GPSS-FORTRAN

B. Schmidt
Institut fur Mathematische Maschinen
und Datenverarbeitung
Universitat Erlangen-Nurnberg

JOHN WILEY & SONS
Chichester New York Brisbane Toronto

British Library Cataloguing in Publication Data:

Schmidt, B.
GPSS–Fortran.—(Wiley series in computing).
1. GPSS (Computer program language)
2. Digital computer simulation
3. FORTRAN (computer program language)
I. Title
001.4'24 QA76.73.G18 80-40968

ISBN 0 471 27881 5

Printed in Great Britain

GPSS-FORTRAN

PREFACE

By contrast with other modelling methods, simulation is said to give expensive results. For the time being, this reservation is justified, but improved hardware and software are reducing many costs. We thus assume that the advantages of simulation will stand out more and more and its range of application broaden.

The costs of simulation fall into three categories:

* long run times,

* high memory demand, and

* large development costs.

It is already certain that emerging technology will drastically trim the costs of computing power and memory capacity. Long run times and high memory demand will no longer bar simulation. Development costs will, on the contrary, probably rise.

One way to cut development costs is to develop a simulator composed of immediately usable, tested modules. Then, to build a model, the user need only set the finished modules together. Experience shows that such a simulator reduces costs significantly, provided: 1) the number of modules is not too large and 2) they are designed to correspond to all possibly occurring elements and functions of the system being modelled. GPSS-F was designed to satisfy both criteria.

The GPSS-F simulator is a Fortran program-package. It consists of a chassis, the main program, into which calls to the modules, GPSS-F´s subroutines, are inserted. Its construction makes it superior in several respects to its grandparent, GPSS (1):

* GPSS-F can be run on any computer with a Fortran compiler. All simulation models are virtually machine-independent.The package is already in use on a wide variety of machines, including IBM 360´s and 370´s, Cyber-series machines and PDP machines.

* Any Fortran programmer can change, extend or improve the simulator easily himself. He can tailor GPSS-F to suit his fancy, or even add new language-features required by his problem.

A beginner can learn the use of GPSS-F by either of two methods. The steam-roller method would be to read chapters 1 and 2, section 4.1 and chapters 7 and 9, ignoring under way any mention of preemption, multifacilities, storages and families.

The butterfly method browses through chapter 1 reading 1.6 carefully, then jumps to chapter 11, where it finds example models, each supplied with reading suggestions. Then it flits from examples to readings, progressing through the seven models from the simple to the ever more complex. The frequent cross-references in the text are navigation-aids for the butterfliers.

Despite the differences, GPSS and GPSS-F are close relatives.

They resemble one another especially in their names: GPSS-F´s and GPSS´s stations are named alike and GPSS-F´s subroutines are often named after GPSS´s blocks. But the younger program has features beyond older one´s. 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-F 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 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.

* Addressible 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.

I should like to thank Prof. F. Hofmann for his generous support. I also thank Prof. G. Niemeyer for allowing FGPSS (2) to be used as a basis for GPSS-F.
I received important help and support from B. Gernoth. She suggested especially valuable improvements. I thank T. Stroup for stimulating discussions that led to improvements in the system and for translating the text into English.

The program package GPSS-F is maintained by Dr. Staedtler Unternehmensberatung, Rother-Strasse 1, D 85 Nuernberg, Germany, from whom copies of the source deck may be ordered.

TABLE OF CONTENTS

1. Modelling a System. 1

1.1 Systems 1
1.1.1 Examples. 1
1.1.2 Types of Systems. 1
1.1.3 Transaction-oriented and Event-oriented Systems. 2

1.2 Models. 3
1.2.1 Similar Structures. 3
1.2.2 Types of Models 5

1.3 Simulation Languages. 7
1.3.1 Types of Languages. 7
1.3.2 Model Formulation and Range of Application. 7

1.4 The Program Package GPSS-F. 8
1.4.1 Model Construction and Range of Application for GPSS-F 8
1.4.2 A System´s Elements and Functions 9
1.4.3 Application Examples. 10

1.5 Developing a Model. 11
1.5.1 Analyzing a System. 11
1.5.2 Building a Model. 12
1.5.3 Testing a Model 14
1.5.4 Using a Model 14
1.5.5 Analyzing Results 15

1.6 Simulating Discrete Systems 15
1.6.1 Bank-Window Model 16
1.6.2 Simulation Flow 20
1.6.3 Waiting Transactions. 23
1.6.4 Conditioned Activation. 26

2. Structure of GPSS-F 29

2.1 Event Management. 29
2.1.1 The Event List. 29
2.1.2 EVENT .30

2.2 Transactions. 32
2.2.1 Transactions´ Data Area 33
2.2.2 Transactions´ States. 34

2.3 Transaction Management. 37
2.3.1 Activation List 38
2.3.2 ACTIV1. 39
2.3.3 ACTIV2. 42

2.4 Transaction Generation and Annihilation. 45
2.4.1 GENERA. 45
2.4.2 TERMIN. 47

2.5 Transaction-State Management. 49
2.5.1 ADVANV. 49
2.5.2 BUFFER. 50
2.5.3 UNLOCK. 51

2.6 The Chassis 51
2.6.1 Chassis Sections. 52
2.6.2 INIT1 55
2.6.3 INIT2 56
2.6.4 INIT3 57
2.6.5 RESET 58

3. Stations and Policies 60

3.1 Policies. .60
3.1.1 Priority and Preemption 60
3.1.2 FIFO (First In, First Out). 61
3.1.3 LIFO (Last In, First Out) 61
3.1.4 SJF (Shortest Job First). 62
3.1.5 Round-Robin (Cyclic Policy) 62
3.1.5 LFB (Limited Feedback). 64

3.2 Dynamic Priority Assignment 64
3.2.1 Types of Dynamic Priority Assignment. 64
3.2.2 UTL (Upper Time Limit). 64
3.2.3 UTLP (Upper Time Limit with Priorities) 65
3.2.4 WTLP (Waiting Time Limit with Priorities) 66

3.3 Policies in GPSS-F. 66
3.3.1 POLICY. 67
3.3.2 PFIFO 69
3.3.3 FIFO. 70
3.3.4 Dynamic Priority Assignment with Subroutine DYNVAL. 71

3.4 The State Vector. 72

3.5 The Watchdog. 73

4. Facilities. 74

4.1 Acquiring and Freeing a Facility. 75
4.1.1 SEIZE 75
4.1.2 WORK. 77
4.1.3 CLEAR 80

4.2 Preemption. 81
4.2.1 PREEMP. .81
4.2.2 Setup and Knockdown in Case of Preemption. 85
4.2.3 SETUP 86
4.2.4 KNOCKD. .87

4.3 Multiple Entries to a Facility. 90

5. Multifacilities 91

5.1 The Structure of Multifacilities. 91

5.2 Acquiring and Freeing a Multifacility .95
5.2.1 MSEIZE. .95
5.2.2 MWORK 98
5.2.3 MCLEAR. 100

5.3 Preemption at Multifacilities 102
5.3.1 MPREEM. 102
5.3.2 Setup and Knockdown in Case of Preemption. 106
5.3.3 MSETUP. 106

5.4 The Plan: Service-Element Management. 107
5.4.1 PLANI and PLANO 107
5.4.2 LFIRST. 109
5.4.3 PRIOR 109

6. Storages 111

6.1 The Structure of Storages 111

6.2 Non-Addressible Storages. 113
6.2.1 ENTER 113
6.2.2 LEAVE 116

6.3 Strategies. 117
6.3.1 Storage Acquisition 117
6.3.2 First-fit 117
6.3.3 Best-fit. 118
6.3.4 Conditioned Storage Assignment. 119
6.3.5 Segmenting. 119
6.3.6 Collecting Free Space 119
6.3.7 Freeing Space 121

6.4 Addressible Storages. 121
6.4.1 The Structure of Addressible Storages. 121
6.4.2 ALLOC 123
6.4.3 FREE. 127

6.5 GPSS-F Strategies 130
6.5.1 STRATA and STRATF 130
6.5.2 FFIT. 132
6.5.3 BFIT. 133

7. Coordinating Transactions 135

7.1 Branching on Condition. 135
7.1.1 Deterministic Branching 135
7.1.2 Stochastic Branching: TRANSF. 136

7.2 Waiting on Condition. 136
7.2.1 Trying the Gate 137
7.2.2 Local and Global Parameters 138

7.3 Gates 139
7.3.1 GATE1 139
7.3.2 GATE2 143
7.3.3 The IT Mechanism. 145

7.4 Gather Stations 146
7.4.1 Coordinating Transactions in one Processing Path 146
7.4.2 GATHR2. 146

7.5 User Chains and Trigger Stations. 148
7.5.1 Coordinating Transaction in Parallel Paths. 148
7.5.2 LINK2 150
7.5.3 UNLIN2. 152

8. Families. 155

8.1 Kindred Tasks 155

8.2 Data Areas for Families 155

8.3 The Birth and Death of Kinsmen. 156
8.3.1 SPLIT 156
8.3.2 ASSEMB. 158

8.4 Coordinating Family Members 160
8.4.1 Gather Stations for Families. 160
8.4.2 GATHR1. 160
8.4.3 User Chains and Trigger Stations for Families. 162
8.4.4 LINK1 163
8.4.5 UNLIN1. 165

9. Random Sequences. 168

9.1 Random Number Generators. 168

9.2 Uniformly Distributed Random Sequences 169

9.3 Arbitrarily Distributed Random Sequences 170
9.3.1 Using the Inverse Cumulative-Distribution Function 170
9.3.2 Exponential and Erlang Distribution. 171
9.3.3 Gauss Distribution. 174
9.3.4 Lognormal Distribution. 175
9.3.5 Approximating Empirical Distributions 175
9.3.6 Truncated Distributions 177

9.4 Random Numbers in GPSS-F. 181
9.4.1 Function RN 181
9.4.2 UNIFRM. 182
9.4.3 ERLANG. 183
9.4.4 GAUSS 184
9.4.5 LOGNOR. 185
9.4.6 BOXEXP. 186
9.4.7 Using the Random Number Generators. 187

10. Gathering Statistical Data 190

10.1 Bins. 190
10.1.1 Statistics Gathered by Bins. 190
10.1.2 ARRIVE 193
10.1.3 DEPART 194
10.1.4 ENDBIN 196

10.2 Frequency Table 197
10.2.1 TABULA 197
10.2.2 EVALUE 199
10.2.3 GRAPH. 201

10.3 Report of the System´s State. 204
10.3.1 REPORT 204
10.3.2 SMLIST 205
10.3.3 SELIST 206

10.4 Saving the System´s State 207
10.4.1 SAVE 207
10.4.2 CONT 207

10.5 Ending Stochastic Simulation. 208

11. Models 211

11.1 Flow Management 211
11.1.1 Bank Window. 211

11.1.2 The Chassis´ Sections. 211
11.1.3 The System´s State at T = 630. 214
11.1.4 Final Results. 215
11.1.5 Model Listing. 216

11.2 Random Sequences. 225
11.2.1 Squirrel 225
11.2.2 Our Use of Random Number Generators 225
11.2.3 The Model´s Construction 226
11.2.4 Model Listing. 229

11.3 Policies and Preemption 238
11.3.1 Tune-Ups 238
11.3.2 The Model´s Construction 238
11.3.3 Model Listing. 240

11.4 Dynamic Priority Assignment 250
11.4.1 Orders 250
11.4.2 The Model´s Construction 250
11.4.3 Model Listing. 252

11.5 Multifacilities 269
11.5.1 Clinic 269
11.5.2 The Model´s Construction 269
11.5.3 Model Listing. 272

11.6 Strategies. 282
11.6.1 Computer 282
11.6.2 The Model´s Construction 283
11.6.3 Startup. 283
11.6.4 Collecting and Displaying Statistics 285
11.6.5 Final Results. 285
11.6.6 Model Listing. 286

11.7 Gates 299
11.7.1 Cable Car. 299
11.7.2 The Model with Type-1 Gates. 299
11.7.3 Model Listing with Type-1 Gates. 302
11.7.4 The Model with Type-2 Gates. 309
11.7.5 Model Listing with Type-2 Gates. 309
11.7.6 The Model with User Chains 311
11.7.7 Model Listing with User Chains 311
11.7.8 Final Results. 313

Appendix 317

A1 The System´s Variables. 317

A2 Description of the Data Area. 323

A3 Parameter Adaption. 328

A4 The Chassis 334

A5 The Subroutines 340

Bibliography 519

1 M O D E L L I N G   A   S Y S T E M

Models simulate systems. More precisely, a simulation studies a system´s behaviour over some period by constructing a second system, with the same structure as the original, but easier to work with. The second system is called a model.

1.1 Systems

A system is a set of elements in some way related to one another. At any instant, it is in some particular state determined by the elements´ states and relationships. The system´s state changes when an element´s state or its relation to other elements changes.

1.1.1 Examples

* The planetary system:
Each planet has some set of private parameters, which describe its state. These could include its diameter, geological structure, surface temperature, mass and period of rotation.
But the system´s momentary state depends not only on the planets´ own states, it depends on planetary relationships as well. Those relational parameters would include the bodies´ relative positions and velocities. If we assume that the planets´ own states don´t change during our observation, the system´s state can change only in its relational parameters.

* A computer installation:
Complex systems usually have elements of various types. If a computer installation is regarded as a system, its elements can be classified as processors, memories, peripherals and jobs.
As before, the system´s parameters are both private and relational: private parameters characterize its elements; relational parameters, the relations and dependencies between them. The relational parameters specify, among other things, which job runs on which processor or which peripherals a job is using. A state change occurs when a parameter´s value changes. A relational parameter could change in that the resources allocated to a job change; a private parameter could change in that a job´s residual running time or priority changes.

1.1.2 Types of Systems

Systems theory classifies systems with a terminology that imposes rough order on the clutter of possibilities.

* Static and dynamic systems:
Static systems are not subject to change. A beam ballance in mechanical equilibrium is a static system, provided it is not

disturbed. Systems whose state can change are dynamic.

* Deterministic and Stochastic Systems:
  Dynamic systems are divided into two further classes. A system is deterministic, when for each system state, the subsequent system state is uniquely determined. If various states can follow, the system is stochastic. In plain English, chance is a part of the system. A radioactive nucleus is a stochastic system, since it emits nucleoids at random. ´At random´ means that no given state leads, without fail, to radioactive decay.

* Continuous and discrete systems:
  If state changes are continuous functions of time, as they are, say, in the planetary system, the system is continuous. A lake with streams feeding and draining it is also a continuous system. So is a spring-driven pendulum. If states change abrubtly and at intervals, the system is discrete. Two examples of discrete state changes are: a computer job frees one system resource and acquires another; the ballance in an account is increased by the amount of a deposit.

1.1.3 Transaction-oriented and Event-oriented Systems

Among the discrete systems, we distinguish the transaction-oriented from the event-oriented. Many systems can be thought of as built from mobile and stationary system elements. The mobile elements wander between the stationary ones, altering them and being altered. The mobile elements are called transactions; the stationary elements, stations. Systems so constituted are transaction-oriented.

Examples:

* Computer installation
  The stations are the system´s resources, such as processors, working storage and peripheral devices. The transactions are jobs or tasks brought into the system; they travel from station to station and they acquire, free and queue up in front of stations.

* Warehouse
  A warehouse consists of a great many racks in which wares are kept. The racks count as stations and the wares as transactions. Wares arrive at the warehouse, are filed according to muster and remain stored a while. Finally, upon receipt of an order, they are removed from the racks and forwarded.

* Street intersection
  The cars are transactions whose essential characteristic is direction-of-travel. A traffic light coordinates their movement: travel is barred where the light is red. The cars form queues at the light until it changes; then they travel on.

When a model mimics one of those systems, it is irrelevant whether its transactions represent jobs in a computer, packages in a warehouse or cars at a traffic light. The essential similarity is, they are all mobile system components. A model´s stations are just as flexible. They are equally adept at representing anything stationary, whether computer processor, warehouse rack, or traffic light.

In `transaction-oriented systems, all state changes originate in the transaction´s movements. An event-orientation doesn´t divide the world so sharply into things that move and things that stay put. Instead, it sees systems as composed of elements that may or may not be stationary, but whose interplay entitles them in any case to full-fledged systemhood. The elements undergo state changes, called events, whose origin need not have anything to do with others of the system´s elements. Obviously, a complex system may be thought of as part transaction-oriented, part event-oriented.

Examples:

* An exchange of letters
Your grandmother remembered to send you a birthday card. Since grandmothers remember such things quite on their own, they are genuine event-makers; their memories are not jogged by the movement of anything else on earth. If you wrote a pleasant reply, your activity is transaction-oriented, since it was occasioned by the birthday card´s arrival.

* Traffic light
If we view the traffic light as a subsystem within the larger system sketched above, we could see it as event-oriented. Irrespective of traffic movements, it changes from red to green and back again.

1.2 Models

Even under the best of circumstances, it may be impossible to study a system directly. In systems planning, for example, the system doesn´t yet exist. Or a system may be inaccessible, or its investigation dangerous or costly. Often, the system can´t be studied because it changes too quickly or too slowly. So the study is carried out on a second system, built for the occasion. Of two systems structured alike, with respect to the values to be studied, the one used to investigate the other is called a model.

1.2.1 Similar Structures

An example should make clear what it means to say that two systems have a common structure:

* A mechanical system
A mass M is suspended from a rigid wall by a spring with elastic constant K and shock absorber with damping factor D. (Fig. 1) If a time-dependent force F(t) acts on the mass, the equation of motion of its center of gravity is: