Simulation of Discrete Systems using GPSS-FORTRAN

B. Schmidt
Institut für Mathematische Maschinen und Datenverarbeitung, Universität Erlangen-Nürnberg, Lehrstuhl IV (Betriebsysteme), Martensstraße 3, D-8520 Erlangen, FRG

The GPSS-FORTRAN simulator is a Fortran program package suitable for modelling discrete systems. It consists of a chassis, the main program, into which calls to the modules, GPSS-FORTRAN's subroutines are inserted. Its construction makes it superior in several respects to its grandparent, GPSS. Other simulation languages and tools for the analysis of discrete systems such as SIMULA, SIMSCRIPT and GASP are compared with GPSS-FORTRAN.

A SURVEY OF SIMULATION

The increasing importance of simulation has led to an ever increasing number of languages, packages and programs for the simulation of discrete systems. In the interest of conciseness we refer to these in the rest of this paper as simulators. Simulators can be divided into simulation languages, which require a special translator, e.g. GPSS, and packages, which are written in a high level programming language, e.g. GASP.

GPSS-FORTRAN is a program-package suitable for modelling discrete systems. It consists of a Fortran main program that serves as a chassis into which subroutine calls are written. The Fortran subroutines are the building blocks with whose help the system's functions are modelled. A complete description of GPSS-FORTRAN with program documentation and numerous examples is to be found in Ref. 1.

A language that offers ready-made simulation constructs should fulfill the following conditions (see also Ref. 2):

1. limited number of building-blocks;
2. fitness of the building-blocks.

The constructs must be so conceived that they suitably mimic a system's important elements and functions. The building kit should not consist of an arbitrary lump of seldom-used pieces.

For discrete systems typical, recurring elements and functions can be distinguished, they fall into three categories: Queue processing, Acquiring and freeing storage space, Coordinating tasks.

Queue processing

If a station cannot, for the moment, process newly arriving tasks, they queue up to wait for it. When the station is ready again, one of the waiting tasks must be chosen from the queue to be processed. A procedure for selecting the next waiting task is called a policy.

Systems-analytic research has shown that any complex policy can be constructed from two elementary mechanisms: preemption and priority management. GPSS-FORTRAN provides for both. Furthermore, each queue may be assigned its own specific policy, two of which are available as pre-written subroutines: 'Selection According to Priority' and 'First In, First Out'.

Acquiring and freeing storage space

A storage is a station that accommodates tasks according to its capacity and their demands for space. Each arriving task requests a certain amount of space in the storage. A warehouse or a parking lot is a storage, since each incoming item or car takes up some of the storage's available space.

The loading up of storage is called its allocation; its unloading is called freeing.

Procedures called strategies are used to decide which of a storage's locations to load or unload. When an arriving task requests space in a storage, an allocation strategy assigns the space according to a specific rule. When a storage is unloaded, a freeing strategy must decide which spaces are to be freed, if the unloading request does not specify uniquely which items are to be removed.

Any technique for managing storage can be set up, if account is kept of the occupied locations and if strategies are available to allocate and free the storage's space. Again, GPSS-FORTRAN provides for both. First, it keeps a note automatically of the positions and contents of filled storage spaces. Second, it permits each storage to be assigned its own strategies. In addition, it provides two pre-written loading strategies: first-fit and best-fit.

Coordinating tasks

The two mechanisms 'Branch on Condition' and 'Wait on Condition' suffice to coordinate the movement of tasks through a discrete system. One speaks of 'branching on condition', in case a task can be sent to one of various stations, depending on the system's state. 'Waiting on condition' means that tasks can be held at some point, until the system's state changes to meet specific conditions.

In GPSS-FORTRAN, tasks can branch or wait at any point in the model. The user specifies the condition under which branching or waiting occurs by means of a logical expression. Hence any coordination procedure can be specified without trouble.
COMPARISON OF GPSS-FORTRAN AND GPSS

GPSS-FORTRAN belongs, along with GPSS, to the higher-level simulation languages. That means both languages offer the following simulation support:

- Random-number generation
- Reporting and analysis of simulation results
- Ready-made constructs to represent particular elements and functions of systems.

The Extended Scope of GPSS-FORTRAN

GPSS-FORTRAN can be regarded as an advancement over GPSS, since it offers new language elements that make it easier to use and more widely applicable. The following points summarize its novelties:

Events. Events are now as easy to handle as transactions, where GPSS restricted itself to transactions.

Transaction-locking. In both simulators, transactions can wait at any point they choose, until the system's state satisfies certain conditions. In GPSS-FORTRAN the user can test the system's state whenever he wants, to see whether a waiting transaction can now proceed. In GPSS, on the other hand, only the simulator's flow management can test the wait conditions; they are outside the user's reach.

Queue processing. Each queue can be administered by its own policy. Furthermore, the transactions' priorities can be assigned dynamically, as well as statically.

Setup time at preemption. It is now possible to take into account the setup time lost in any preemption. A model that ignores that time yields false results, if the setup is not short by comparison with the service.

Multifacilities. A multifacility is a new type of station. It consists of several ordinary facilities operating in parallel that take transactions from a common queue.

Addressable storages. A transaction can acquire and free specific locations; the simulator keeps track of each location's contents.

Coordinating transactions. A model can coordinate its transactions' movements more easily. In particular, user chains are more broadly applicable.

Despite the differences, GPSS and GPSS-FORTRAN are close relatives. They resemble one another especially in their names: GPSS-FORTRAN's and GPSS's stations are named alike and GPSS-FORTRAN's subroutines are often named after GPSS's blocks.

The extended acceptability of GPSS-FORTRAN

The decisive disadvantage of GPSS lies in the fact that the system elements and functions are fixed and can hardly be altered or augmented. Its corset-ribs restrict this simulator's movement unacceptably. This problem becomes especially apparent in ambitious research on complex systems. Subtleties and special cases are here

the order of the day and they often cannot be modelled by uniform language constructs. GPSS's insufficient flexibility is discussed further in Refs. 4-7.

As regards flexibility, GPSS-FORTRAN knows no bounds, since it is embedded in a higher-level language. That means that changes and improvements can be built into GPSS-FORTRAN by any user with a special problem. If GPSS-FORTRAN's constructs don't suffice, the user can easily implement new constructs tailored to his needs. The user has free access: to flow control and to all of the simulator's data areas.

The extended portability of GPSS-FORTRAN

In order to limit the cost of model-construction and to avoid multiple development it should be possible to build portable models. The portability of parameter-driven simulation models is especially important (see Ref. 3).

GPSS-FORTRAN is a pure Fortran package that runs on any machine with a Fortran IV compiler. Models developed using GPSS-FORTRAN are hence largely machine-independent.

GPSS-FORTRAN COMPARED WITH SIMULA, SIMSCRIPT AND GASP

SIMULA, SIMSCRIPT and GASP belong to the low-level simulation languages; they support only flow control, random-number generation and analysis of simulation results. They don't offer however any language constructs for modelling systems' elements and functions (see also Ref. 3).

Languages like SIMULA, SIMSCRIPT and GASP are either extended general programming languages or they offer general programming constructs like those of higher-level languages. That means that they can be extended as easily as GPSS-FORTRAN.

The disadvantage of such languages is the large expense involved in building a model. This problem stems from the lack of language support. Since neither SIMULA nor SIMSCRIPT nor GASP offer macro language elements, the user is forced to program system elements and functions for himself. This process is time-consuming and error-prone.

Experience has shown that modelling in higher-level languages like GPSS-FORTRAN goes three to four times faster than in SIMULA, SIMSCRIPT or GASP. Rising software costs make the savings won by the use of higher-level languages like GPSS-FORTRAN ever more attractive.

We have discussed here only two criteria: breadth of application and ease of implementation. Simulation languages have been assessed according to other criteria in Ref. 5. For example Simula has more modern language characteristics than Fortran; Fortran is on the other hand more widespread.

SUMMARY

GPSS-FORTRAN is a higher-level language that offers ready-made constructs for modelling important system elements and functions. That facilitates modelling greatly. GPSS also has this advantage.

© Heyden & Son Ltd, 1982
Since however GPSS-FORTRAN is a Fortran program-package whose constructs can be modified and extended by the user, it is more flexible than GPSS. GPSS-FORTRAN shares with the above-named languages the advantage of a broad range of application.

In summary, GPSS-FORTRAN has the respective advantages of GPSS and the other languages, without taking on their disadvantages. This is to be seen clearly in the table taken from Ref. 5.

### Table 1. Language comparison

<table>
<thead>
<tr>
<th></th>
<th>GPSS-FORTRAN</th>
<th>GPSS</th>
<th>GASP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applicability</td>
<td>+</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Model-building</td>
<td>+</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td></td>
<td>very broad, no limitations</td>
<td>very easy, good language support</td>
<td>large expenditure, no language support</td>
</tr>
</tbody>
</table>

REFERENCES


Received December 1980

© Heyden & Son Ltd, 1982