A. ALAN B. PRITSKER
ROBERT E. YOUNG

SIMULATION WITH GASP_ PL/I

Simulation with GASP_PL/I

A PL/I Based Continuous/ Discrete Simulation Language

A. ALAN B. PRITSKER
Purdue University

ROBERT E. YOUNG
Wayne State University

A WILEY-INTERSCIENCE PUBLICATION

JOHN WILEY & SONS, New York • London • Sydney • Toronto

Library of Congress Cataloging in Publication Data:

Pritsker, A. Alan B. 1933-
Simulation with GASP_PL/I.

"A Wiley-Interscience publication."
Bibliography: p.
Includes index.
1. GASP (Computer program language) 2. PL/I (Computer program language) I. Young, Robert E., joint author. II. Title.

QA76.73.G15P75 001.6'424 75-23182
ISBN 0-471-70046-0

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Simulation with GASP_PL/I

To Caryl, Pam, Ken, and Jeff who provide me
with the joys and pleasures of parenthood.

A. Alan B. Pritsker

To my family who "kept the faith" all these years.

Robert E. Young

PREFACE

This book provides engineers, ecologists, management scientists, and systems analysts with the information necessary to use GASP_PL/I: A Combined Continuous/Discrete PL/I Based Simulation Language. GASP_PL/I is practical and useful. Being PL/I based, it is also relatively easy to learn. The simulation concepts included within GASP_PL/I can be quickly grasped because they are presented in the familiar framework of PL/I. Since GASP_PL/I is a combined language, methods for writing discrete (next event) simulation programs or simulation programs involving continuous variables are available. The procedures for obtaining either a discrete or a continuous simulation program using GASP_PL/I are detailed in this book. Combining discrete and continuous simulations into a single program is most important. GASP_PL/I provides the conceptual basis for accomplishing this, as well as the supporting routines that facilitate the development of combined simulation programs.

The conceptual and programming bases for GASP_PL/I are presented in Chapters 1 and 2. The design of GASP_PL/I is based on the material presented in these chapters. Chapter 3 describes the GASP_PL/I subprograms and their use. Chapter 4 describes the user-required subprograms and data input formats.

Since simulation modeling is an art, it is necessary to have examples and case studies that demonstrate the use of a simulation language. For this reason, 10 complete simulation studies are included in this book. A standard format that includes a problem statement, objectives in solving the problem, special GASP_PL/I features illustrated by the example, procedures used in modeling and writing the GASP_PL/I program, and a

discussion of simulation reports is used for each example. Since the emphasis is on simulation using GASP_PL/I, the procedure section of each study is stressed. PL/I listings for all system models are presented with a listing of input data cards. Flow charts and final output reports are presented where appropriate.

Chapter 5 includes four discrete simulation examples. An inventory situation, two queueing systems, and the simulation of a network are involved in these examples. In Chapter 6 three examples of the use of GASP_PL/I for simulating systems involving continuous variables are presented. The examples involve a pilot ejection system, a gas station, and a soaking pit furnace. In these examples differential equations are used to model the system behavior. Discrete events are included in two of the examples.

In Chapter 7 two examples of advanced combined simulation are presented. A fleet of tankers that provides crude oil input to a refinery is modeled. The input to the refinery is deposited in a storage tank, which is modeled as a continuous variable. The interactions between discrete events and a continuous demand for crude oil by the refinery make this an interesting combined simulation example. The second advanced simulation example involves a system of four chemical reactors. Maintenance events that discretely change the status of the system are included in the example.

The last chapter describes how GASP_PL/I can be used to analyze Systems Dynamics models. Forrester's World Dynamics model is programmed in GASP_PL/I and analyzed. Procedures for including discrete events and those based on the status of the system are described. GASP_PL/I provides extensive capabilities for analyzing Systems Dynamics, as well as econometric models.

GASP_PL/I is a direct consequence of the advent of the GASP IV FORTRAN based simulation language (see A. A. B. Pritsker, *The GASP IV Simulation Language*, New York: John Wiley and Sons, Inc., 1974). It is our premise that a user who programs in a particular language would like to continue to do so and, in general, would prefer not to learn a new language in order to write a simulation program. On the basis of this premise, GASP_PL/I has been developed. It is intended not as a replacement for GASP IV, but as an alternative for the user who writes in PL/I.

In general, GASP_PL/I is a more powerful language than GASP IV because of the inherent properties of PL/I. However, the form and variable names of GASP_PL/I correspond directly to the ones used in GASP IV. Thus, any user of GASP IV with a knowledge of PL/I should have no conversion difficulties.

Since GASP_PL/I is based on GASP IV, we express our appreciation to all who contributed to the development of GASP IV. In particular, we

acknowledge the efforts of Dr. Nicholas R. Hurst, a codeveloper of GASP IV, and Mr. C. Elliott Sigal, a contributor to the development of GASP IV, who assisted us in the preparation of this book. We also thank Dr. Lawrence P. McNamee of the University of California at Los Angeles for his encouragement and support, without which GASP_PL/I would not exist. Lastly, we are grateful to Anne Pritsker for the many hours spent in typing, cutting, pasting, and, in general, putting the book together.

The GASP_PL/I program is maintained and supported by Pritsker & Associates, Inc., 1710 South Street, Lafayette, Indiana 47904, from whom copies of the source deck may be purchased.

A. Alan B. Pritsker
Robert E. Young

West Lafayette, Indiana
Detroit, Michigan
June 1975

CONTENTS

Chapter One

GASP_PL/I

Models, Systems, and Simulation

Simulation is a problem solving procedure for defining and analyzing a model of a system. In this book, only digital computer simulation is of interest, and a more restrictive definition is used: *simulation is the establishment of a mathematical-logical model of a system and the experimental manipulation of it on a digital computer.*

A simulation language provides the structure and the terminology to facilitate the building of simulations. GASP_PL/I is such a computer language; it helps the user to build a computer simulation program that can be both the model of a system and the analysis vehicle. Thus this program can be thought of as a model of a system and as a generator of statistical data about the model of the system. After gaining familiarity with simulation through use, this dual designation seems natural and simplifies communication.

This book documents the GASP_PL/I simulation language. It describes the philosophy and world view embodied in GASP_PL/I and emphasizes details of the language so that its structure may readily be used for building simulation programs. Since building such a program is an art, and the best way to obtain confidence in an art form is through experience, many detailed examples of the use of GASP_PL/I are presented.

The reasons for building and using simulation programs stem from the observation that a simulator is an artificial laboratory. Once a system is

modeled and programmed, experiments can be performed using the model. These experiments, or simulations, permit inferences to be drawn about systems

- Without building them, if they are only proposed systems.
- Without disturbing them, if they are operating systems that are costly or unsafe to experiment with.
- Without destroying them, if the object of an experiment is to determine their limits of stress.

In this way, simulation programs can be used for design, procedural analysis, and performance assessment (108).

Simulation can be applied to problems that are large or small, simple or complex, statistical or deterministic. It can be performed when mathematical analyses fail or cannot be performed at all and is used by various disciplines in diverse ways (19,46,98). Bibliographies (8,94) about simulation are available, and many reference and textbooks have been published on simulation methods and applications in the last 7 years (19,20, 27,29,33,46,47,85,89,91,94,95,108,121).

SIMULATION METHODOLOGY

Given that a system and its associated performance measures have been defined and that a problem is posed within the system definition, four basic tasks should be performed in a simulation project:

1. Determine that the problem requires simulation. The crucial factors are the cost, the feasibility of conducting real world experiments, and the possibility of mathematical analysis.
2. Build a model to solve the problem.
3. Write a computer program that converts the model into an operating simulation program.
4. Use the computer simulation program as an experimental device to resolve the problem.

If simulation is to be employed successfully, tools are needed to make these four tasks easier. GASP_PL/I addresses tasks (2) and (3) directly, but pertains to (1) and (4) more by its presence than by direct technical contributions. GASP_PL/I enables statistical experiments to be conducted, but it does not deal with actual experimental design. GASP_PL/I can make simulation economical and technically feasible, but it does not provide information that makes the "simulate or not" decision any easier.

ADVANTAGES OF GASP_PL/I

GASP_PL/I provides a view of the world that simplifies model building. As a philosophy, it embodies concepts that facilitate the representation of the relevant aspects of system behavior (task 2). As a programming language, GASP_PL/I gives the computer programmer a set of PL/I statements designed to carry out the most important functions in simulation programming (task 3). Modeling concepts are translated by GASP_PL/I into PL/I routines that can be easily used. This provides the link between the modeling and programming activities that is so important to a successful simulation study. Thus, when using GASP_PL/I, tasks (2) and (3) are integrated, thus greatly simplifying the design and analysis activities required of the problem solver. This simulation power accelerates the pace of going from task (1) to task (4), and in many cases allows tasks (1) through (4) to be performed repeatedly.

GASP_PL/I has seven distinct features that make it particularly attractive as a simulation language:

1. *GASP_PL/I is PL/I* based and requires no separate compiling system. It is easily maintained and can be implemented on new computing systems and on the computing systems of different manufacturers.
2. *GASP_PL/I is modular* and can be made to fit on all machines that have a PL/I compiler.
3. *GASP_PL/I is easy to learn* since the host programming language is usually known and only the simulation concepts need to be mastered. The implementation of these concepts is apparent to the user.
4. *GASP_PL/I is used for discrete, continuous, and combined modeling,* and is one of two well-documented simulation languages with this capability.*
5. *GASP_PL/I is easily modified and extended* to meet the needs of particular applications.
6. *GASP_PL/I* utilizes dynamic storage allocation.
7. *GASP_PL/I* has free-form data input.

CHARACTERISTICS OF SYSTEMS AND MODELS (68,107)

A system is a collection of items from a circumscribed sector of reality that is the object of study or interest. Therefore, a system is a relative thing. In one situation a particular collection of objects may be only a

* GASP IV is the other: see Reference 107.

small part of a larger system—a subsystem; in another situation that same collection of objects may be the primary focus of interest and would be considered as the system. The scope of every system, and of every model of a system, is determined solely by its reason for being identified and isolated. The scope of every simulation model is determined by the particular problems the model is designed to solve.

To consider the scope of a system, one must contemplate its boundaries and contents. The boundary of a system may be physical; however, it is better to think of a boundary in terms of cause and effect. Given a tentative system definition, some external factors may affect the system. If they completely govern its behavior, there is no merit in experimenting with the defined system. If they partially influence the system, there are several possibilities:

The system definition may be enlarged to include them.
They may be ignored.
They may be treated as inputs to the system.

If treated as inputs, it is assumed that the factors are functionally specified by prescribed values, tables, or equations. For example, when defining the model of a company's manufacturing system, if the sales of the company's product are considered as inputs to the manufacturing system, the model will not contain elements that influence sales. The model of the manufacturing system does not contain a cause and effect sales relation; it only includes a statistical description of historical or predicted sales, which is used as an input. In such a model of the manufacturing system, the sales organization is outside the boundaries of the "defined" system. In systems terminology, objects that are outside the boundaries of the system, but can influence it, constitute the *environment* of the system.

Thus systems are collections of mutually interacting objects that are affected by outside forces. Figure 1-1 shows such a system.

The objects within the boundaries of a system, such as people, equipment, orders, and raw materials, are called *entities*. There are many types of entities and each has various characteristics or *attributes*. Although they engage in different types of activities, entities may have a common attribute requiring that they be grouped together. Groupings of entities are called *files* in GASP_PL/I. Inserting an entity into a file implies that it has some relation with other entities in the file.

The aim of a simulation model is to reproduce the activities that the entities engage in and thereby learn something about the behavior and performance potential of the system. This is done by defining the states of the system and constructing activities that move it from state to state.