical distribution). Processes may also CALL other processes to be performed directly. (This is similar to calling a subroutine in a programming environment).

If more than one process is due to start at the same time, conflicts may arise because of changing values of resources. These can be resolved by using the priority codes which GSS recognizes to insure that "simultaneous" processes are executed in the order required by the model.

5.2 Process Duration

All of the processes in a GSS simulation model are regarded as "instantaneous"; that is, they execute while the simulation clock "stands still". In such cases, any changes which the process may make to the state of a resource will be implemented directly, before advancing the simulation clock. When entities operate concurrently over finite time periods, they can be modeled by scheduling multiple processes over the time period, or by rescheduling a given process multiple times.

5.3 Process

Each process in a model must be constructed using the GSS language. Each process has a unique name which enables GSS to identify it. (Figure 6 provides an example of a process.) In addition to the name, there are a number of items of information which GSS requires in order to create a process correctly. These are as follows:

RESOURCES - a list of the resources used by the process must be given. Every attribute name used in a GSS process must be part of one of the resources in the list, and must be unique. The attributes of different resources which are used by the same process must have unique names.
INDICES - a list of indices used to indicate the passage of pointers from one process to another. For example, in a simulation of an environment containing multiple terminals, it would be necessary when scheduling a transmission to note, using an index, which terminal is to make the transmission.
TIME UNITS - a definition of the basic units of simulation time (hours, minutes, etc.) in which the process is expressed. The intrinsic units of the GSS simulation clock are SECONDS; if others are chosen here, GSS will do the conversion automatically.

After specifying these items, the process is described using a sequence of rules which, in turn, are made up of GSS statements. Each statement must follow the semantics of the GSS language.

6. DESCRIBING MODELS

Section 2 briefly described how a simulation problem could be defined using a hierarchy of models. In the simplest sense, one model can serve to describe the complete system being studied by simulation. However, the analyst using the simulation is usually faced with the practical problems of modeling the complexities of physical systems, and changing the resolution of his models to further investigate different aspects of the problem being analyzed. In this case, the system is best characterized in terms of models of its components, grouped in a hierarchical structure. Referring to Figure 3 in Section 2, the hierarchical structure serves to "push down" the complexity of the bottom layers of detailed models as they are refined. This enables the modeler to concentrate on the larger picture at the higher levels which can then be seen more clearly. The hierarchical structure also allows the modeler to easily make changes in the resolution of various models without disturbing the surrounding framework.

A GSS model is an important conceptual convenience which allows the analyst to push down complexity as he sees fit. Using GSS, the analyst can group processes together in any way he desires to form a model. However, to aid in changing models within a simulation, models are best constructed along lines of physical distinction which are naturally suited to the system being modeled.

Once a modeler has decided which processes he wants to group together to form a model, he merely names them in the **Model**. GSS models are then well defined entities. The facility also exists for the modeler to use already defined model names as "submodels" within a model currently being defined. Thus, a model consists of a list of submodels (already defined models) and processes, providing the hierarchical structure described above.

Referring to Figure 7 in Section 2, the model named BLUE_AIRCRAFT consists of a list of entities each of which may either be a process or a submodel. Whenever the model BLUE_AIRCRAFT is included as part of a simulation control specification (refer to Figure 8, Section 2), it automatically includes those models and processes listed, along with their associated resources.

6.1 Generality of GSS Models

Figure 11 provides an illustration of how models can be selected using GSS. The physical system to be simulated is represented by four processes, P_1, P_2, P_3 and P_4 with their associated resources. One representation of the system uses three models, M_1, M_2 and M_3 as shown. It is not necessary to define the resources as being part of the model, since they are associated with the processes automatically. The dotted "control" lines, depicting one process scheduling or calling another, are not pertinent to the choice of model boundary lines, except as they may influence the analyst's decision. An alternative representation would be to group P 1 and P_2 together as a model, with P_3 and P_4 grouped as another. There is no "right" way to build a GSS model: it depends on the modeler's perception of the best way to represent the elements of the physical system being modeled.

6.2 Starting Models

An important property of a model is whether or not it needs to be "started" in a simulation, by being scheduled to execute at the beginning of a simulation run. At least one model must be started in any simulation, to indicate to the GSS run-time support system how to begin the simulation run. If through the use of SCHEDULE or CALL statements, the other models will run as intended by the modeler, he need not start them. Although certain types of models must be started (these are described below), the decision to start a model generally depends on the desired representation of the physical system being modeled.

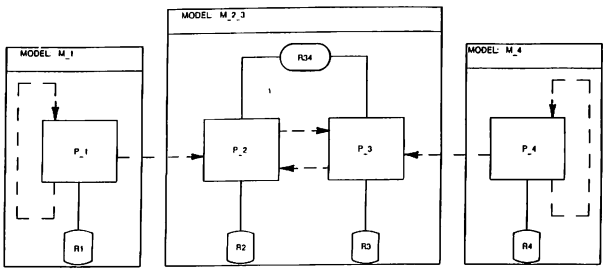

Figure 11. Selecting GSS Models

To determine which type of models must be started, consider that models have boundaries around them that include all the processes which comprise the model, reference the models shown in Figure 6. Each process in a model might schedule or call another process, or be scheduled or called by another process (shown by the dotted lines). If P_1 schedules or calls P_2, then P_2 is said to have an "inbound control path" or just an inbound path, and P_1 has an outbound path. If a model has any processes in it that have inbound paths from outside the model (i.e., there is at least one process outside the model which schedules or calls the model), then that model is said to have an inbound control path and hence referred as "DEPENDENT" model. Obviously, if a model has no inbound control paths, then it will never run in a simulation unless it is started. This includes the case where it schedules itself, (e.g. model M_1), defined as having only "self" inbound paths and hence referred as "INDEPENDENT" model.

As indicated above, models with no inbound paths, or only self inbound paths, must be started. Models M_1 and M 4 are of this category. Models with external inbound paths (e.g. model M_2_3) may or may not be started; it is purely up to the way the analyst wishes to model the physical system.

To start a model, the analyst must denote one or more of the processes in the model as a **starting process** by flagging it in the model with the letter "S" and by listing this model name in the MODEL SECTION of the Simulation. Submodels within models are always considered started so that they resolve (push) to their lowest level processes whereby the started processes are determined. This process will be scheduled to execute at the very beginning of the simulation run. If more than one starting process is flagged, all will be started while the simulation clock remains at zero, with the actual order of execution determined by the order of listing within the model.

In the example shown below, three processes are to be started, namely LOCATE_TERMINAL, INITIALIZE TERMINAL, and START_TRANSMISSION. They will be scheduled to execute in that order at the very beginning of the simulation run.

```
MODEL: BASIC_TERMINAL

SUBMODELS
        LOCATE_TERMINAL,S
        INITIALIZE_TERMINAL,S
        START_TRANSMISSION,S
        END_TRANSMISSION
        START_RECEIVE
        END_RECEIVE
```

7. GSS SIMULATION CONTROL SPECIFICATIONS

A GSS Simulation Control Specification is the top-level control which draws together all of the elements required for a specific simulation run from the GSS knowledge base which the user has created. An example was shown in Figure 8 in Section 2. The specification consists of a sequence of section titles which represent the phases of the simulation run. The DEFINITION, INPUT, MODEL, DATA REDUCTION, and EVALUATION sections can contain models and/or processes which make up the simulation. In the MODEL SECTION, only the 'INDEPENDENT' models are named.

8. MODELING ENTITIES AND ENVIRONMENTS

In simulation problems, the physical environment of an item being modeled is key to maintaining independence between models. Section 2 gave examples of the types of environments which might be modeled in GSS to accommodate this isolation. Typically, they are characterized by a coordinate system, with varying values of environmental factors (weather, terrain, RF interference, etc.) depending on location. Figure 12 depicts how environments may be shared between different models.

In GSS, environments are represented using both resources, and associated processes to model how the environment affects the behavior of the total system. For example, an electromagnetic environment surrounding three transceivers might be represented by a process such as DISTRIBUTE_POWER, and a resource E_M ENVIRONMENT as described below.

```
RESOURCE NAME: E_M_ENVIRONMENT

TRANSMITTER    INDEX
TR_LOCATION
      1 X_LOC_TR REAL
      1 Y_LOC_TR REAL
      1 H_ANT_TR REAL
TR_ANT_GAIN    REAL
RECEIVER       INDEX
RC_LOCATION
      1 X_LOC_RC REAL
      1 Y_LOC_RC REAL
      1 H_ANT_RC REAL
RC_ANT_GAIN    REAL
THRESHOLD                              REAL
TRANSCEIVER_CONNECT_TR QUANTITY(3)
      1 TRANSCEIVER_CONNECT_RC QUANTITY(3)
            2 ATTENUATION_FACTOR REAL
            2 SIGNAL_POWER       REAL
            2 NOISE_POWER        REAL
            2 SIGNAL_TO_NOISE    REAL
```

Here the attribute named TRANSCEIVER CONNECT_TR represents a 3x3 table of connectivity values for the three transceivers in the electromagnetic environment. As the radios transmit and receive, this resource holds information on the radiation in the environment at each transceiver. A special process, ENVIRONMENT_PROCESS, will use it to determine the strength of a signal transmitted from a certain location at a certain time, and received at any other specified location within the environment.

9. CONCLUSION

The General Simulation System (GSS) has proven to be an excellent tool to support analysis of both mobile and switched communications system environments.

One of the major benefits of GSS is the ability to model a system along physical equipment lines, using its English-like language. This makes it easy for communications engineers to read, understand and to make changes in the detailed specification of the models. There is no need to learn a complex programming language.

Model building in GSS follows a clear modular procedure, enabling a very complex set of models to be built from basic units which work together simultaneously in a very large simulation. Also, since GSS provides the modeler with a simulation operating system, one need only schedule processes to occur relative to each other or relative to the simulation clock. The GSS operating system takes care of the details necessary to ensure that the model is running as expected.

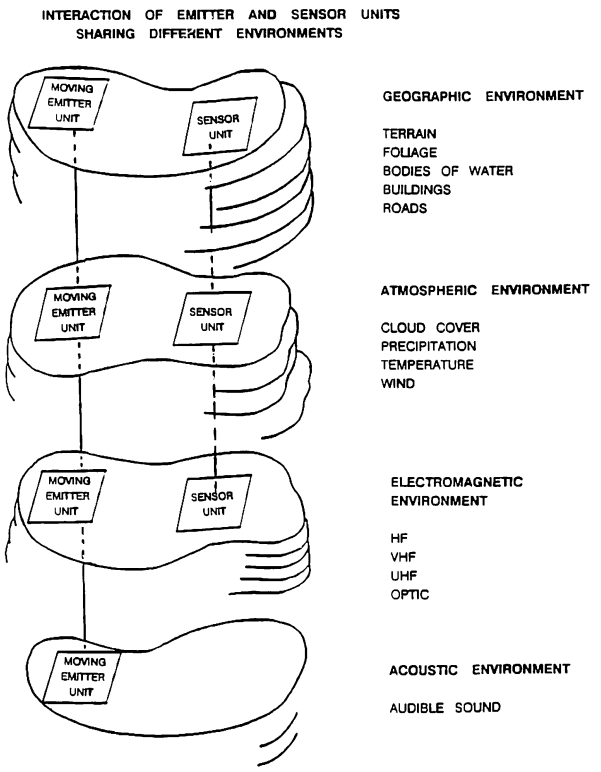

Figure 12. Illustration of how two GSS models can share multiple environments