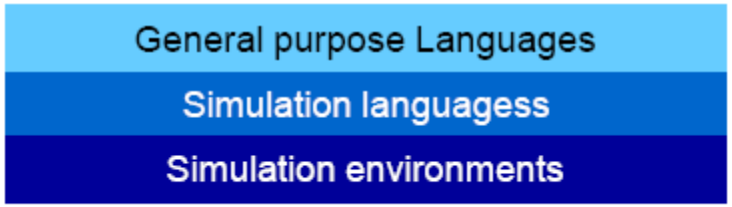
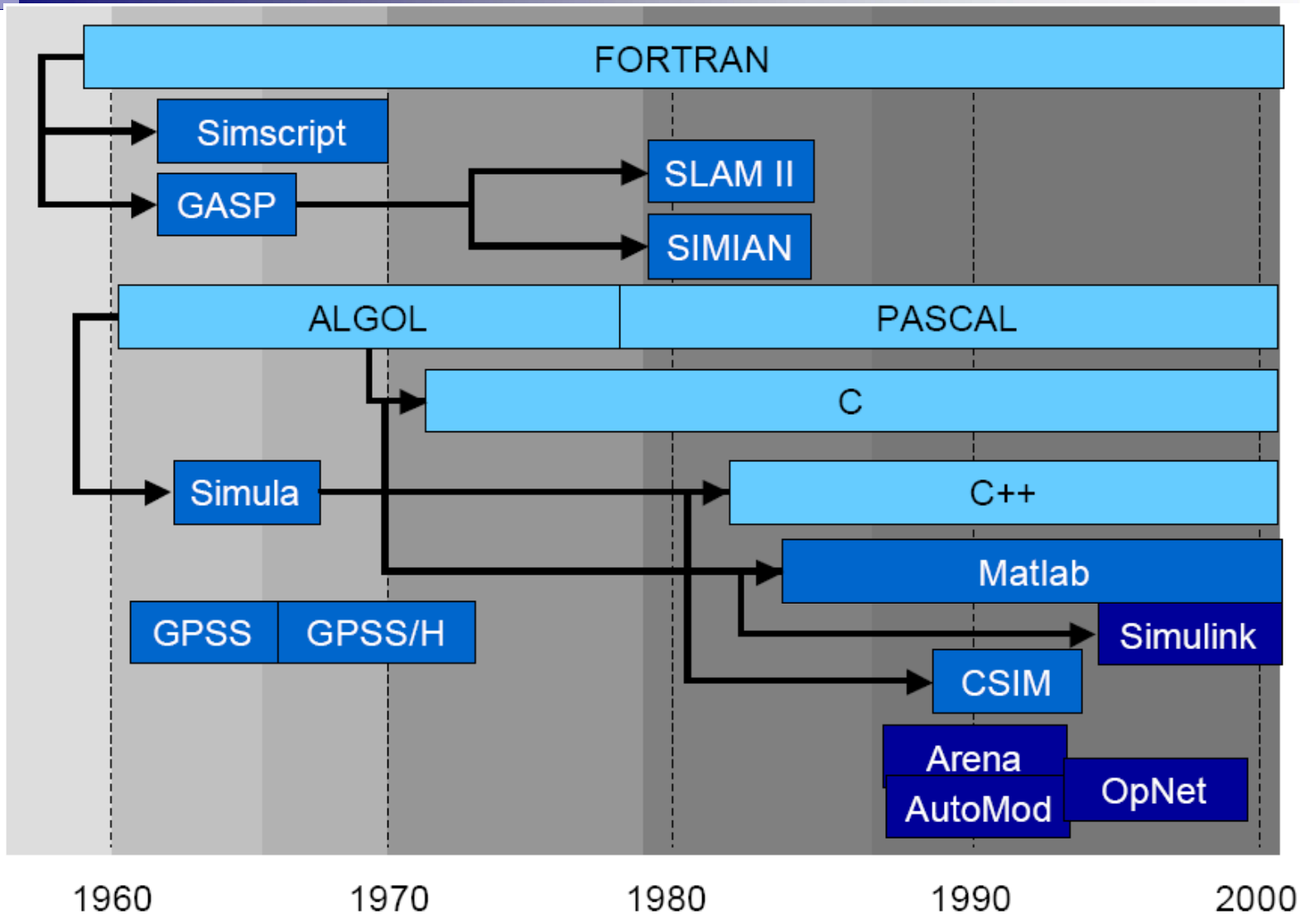
Outline and Purpose

- Discuss the history of simulation software.
- Discuss features and attributes of simulation software, organized into three categories:
  - General-purpose programming languages,
    - Flexible and familiar.
    - Well suited for learning DES principles and techniques
    - e.g., C, C++, and Java.
  - Simulation programming language,
    - e.g., GPSS/H™, SIMAN V® and SLAM II®.
  - Simulation environment
    - Good for building models quickly
    - Provide built-in features (e.g., queue structures)
    - Graphics and animation provided
    - E.g.: Arena, Automod,…



History of Simulation Software



History of Simulation Software

- 1995 - 60 The Period of Search
  - Search for unifying concepts and the development of reusable routines to facilitate simulation.
  - Mostly conducted in FORTRAN
- 1961 - 75 The Advent
  - Appearance of the forerunners of simulation programming languages (SPLs.)
  - The first process interaction SPL, GPSS was developed at IBM
- 1966 - 70 The Formative Period
  - Concepts were reviewed and refined to promote a more consistent representation of each language's worldview

Sources: Nance (1995) and panel discussion at the 1992 Winter Simulation conference (Wilson, 1992).

History of Simulation Software

- 1971 - 78 The Expansion Period
  - Major advances in GPSS came from outside IBM
  - GPSS/NORDEN, a pioneering effort that offered an interactive, visual online environment (in Norden Systems.)
  - GASP added support for the activity-scanning worldview and event-scheduling worldview (at Purdue.)
- 1979 - 86 The Period of Consolidation and Regeneration
  - Beginnings of PSLs written for, or adapted to, desktop computers and microcomputers.
  - Two major descendants of GASP appeared: SLAM II and SIMAN (provide multiple modeling perspectives and combined modeling capabilities).
- 1987 – Now The Period of Integrated Environments
  - Growth of SPLs on the personal computer and the emergence of simulation environments with graphical user interfaces, animation and other visualization tools.
  - Recent advancements have been made in web-based simulation.

Sources: Nance (1995) and panel discussion at the 1992 Winter Simulation conference (Wilson, 1992).

Selection of Simulation Software

- Advice when evaluating and selecting simulation software:
  - Consider the accuracy and level of detail obtainable, ease of learning, vendor support, and applicability to your applications.
  - Execution speed is important.
  - Beware of advertising claims and demonstrations.
  - Ask the vendor to solve a small version of your problem.

Selection simulation Software

- Model building feature
- Runtime environment
- Animation of layout features
- Output features
- Vendor support and product documentation

Model building feature

- Modeling world-view
- Input data analysis capability
- Graphical model building
- Conditional routing
- Simulation programming
- Syntax
- Input flexibility
- Modeling conciseness
- Randomness
- Specialized components and templates
- User-built objects
- Interface with general programming language

Runtime environment

- Execution Speed
- Model size; number of variables and attributes
- Interactive debugger
- Model status and statistics

Animation of layout features

- Type of animation
- Import drawing and objects file
- Dimension
- Movement
- Quality of motion
- Libraries of common objects
- Navigation
- Views
- Display step
- Selectable objects
- Hardware requirments

Output features

- Optimization
- Standardized Report
- Statistical Analysis
- Business Graphic
- File Export
  - Database

Vendor support and product documentation

- Training
- Documentation
- Help system
- Tutorials
- Support
- Upgrades, maintenance
- Track report

Selection of Simulation Software

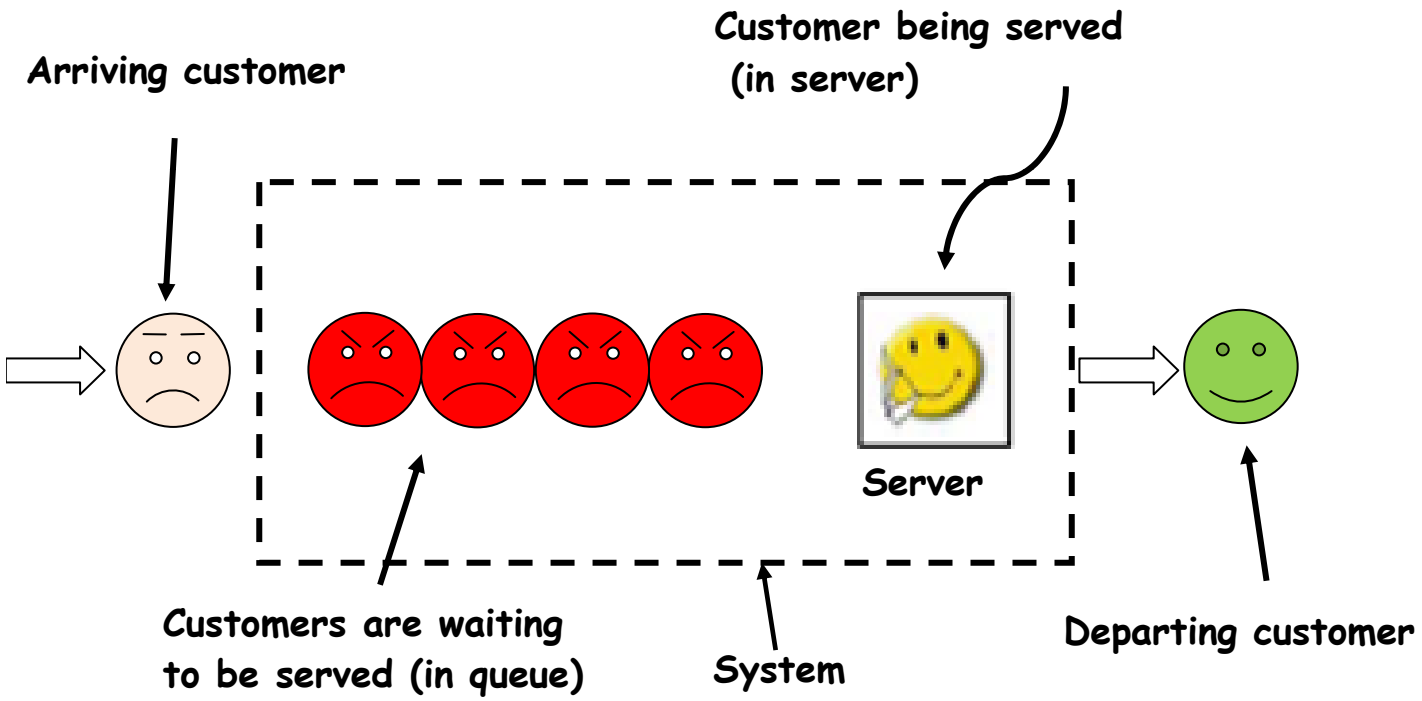
- Advice when evaluating and selecting simulation software:
  - Beware of “checklists” with “yes” and “no” as the entries, e.g. many packages claim to have a conveyor entity, however, implementations have considerable variation and level of fidelity.
  - Determine whether the simulation package and language are sufficiently powerful to avoid having to write logic in any external language.
  - Beware of “no programming required,” unless either the package is a near-perfect fit to your problem domain, or programming is possible with the supplied blocks, nodes, or process-flow diagram.

An Example Simulation

- The checkout counter: a typical single-server queue
  - The simulation will run until 1000 customers have been served.
  - Interarrival times of customers ~ Exp(4.5 min).
  - Service times are (approx.) ~ Normal(3.2 min, 0.6 min).
  - When the cashier is busy, a queue forms with no customers turned away.
  - Manual simulation in Examples 3.3 and 3.4.
  - Two events: the arrival and departure events (logic illustrated in Figures 3.5 and 3.6.)

- This example is used to illustrate simulations in Java, GPSS/H and SSF in the following slides.

Global View


Arriving customer
Customer being served
(in server)
Server
Customers are waiting
to be served (in queue)
System
Departing customer

Event-scheduling/time-advance algorithm

Clock	System State	…	Future Event List	…
t	(5,1,6)		$(3,t_1)$ $(1,t_2)$ $(1,t_3)$ … $(2,t_n)$	

1. Remove event notice for imminent event (at $t=t_1$)
2. Advance CLOCK to imminent event time
3. Execute imminent event
   Update system state, change entity attributes, set membership, as needed
4. Generate future events, if needed, and place them on FEL, ranked by time of occurrence
5. Update cumulative statistics and counters

Clock	System State	…	Future Event List	…
t_1	(5,1,5)		$(1,t_2)$ $(4,t^*)$ $(1,t_3)$ … $(2,t_n)$	

Simulation in Java

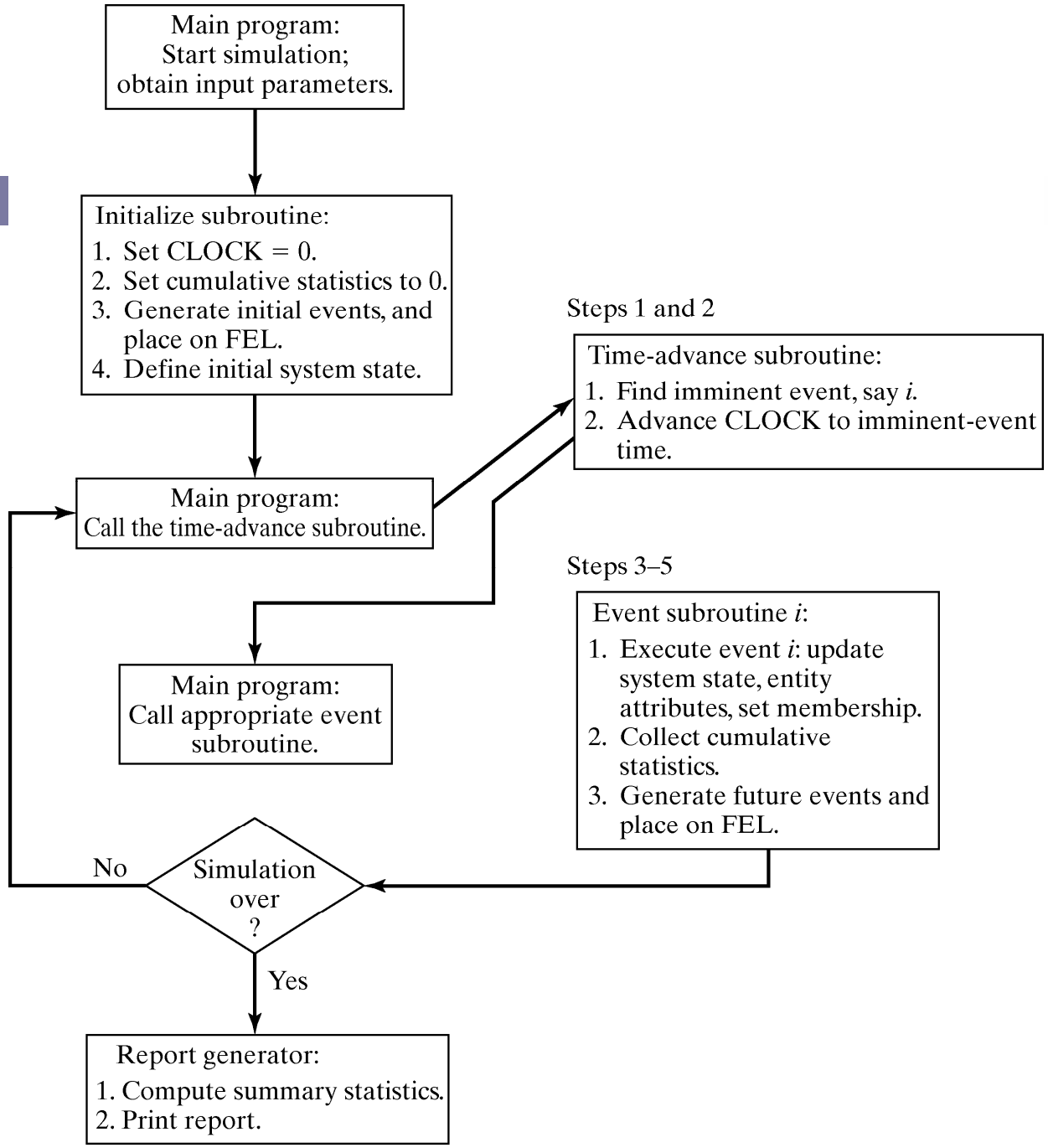
- Java is widely used programming language that has been used extensively in simulation.
- It does not provide any facilities directly aimed at aiding the simulation analyst.
- The runtime library provides a random-number generator.
- It supports modular construction of large models.
- Simulation libraries such as SSG alleviate the development burden.
  - Provides access to standardized simulation functionality and hide low-level scheduling minutiae.

Simulation in Java

- Discrete-event simulation model written in Java contains the following :
  - Basic components:
    - System state
    - Entities and attributes
    - Sets
    - Events
    - Activities
    - Delays
  - Common components (when organizing model in a modular fashion by using methods):
    - Clock
    - Initialization method
    - Min-time event method
    - Event methods
    - Random-variate generators
    - Main program
    - Report generator.

Simulation in Java:

- The overall structure of Java simulation is:



Simulation in GPSS

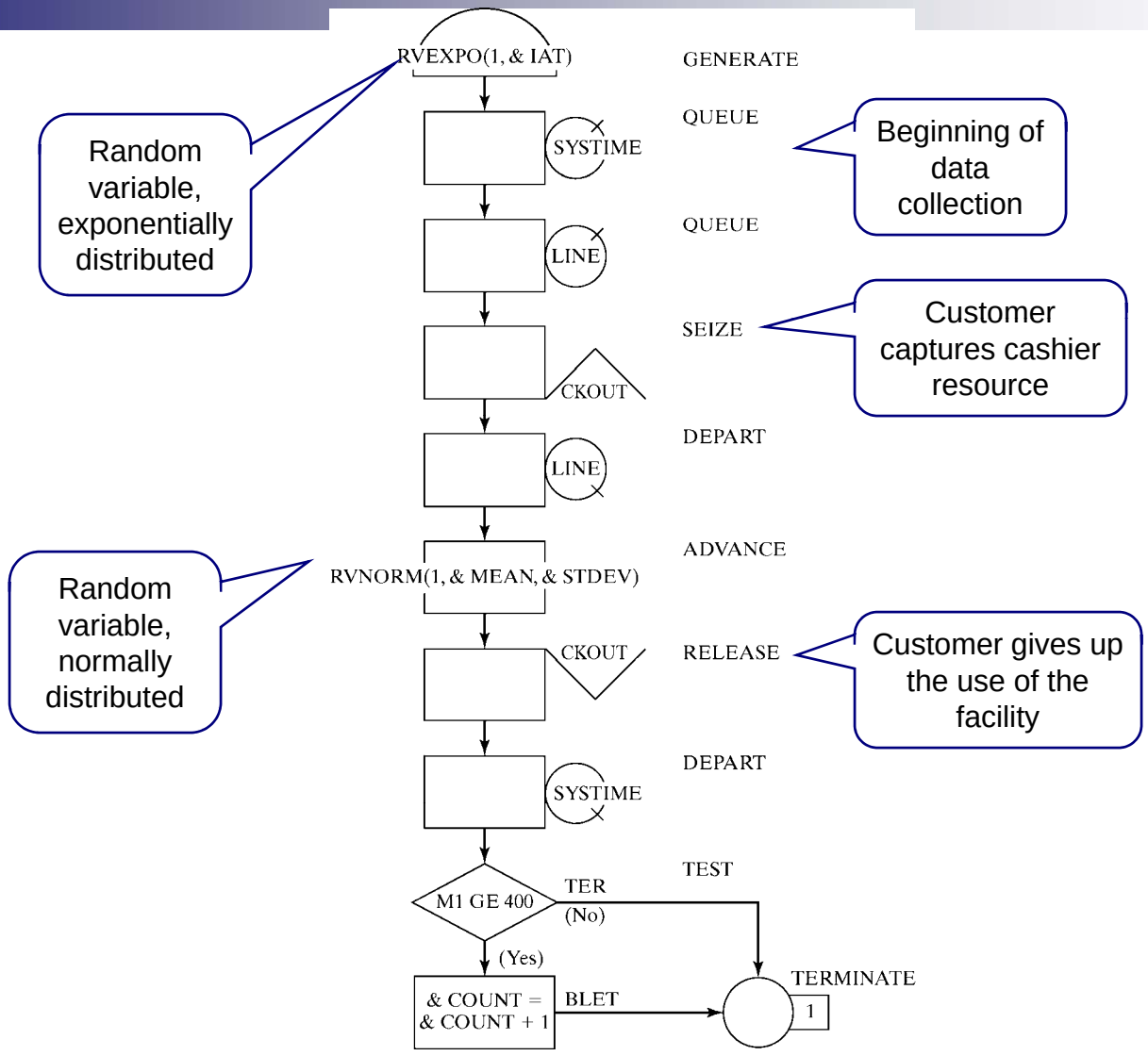
- GPSS is a highly structured, special-purpose simulation programming language.
  - Based on the process-interaction approach.
  - Oriented toward queueing systems.
- Use of block diagram:
  - Provides a convenient way to describe the system.
  - With over 40 standard blocks.
  - Blocks represents events, delays and other actions that affect transaction flow.
- Block diagram is converted to block statements, control statements are added, and result in a GPSS model.

Simulation in GPSS

- The 1st version was released by IBM in 1961.
- GPSS/H is the most widely used version today.
  - Released in 1977
  - Flexible yet powerful.
  - The animator is Proof Animation™.

Single-Server Queue Example

[Simulation in GPSS/H]



Single-Server Queue Example

[Simulation in GPSS/H]

- First, define ampervariables.

```
 SIMULATE
*
*      Define Ampervariables
*
       INTEGER        &LIMIT
       REAL           &IAT,&MEAN,&STDEV,&COUNT
       LET            &IAT=4.5
       LET            &MEAN=3.2
       LET            &STDEV=.6
       LET            &LIMIT=1000
```

Trends in Simulation Packages

- High-fidelity simulation
  - High-accuracy simulation of complex systems
- Data exchange standards
  - Simulation input/output can be interfaced to other packages
- Distributed (client/server) computing support
  - Large organization/wide-area collaboration (e.g., across LAN, Internet)
- General purpose simulations vs. specialized simulations
  - Do it once, make it reusable
- Richer object libraries/reusable block sets
- Multiple computer simulations to accelerate simulations

Implementation Directions

- Top Down
  - Define high level structure first, fill in details
  - Nothing is working until the details are done
- Bottom Up
  - Define the details first, stitch them together
  - Interfaces will change as more details are defined
- Straight through
  - Start at system input, progress through to final output (or vice versa)
- Outside In
  - Front and back interfaces are defined first, interior details later, meet in middle
  - Pieces may not join at the center properly
- Inside Out
  - Inner connections are completed, outer pieces are added
  - There is something to test from the beginning

Simulation Software (Not discussed in the book)

- OpNet Modeler/IT Guru
  - graphical modeling of complex networks
- Matlab/SIMULINK
  - block diagram focus
  - focus on scientific/technical applications
  - rich set of Blocksets/Toolboxes
- MathCAD
  - equation-based worksheets
  - includes symbolic programming (e.g., simplification/expansion of equations)

Simulation Software cntd.

- Software package discussed:
  - Arena
  - AutoMod
  - Delmia/QUEST
  - Extend
  - Flexsim
  - Micro Saint
  - ProModel
  - Simul8
  - WITNESS

Experimentation and Statistical-Analysis Tools

- Virtually all simulation packages offer support for statistical analysis of simulation output.
- In recent years, many packages have added optimization as one of the analysis tools.
  - Optimization is used to find a “near-optimal” solution.
  - User must define an objective or fitness function, e.g. cost.
  - Recent advances in the field of metaheuristics has offered new approaches to simulation optimization.
- Products discussed:
  - Arena’s Output and Process Analyzer
  - AutoStat
  - OptQuest
  - SimRunner

Arena's Output and Process Analyzer

[Experimental and Analysis Tools]

- Output Analyzer
  - Provides confidence intervals, comparison of multiple systems, and warm-up determination to reduce initial condition bias.
- Process Analyzer
  - Adds sophisticated scenario-management capabilities to Arena for comprehensive design of experiments.
  - Allows a user to define scenarios, make the desired runs, and analyze the results.
- OptQuest is used for optimization.

Summary

- Three types of software for simulation models developments:
  - General-purpose programming languages, e.g., Java, C.
    - Not specifically designed for use in simulation.
    - Simulation libraries, e.g., SSF, are sometimes available for standardized simulation functionality.
    - Helps to understand the basic concepts and algorithms.
  - Simulation programming languages, e.g., GPSS/H™, SIMAN V® and SLAM II®.
    - Designed specifically for simulation of certain systems, e.g. queueing systems.
  - Simulation environment, e.g., Arena, AutoMod.
    - Output analyzer is an important component, e.g. experimental design, statistical analysis.
    - Many packages offer optimization tools as well.