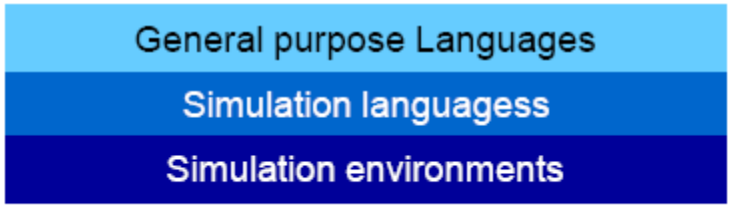
Chapter 4
Simulation Software

Banks, Carson, Nelson & Nicol

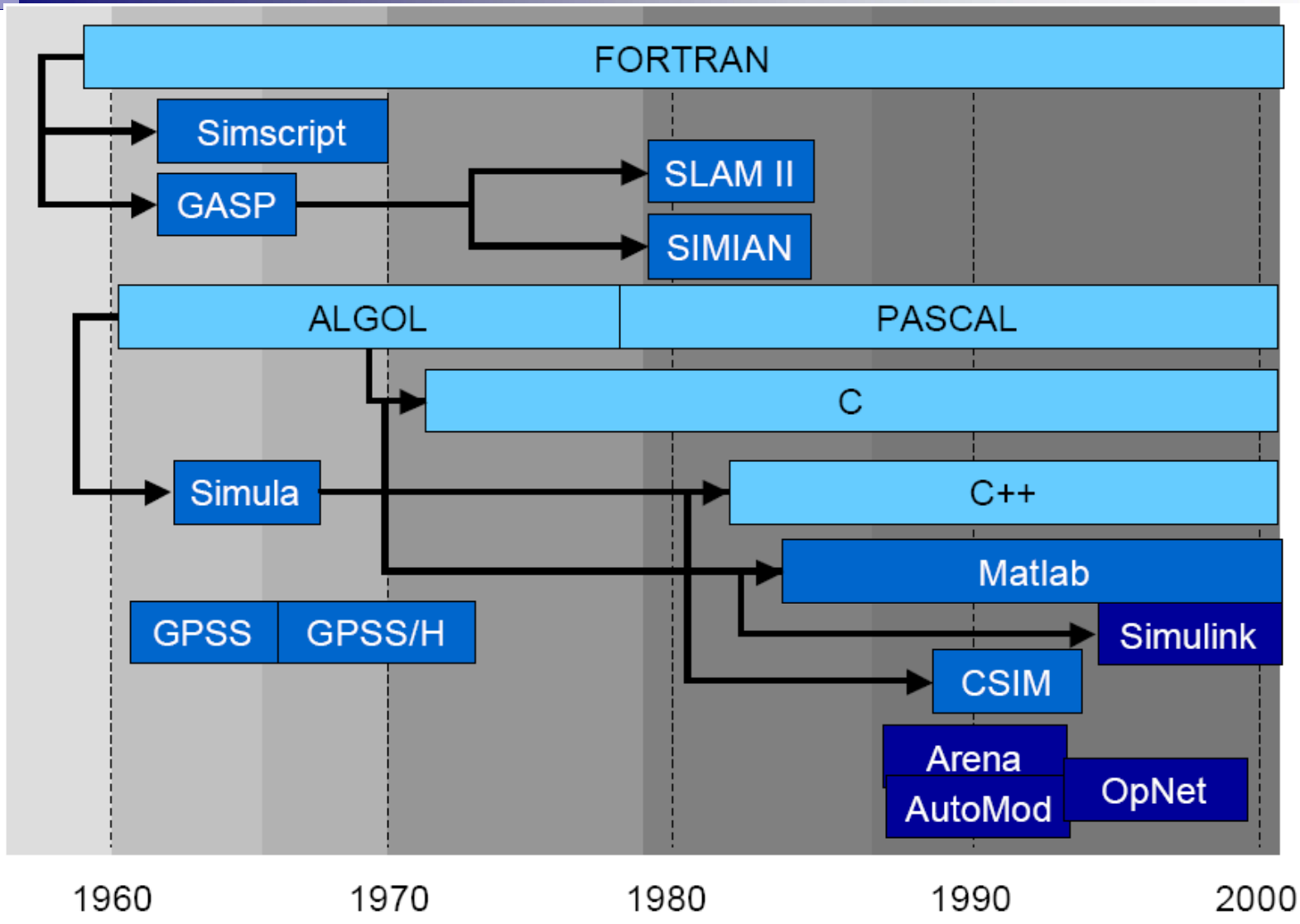
Discrete-Event System Simulation

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

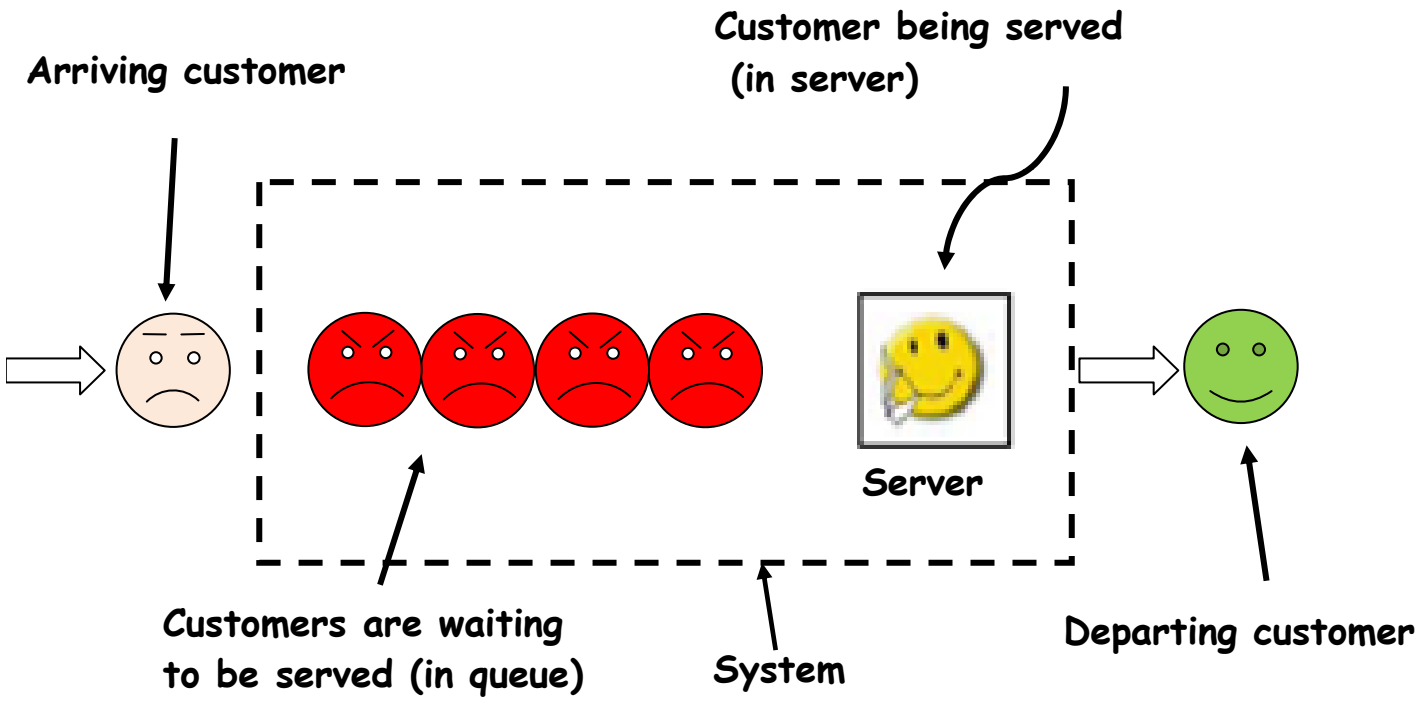
- Advice when evaluating and selecting simulation software:
  - Beware of “checklists” with “yes” and “no” as the entries, e.g. many packages claim to have a conveyor entity, however, implementations have considerable variation and level of fidelity.
  - Determine whether the simulation package and language 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

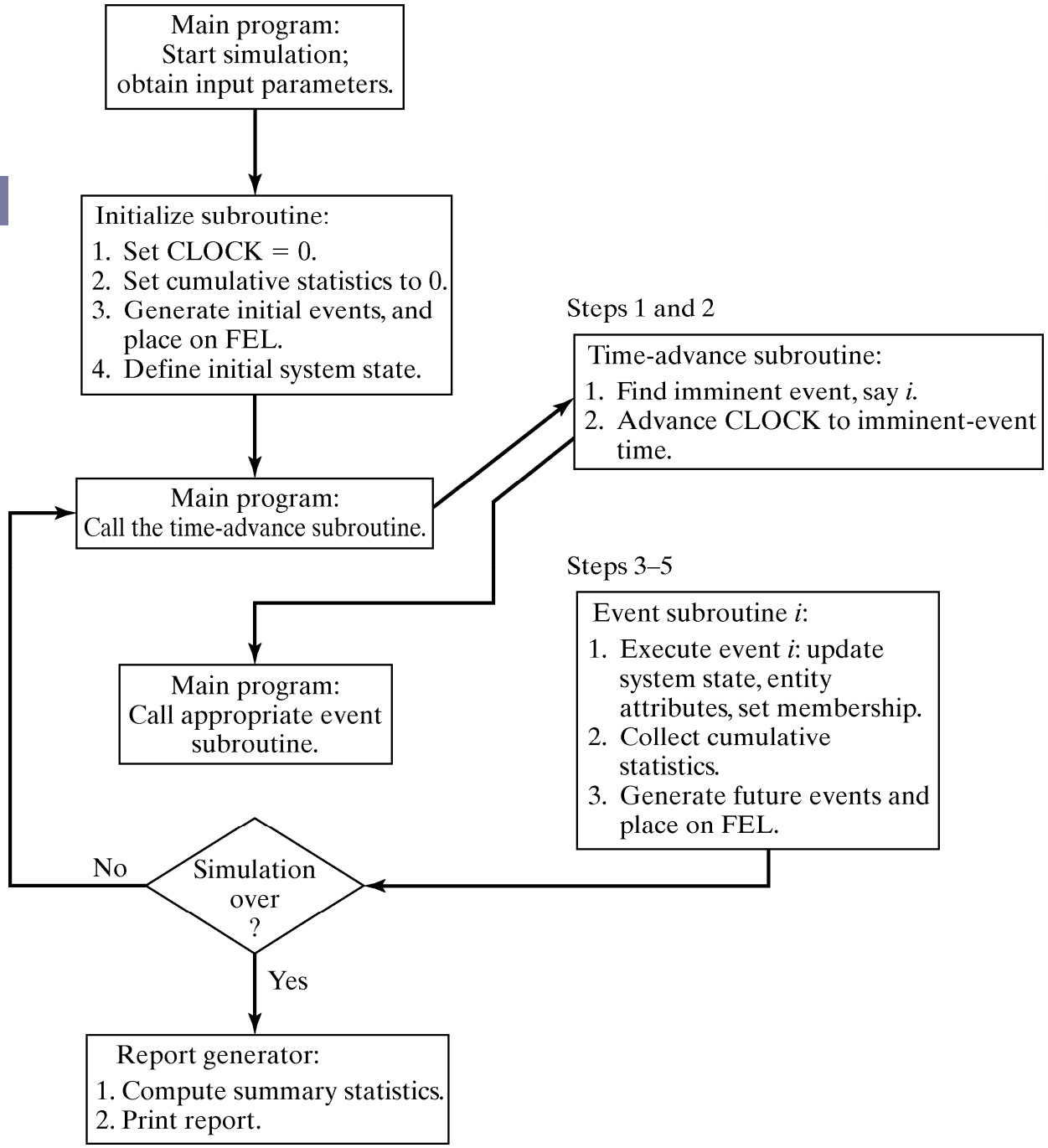
- Java is widely used programming language that has been used extensively in simulation.
- It does not provide any facilities directly aimed at aiding the simulation analyst.
- The runtime library provides a random-number generator.
- It supports modular construction of large models.
- Simulation libraries 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:

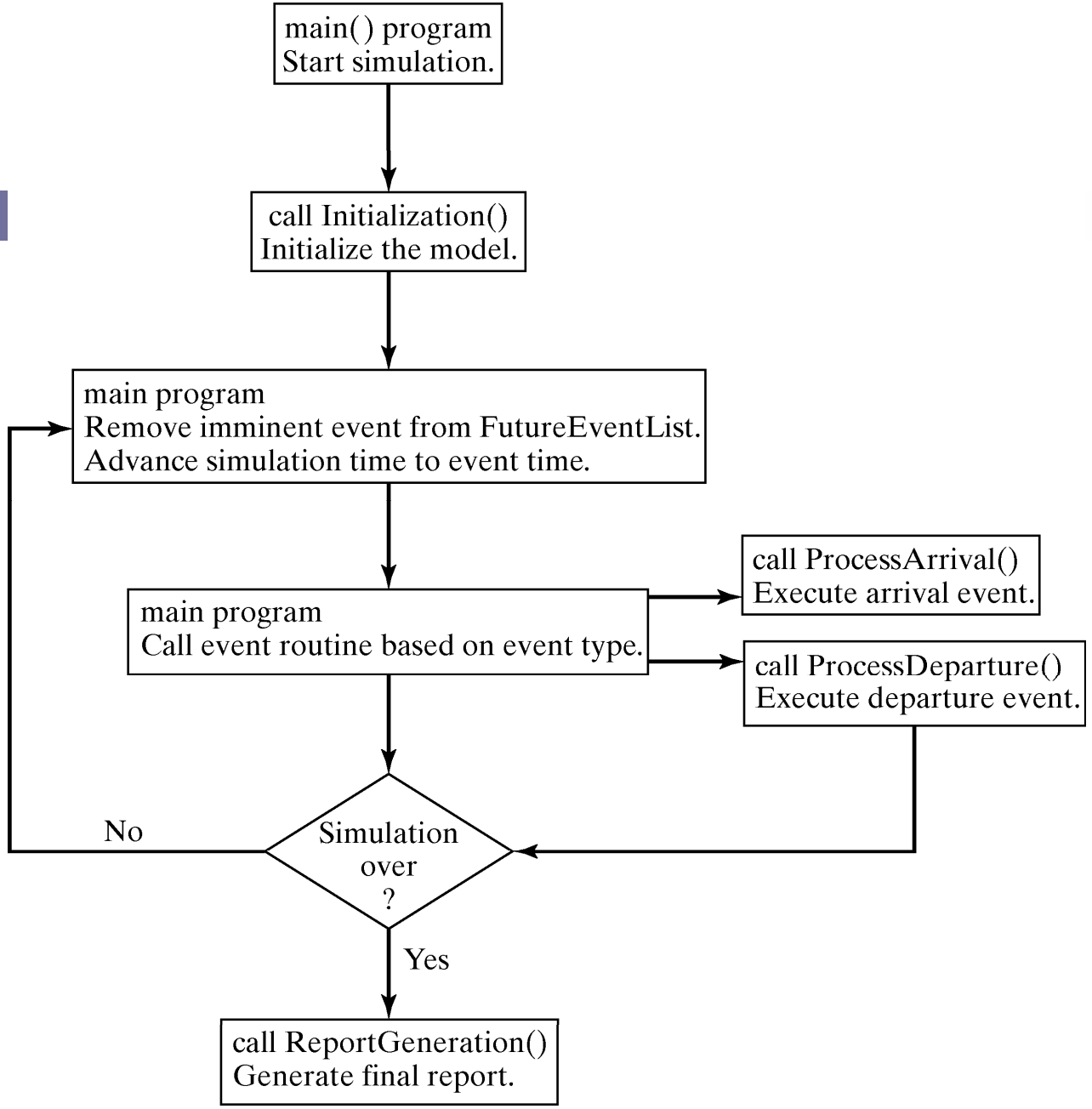
- The overall structure of Java simulation is:



Single-Server Queue Example :

[Simulation in Java]

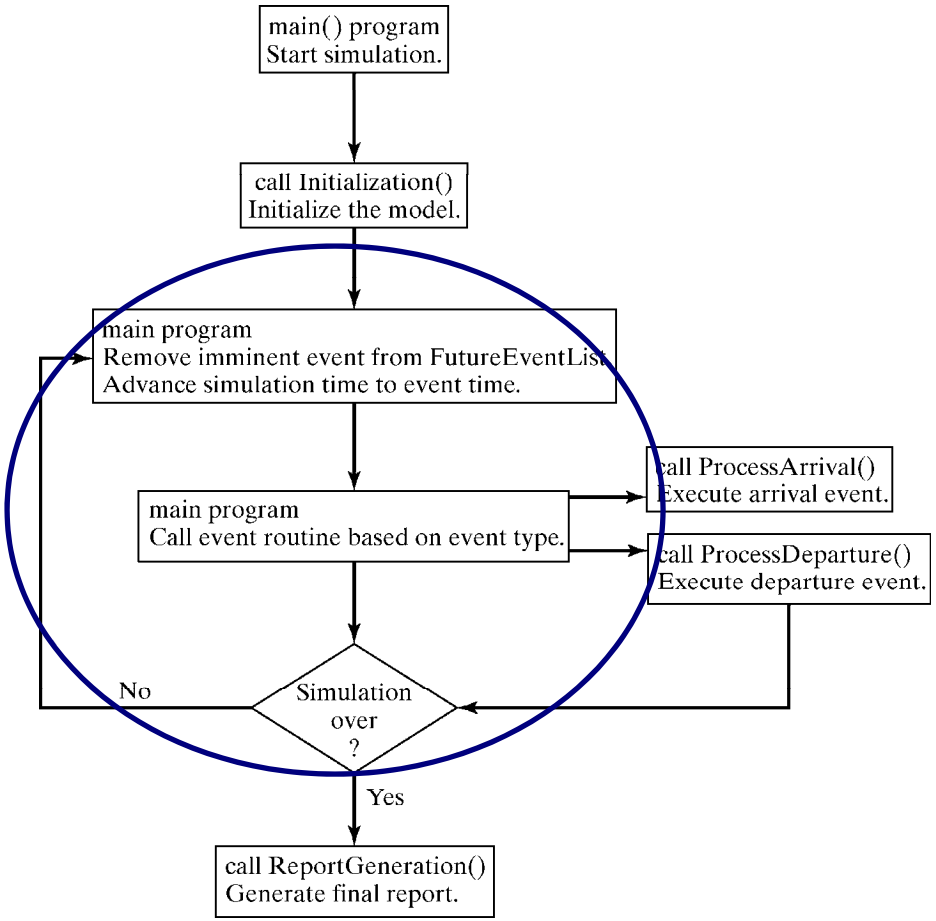
- The overall structure of Java simulation structure of the grocery checkout counter example:



Single-Server Queue Example

[Simulation in Java]

- The main program:



The Checkout Counter: Variables

System state	***QueueLength ,NumberInService***
Entity attributes and set	***Customers*** FCFS queue of customers
Future event List	***FutureEventList***
Activity durations	***MeanInterArrivalTime, MeanServiceTime***
Input parameters	***MeanInterArrivalTime, MeanServiceTime, SIGMA*** standard deviation, ***TotalCustomers*** (The stopping criterion)
Simulation variables	***Clock***
Statistical accumulators	***LastEventTime ,TotalBusy, Max QueueLength, SumResponseTime, NumberOfDepartures ,LongService*** who spends 4 or more minutes
Summary statistics	***RHO*=*BusyTime/Clock*** Proportion of time server is busy ***AVGR*** average response time ,***PC4*** proportion of customers who spent 4 or more minutes

The Checkout Counter: Functions and Methods

Functions	exponential (*mu*) normal (*mu,SIGMA*)
Methods	Initialization ProcessArrival ProcessDeparture ReportGeneration

Single-Server Queue Example

[Simulation in Java]

- Structure of the main program:

```
class Sim {

// Class Sim variables
public static double Clock, MeanInterArrivalTime, MeanServiceTime, SIGMA,
LastEventTime,
       TotalBusy, MaxQueueLength, SumResponseTime;
public static long  NumberOfCustomers, QueueLength, NumberInService,
       TotalCustomers, NumberOfDepartures, LongService;

public final static int arrival = 1;
public final static int departure = 2;

public static EventList FutureEventList;
public static Queue Customers;
public static Random stream;
```

… continued on next slide …

Single-Server Queue Example

[Simulation in Java]

- Structure of the main program (continued):

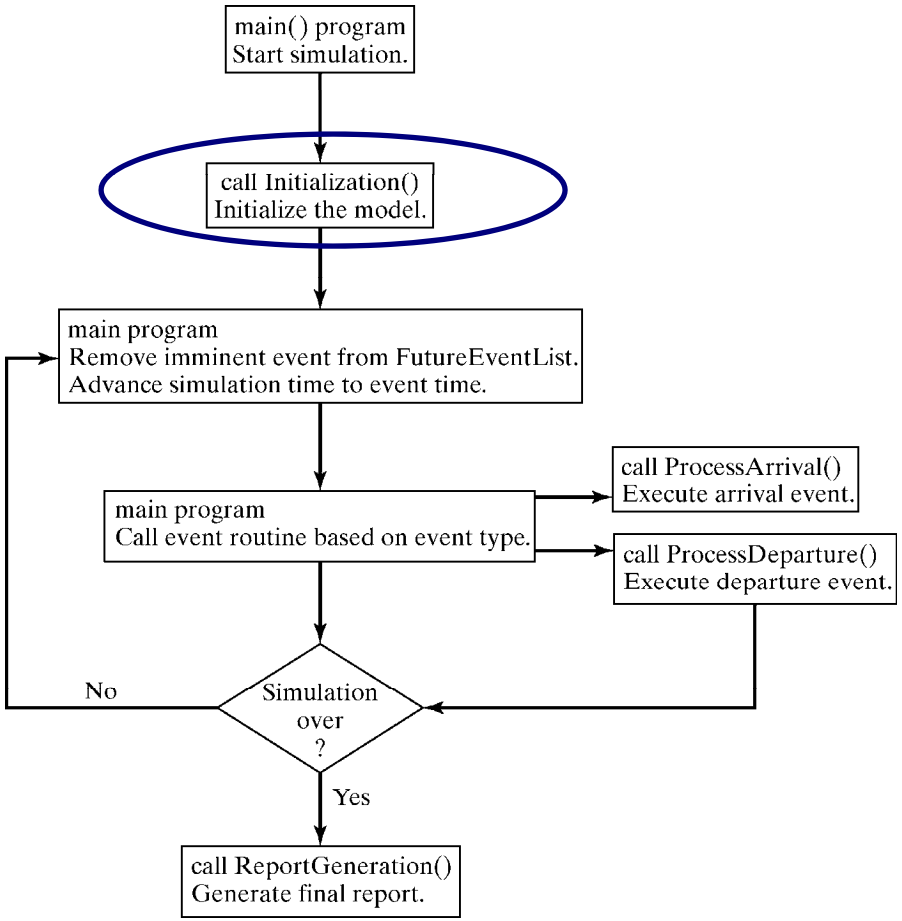
… continued from last slide …

```
public static void main(String argv[]) {

 MeanInterArrivalTime = 4.5; MeanServiceTime = 3.2;
 SIGMA              = 0.6; TotalCustomers  = 1000;
 long seed          =1000;  //Long.parseLong(argv[0]);

 stream = new Random(seed);          // initialize rng stream
 FutureEventList = new EventList();
 Customers = new Queue();

 Initialization();
// Loop until first "TotalCustomers" have departed
 while(NumberOfDepartures < TotalCustomers ) {
   Event evt = (Event)FutureEventList.getMin();  // get imminent event
   FutureEventList.dequeue();                    // be rid of it
   Clock = evt.get_time();                      // advance simulation time
   if( evt.get_type() == arrival ) ProcessArrival(evt);
   else  ProcessDeparture(evt);
   }
 ReportGeneration();
}
```

Single-Server Queue Example

[Simulation in Java]

- The initialization method:



Single-Server Queue Example

[Simulation in Java]

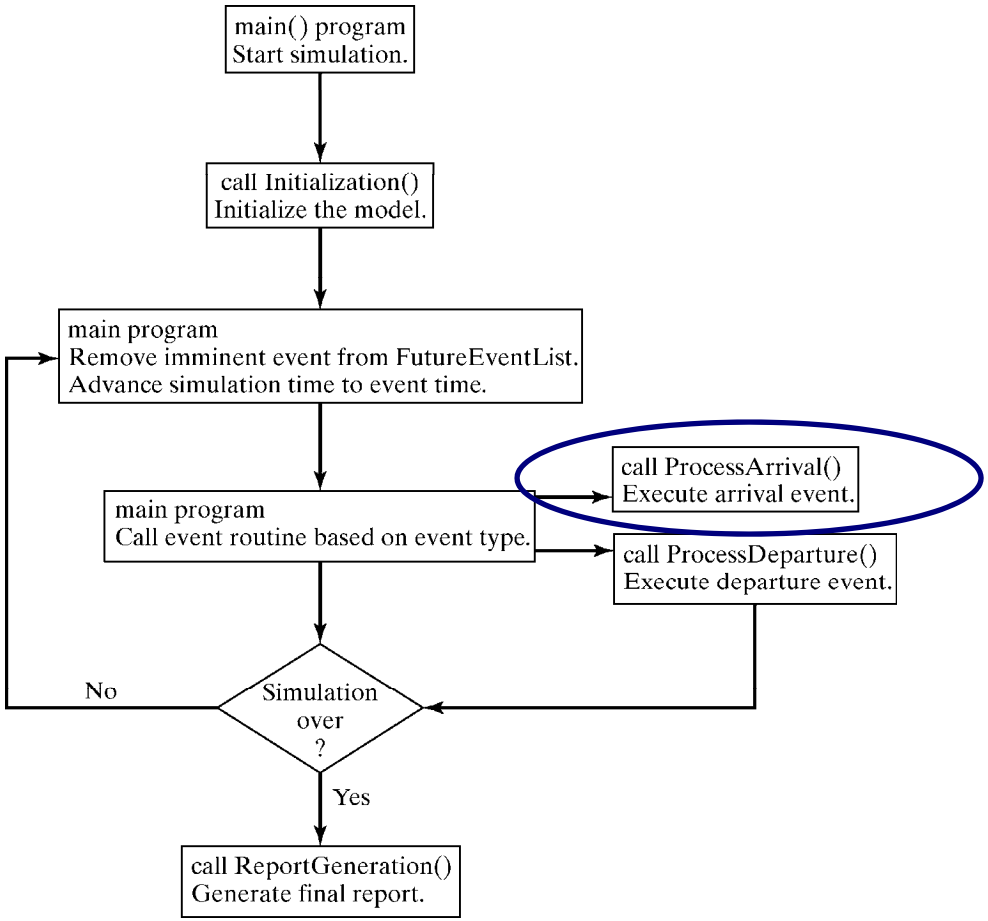
- Structure of the initialization method:

```
// seed the event list with TotalCustomers arrivals
 public static void Initialization()   {
  Clock = 0.0;
  QueueLength = 0;
  NumberInService = 0;
  LastEventTime = 0.0;
  TotalBusy = 0 ;
  MaxQueueLength = 0;
  SumResponseTime = 0;
  NumberOfDepartures = 0;
  LongService = 0;
  // create first arrival event
  Event evt = new Event(arrival, exponential( stream,
MeanInterArrivalTime));
  FutureEventList.enqueue( evt );
 }
```

Single-Server Queue Example

[Simulation in Java]

- The arrival event method:



Single-Server Queue Example

[Simulation in Java]

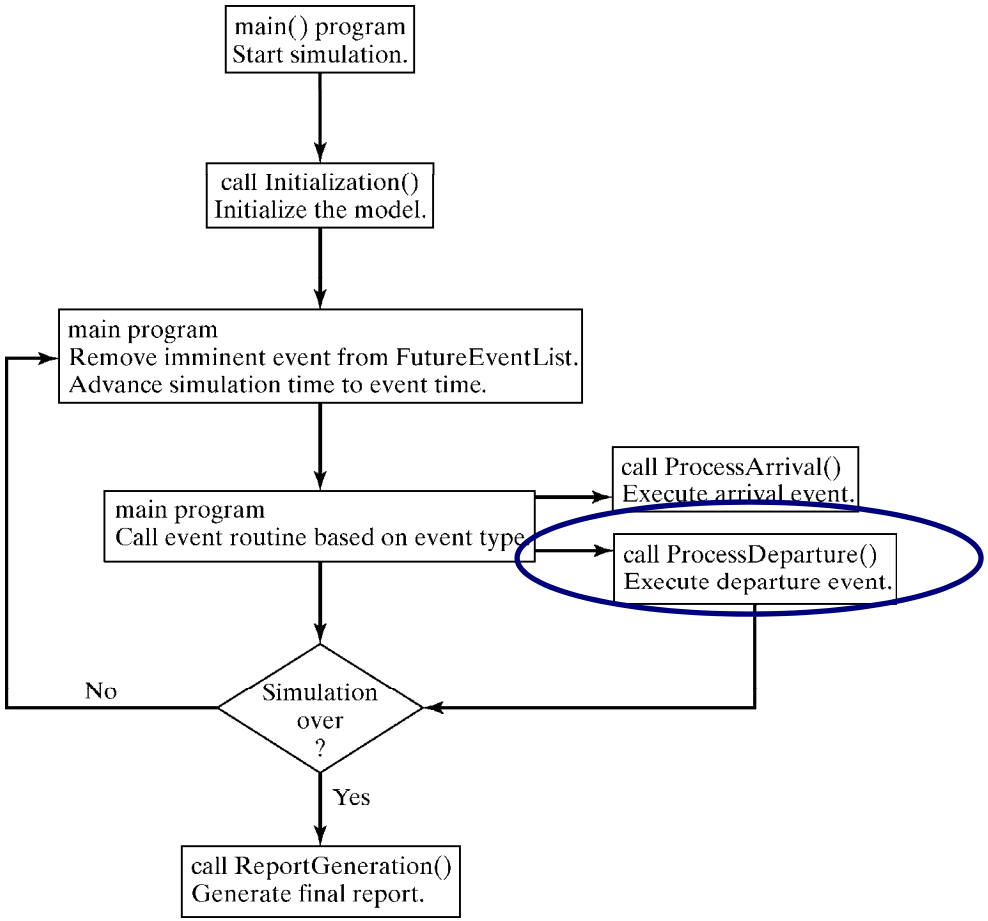
- Structure of the arrival event method:
  - Update server status
  - Collect statistics
  - Schedule next arrival

```
public static void ProcessArrival(Event evt) {
  Customers.enqueue(evt);
  QueueLength++;
  // if the server is idle, fetch the event, do statistics
  // and put into service
  if( NumberInService == 0) ScheduleDeparture();
  else TotalBusy += (Clock - LastEventTime);  // server is busy
  // adjust max queue length statistics
  if (MaxQueueLength < QueueLength) MaxQueueLength =
QueueLength;
  // schedule the next arrival
  Event next_arrival = new Event(arrival,
Clock+exponential(stream, MeanInterArrivalTime));
  FutureEventList.enqueue( next_arrival );
  LastEventTime = Clock;
 }
```

Single-Server Queue Example

[Simulation in Java]

- The departure event method:



Single-Server Queue Example

[Simulation in Java]

- Structure of the departure event method:
  - Obtain the job at the head of the queue

```
public static void ScheduleDeparture() {

 double ServiceTime;
 // get the job at the head of the queue

 while (( ServiceTime = normal(stream, MeanServiceTime, SIGMA)) < 0 );
 Event depart = new Event(departure,Clock+ServiceTime);
 FutureEventList.enqueue( depart );
 NumberInService = 1;
 QueueLength--;
}
```

… continued on next slide …

Single-Server Queue Example

[Simulation in Java]

- Structure of the departure event method (continued):
  - Get the description of finishing customer
  - Schedule departure of the next customer if queue is not emptied
  - Collect statistics

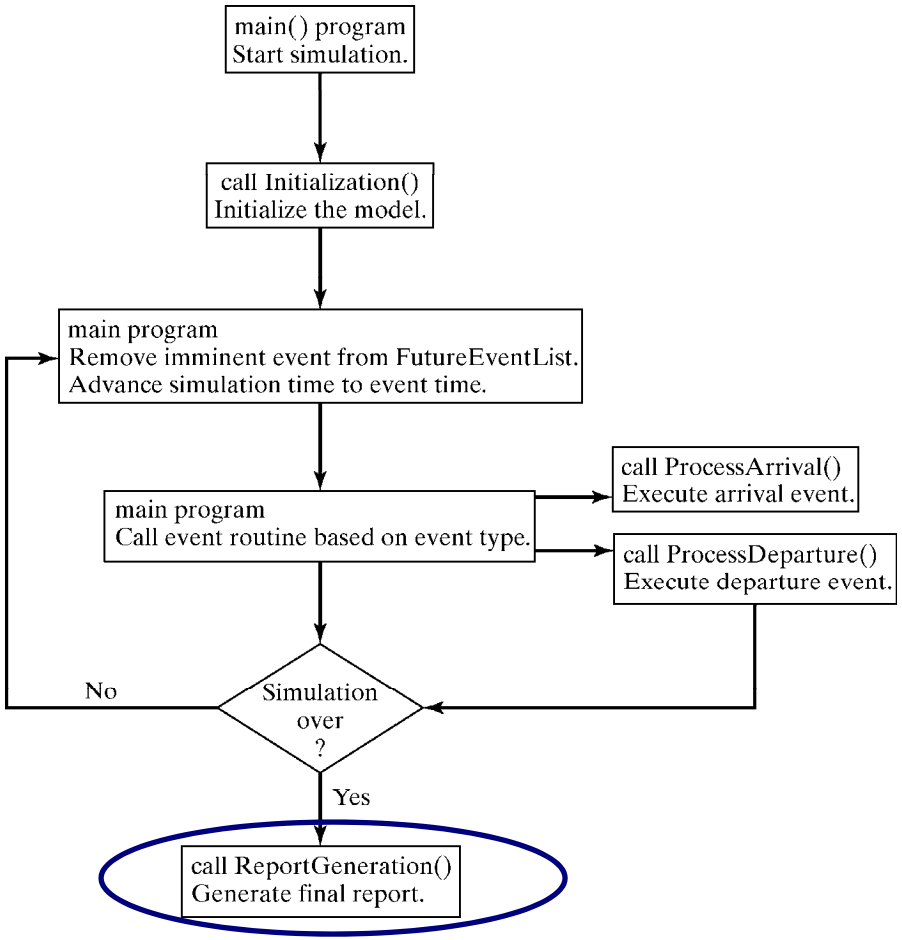
… continued from last slide …

```
public static void ProcessDeparture(Event e) {
 // get the customer description
 Event finished = (Event) Customers.dequeue();
 // if there are customers in the queue then schedule
 // the departure of the next one
  if( QueueLength > 0 ) ScheduleDeparture();
  else NumberInService = 0;
  // measure the response time and add to the sum
  double response = (Clock - finished.get_time());
  SumResponseTime += response;
  if( response > 4.0 ) LongService++; // record long service
  TotalBusy += (Clock - LastEventTime );
  NumberOfDepartures++;
  LastEventTime = Clock;
 }
```

Single-Server Queue Example

[Simulation in Java]

- Report generator:



Single-Server Queue Example

[Simulation in Java]

- Structure of the report generator:

```
public static void ReportGeneration() {
double RHO   = TotalBusy/Clock;
double AVGR  = SumResponseTime/TotalCustomers;
double PC4   = ((double)LongService)/TotalCustomers;


        System.out.println( "SINGLE SERVER QUEUE SIMULATION -
        GROCERY STORE CHECKOUT COUNTER ");
        System.out.println( "\tMEAN INTERARRIVAL TIME                    "
        + MeanInterArrivalTime );
        System.out.println( "\tMEAN SERVICE TIME                         "
        + MeanServiceTime );
        System.out.println( "\tSTANDARD DEVIATION OF SERVICE TIMES
        "
        + SIGMA );
```

… continued on next slide …

Single-Server Queue Example

[Simulation in Java]

- Structure of the report generator (continued):

… continued from last slide …

```
        System.out.println( "\tNUMBER OF CUSTOMERS SERVED
        "
        + TotalCustomers );
        System.out.println();
        System.out.println( "\tSERVER UTILIZATION
        “ + RHO );
        System.out.println( "\tMAXIMUM LINE LENGTH
        " + MaxQueueLength );
        System.out.println( "\tAVERAGE RESPONSE TIME
        " + AVGR + "  MINUTES" );
        System.out.println( "\tPROPORTION WHO SPEND FOUR ");
        System.out.println( "\t MINUTES OR MORE IN SYSTEM
        " + PC4 );
        System.out.println( "\tSIMULATION RUNLENGTH
        " + Clock + " MINUTES" );
        System.out.println( "\tNUMBER OF DEPARTURES
        " + TotalCustomers );
}
```

Single-Server Queue Example

[Simulation in Java]

- Sim class methods to generate exponential and normal random variates:

```
public static double exponential(Random rng, double mean) {
 return -mean*Math.log( rng.nextDouble() );
}
public static double SaveNormal;
public static int  NumNormals = 0;
public static final double  PI = 3.1415927 ;

public static double normal(Random rng, double mean, double sigma) {
      double ReturnNormal;       // should we generate two normals?
      if(NumNormals == 0 ) {
        double r1 = rng.nextDouble();
        double r2 = rng.nextDouble();
        ReturnNormal = Math.sqrt(-2*Math.log(r1))*Math.cos(2*PI*r2);
        SaveNormal   = Math.sqrt(-2*Math.log(r1))*Math.sin(2*PI*r2);
        NumNormals = 1;
      } else {
        NumNormals = 0;
        ReturnNormal = SaveNormal;
      }
      return ReturnNormal*sigma + mean ;
 }
```

Single-Server Queue Example

[Simulation in Java]

- The output:

```
SINGLE SERVER QUEUE SIMULATION - GROCERY STORE CHECKOUT
COUNTER
MEAN INTERARRIVAL TIME                    4.5
MEAN SERVICE TIME                         3.2
STANDARD DEVIATION OF SERVICE TIMES       0.6
NUMBER OF CUSTOMERS SERVED                1000
SERVER UTILIZATION                        0.7175
MAXIMUM LINE LENGTH                       7.0
AVERAGE RESPONSE TIME                     6.7358  MINUTES
PROPORTION WHO SPEND FOUR
MINUTES OR MORE IN SYSTEM                 0.675
SIMULATION RUNLENGTH                      4455.02 MINUTES
NUMBER OF DEPARTURES                      1000
```

- Note: Most of the output statistics are estimates that contain random error.

Simulation in GPSS

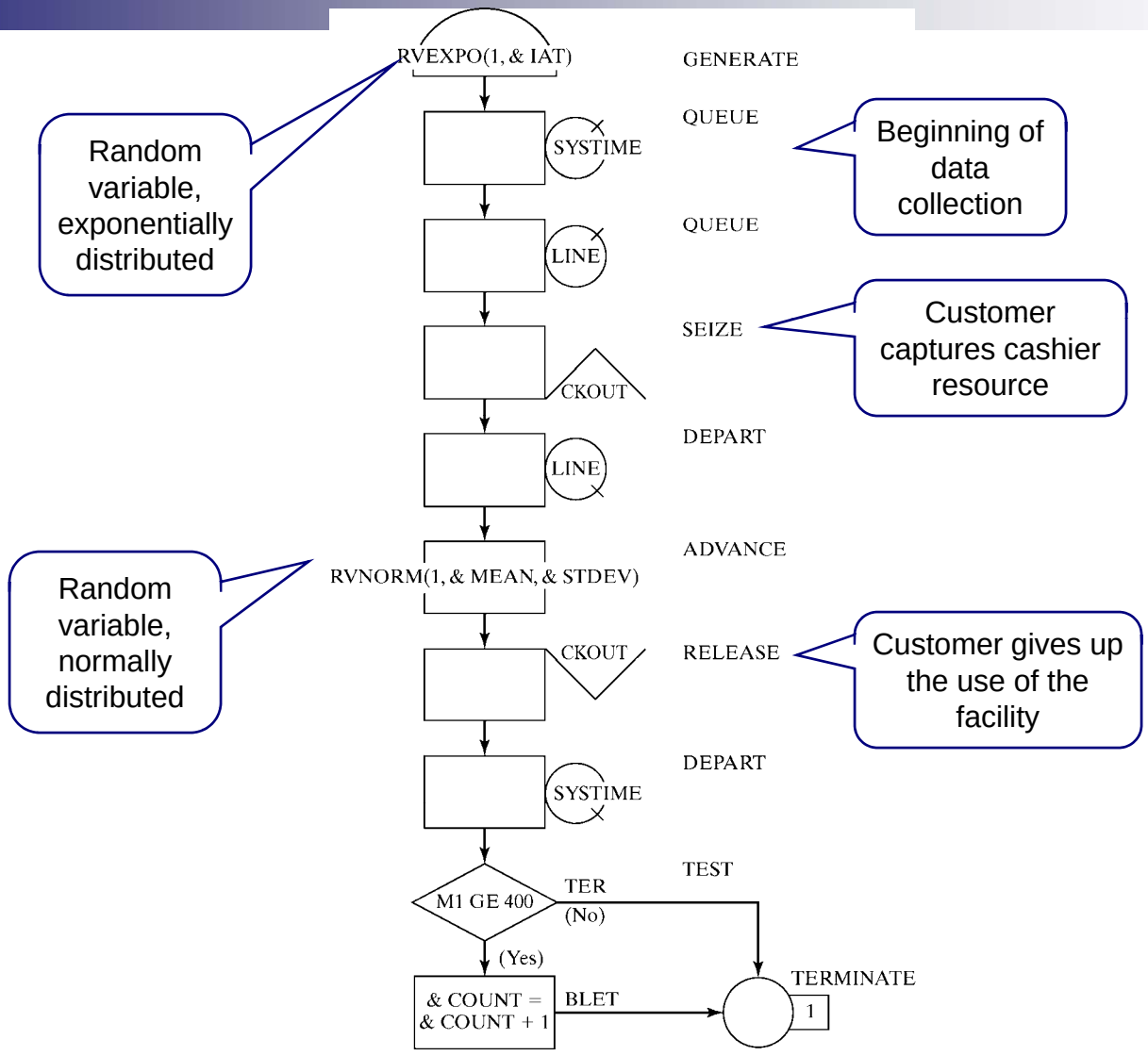
- GPSS is a highly structured, special-purpose simulation programming language.
  - Based on the process-interaction approach.
  - Oriented toward queueing systems.
- Use of block diagram:
  - Provides a convenient way to describe the system.
  - With over 40 standard blocks.
  - Blocks represents events, 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
```

Single-Server Queue Example

[Simulation in GPSS/H]

- Write input data to file and provide formatting information.

```
*       Write Input Data to File
*
        PUTPIC       FILE=OUT,LINES=5,(&IAT,&MEAN,&STDEV,&LIMIT)
        Mean interarrival time                          **.**  minutes
        Mean service time                               **.**  minutes
        Standard deviation of service time               **.**  minutes
        Number of customers to be served                 *****
```

Single-Server Queue Example

[Simulation in GPSS/H]

- GPSS/H block section description and inputs.
- START control statement controls simulation execution.

```
*     GPSS/H Block Section
*
      GENERATE      RVEXPO(1,&IAT)              Exponential arrivals
      QUEUE         SYSTIME                     Begin response time data collection
      QUEUE          LINE                       Customer joins waiting line
      SEIZE         CHECKOUT                    Begin checkout at cash register
      DEPART        LINE                        Customer starting service leaves queue
      ADVANCE       RVNORM(1,&MEAN,&STDEV)       Customer's service time
      RELEASE       CHECKOUT                    Customer leaves checkout area
      DEPART        SYSTIME                     End response time data collection
      TEST GE       M1,4,TER                    Is response time GE 4 minutes?
      BLET          &COUNT=&COUNT+1             If so, add 1 to counter
      TER           TERMINATE                   1
*
      START      &LIMIT       Simulate for required number
```

Single-Server Queue Example

[Simulation in GPSS/H]

- Write desired output data to file OUT.

```
*     Write Customized Output Data to File
*
      PUTPIC       FILE=OUT,LINES=7,(FR(CHECKOUT)/1000,QM(LINE),_
      QT(SYSTIME),&COUNT/N(TER),AC1,N(TER))
      Server utilization                                  .***
      Maximum line length                                 **
      Average response time                               **.**    minutes
      Proportion who spend four minutes                   .***
          or more in the system
      Simulation runlength                                ****.** minutes
      Number of departures                                ****
*
      END
```

Single-Server Queue Example

[Simulation in GPSS/H]

- Sample output report:

Mean interarrival time	4.50 minutes
Mean service time	3.20 minutes
Standard deviation of service time	0.60 minutes
Number of customers to be served	1000
Server utilization	0.676
Maximum line length	7
Average response time	6.33 minutes
Proportion who spend four minutes or more in the system	0.646
Simulation runlength	4767.27 minutes
Number of departures	1000

Simulation in SSF

- The Scalable Simulation Framework (SSF) is an Application Program Interface (API)
  - Describes a set of capabilities for object-oriented, process-view simulation.
  - The API is sparse and allows implementations to achieve high performance, e.g. on parallel computers.
  - A widely used base, particularly in network simulation by using the add-on framework SSFNet.
- SSF API defines 5 base classes:
  - *Processes*: implements threads of control (where the *action* method contains the execution body of the thread.)
  - *Entity*: describes simulation objects.
  - *inChannel* and *outChannel*: communication endpoints.
  - *Event*: defines messages sent between entities.

Single-Server Queue Example

[Simulation in SSF]

- *SSQueue* is a class that contains the whole simulation experiment:
  - Defines experimental constants.
  - Contains SSF communication endpoints.
  - Defines an inner class, arrival.

```
class SSQueue extends Entity {

 private static Random rng;
 public static final double MeanServiceTime = 3.2;
 public static final double SIGMA = 0.6;
 public static final double MeanInterarrivalTime = 4.5;
 public static final long ticksPerUnitTime = 1000000000;
 public long generated=0;
 public Queue Waiting;
 outChannel out;
 inChannel in;

 public static long      TotalCustomers=0, MaxQueueLength=0,
                         TotalServiceTime=0;
 public static long      LongResponse=0, umResponseTime=0,
                         jobStart;

 class arrival {
    long id,  arrival_time;
    public arrival(long num, long a) { id=num; arrival_time = a; }
 }
```

… continued on next slide…

Single-Server Queue Example

[Simulation in SSF]

- *Arrival* is an SSF process:
  - Stores the identity of entity
  - Creates a random-number generator,
  - Generates and enqueues a new arrival, then blocks for an inter-arrival time.

… continued from last slide…

```
class Arrivals extends process {
   private Random rng;
   private SSQueue owner;
   public Arrivals (SSQueue _owner, long seed) {
        super(_owner); owner = _owner;
        rng = new Random(seed);
   }
   public boolean isSimple() { return true; }
   public void action() {
      if ( generated++ > 0 ) {
        // put a new Customer on the queue with the present arrival time
        int Size = owner.Waiting.numElements();
        owner.Waiting.enqueue( new arrival(generated, now()));
        if( Size == 0) owner.out.write( new Event() );   // signal start of burst
      }
      waitFor(owner.d2t( owner.exponential(rng,
                  owner.MeanInterarrivalTime)) );
  }
 }
}
```

Single-Server Queue Example

[Simulation in SSF]

- *Server* process:
  - This process is called when a job has completed service or by a signal from the arrival process.
  - Update statistics.
  - Customer is dequeued from the waiting list or the process suspends if no customer was waiting.

```
class Server extends process {
 private Random rng;
 private SSQueue owner ;
 private arrival in_service;
 private long service_time;

 public Server(SSQueue _owner, long seed) {
       super(_owner);
       owner = _owner;
       rng = new Random(seed);
 }
 public boolean isSimple() { return true; }
```

… continued on next slide…

Single-Server Queue Example

[Simulation in SSF]

- *Server* process (continued):

… continued from last slide…

```
public void action() {
// if in_service is not null, we entered because of a job completion
    if( in_service != null ) {
        owner.TotalServiceTime += service_time;
        long in_system = (now() -in_service.arrival_time);
        owner.SumResponseTime  +=  in_system;
        if( owner.t2d(in_system) > 4.0 ) owner.LongResponse++;
        in_service = null;
        if( owner.MaxQueueLength < owner.Waiting.numElements() + 1 )
            owner.MaxQueueLength = owner.Waiting.numElements() + 1;
        owner.TotalCustomers++;
    }
    if( owner.Waiting.numElements() > 0 ) {
        in_service = (arrival)owner.Waiting.dequeue();
        service_time = -1;
        while ( service_time < 0.0 )
          service_time = owner.d2t(owner.normal( rng, owner.MeanServiceTime, owner.SIGMA));
    waitFor( service_time );
    } else {
        waitOn( owner.in );  // we await a wake-up call
    }
  } }
```

Simulation in SSF

- SSF bridges the gap between models developed in pure Java and models developed in languages specifically designed for simulation.
- It also provides the flexibility offered by a general-programming language, yet has essential support for simulation.

Simulation Software

- All the simulation packages described in later subsections run on a PC under Microsoft Windows 2000 or XP.
- Common characteristics:
  - Graphical user interface, animation
  - Automatically collected outputs.
  - Most provide statistical analyses, e.g., confidence intervals.
- All packages considered in this chapter take the process-interaction worldview, a few also allow event-scheduling models and mixed discrete-continuous models.
- For animation, some emphasize scale drawings in 2-D or 3-D; others emphasize iconic-type animation.
- Almost all offer dynamic business graphing, e.g., time lines, bar charts and pie charts.

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

Arena [Simulation Software]

- Arena can be used for simulating discrete and continuous systems.
- At the heart of Arena is the SIMAN simulation language.
- The Arena Basic Edition:
  - For modeling business processes and other systems in support of high-level analysis needs.
- The Arena Standard Edition:
  - For modeling more detailed discrete and continuous systems.
  - Models are built from graphical objects called modules to define system logic and physical components.
  - Includes modules focused on specific aspects of manufacturing and material-handling systems.
- The Arena Professional Edition:
  - With capability to craft custom simulation objects that mirror components of real system, including terminology, process logic, data, etc.

Arena [Simulation Software]

- The Arena family includes:
  - OptQuest: an optimization software package.
  - Arena Contact Center and Arena Packaging: designed specifically to model call centers and high-speed production lines.
- Arena's Input Analyzer automates the process of selecting the proper distribution and its inputs.
- The Output Analyzer and Process Analyzer automate comparison of different design alternatives.

AutoMod [Simulation Software]

- AutoMod Product Suite includes:
  - AutoMod simulation package, AutoStat for experimentation and analysis, and AutoView for making AVI moves of the built-in 3-D animation.
- Main focus: manufacturing and material-handling systems (has built in templates.)
- Also contains a full simulation programming language.
- Strength: detailed, large models used for planning, operational decision support, and control-system testing.
- An AutoMod model consists of one or more systems.
  - A system can be either a process system or a movement system.
  - A model may contain any number of systems, which can be saved and reused as objects in other models.
- AutoStat provides a complete environment for the user to define scenarios, conduct experimentation and perform analysis.
- Optimization is based on an evolutionary strategies algorithm.

Extend [Simulation Software]

- Extend OR, Industry, and Suite are used for simulating discrete and mixed discrete-continuous systems.
- Extend CP is for continuous modeling only.
- Extend combines a block diagram approach to model-building with a development environment for creating new blocks.
  - Models are built by placing and connecting blocks, and entering the parameters on the block's dialog window.
  - For creating new blocks, Extend comes with a compiled C-like programming environment.
- Input parameters can be changed interactively during a model run and can come from external sources.
- Provides iconic process-flow animation of the block diagram.
- Has an open architecture and also supports linking to and using code written in external languages.

Flexsim [Simulation Software]

- Flexsim is a discrete-event, object-oriented simulator developed in C++, using Open GL technology.
- Flexsim is commonly used to
  - To build models that behave like the actual physical or conceptual systems they represent.
  - To improve production efficiencies and reduce operating costs through simulation, experimentation, and optimization of dynamic flow systems.
  - Engineers and managers use Flexsim to evaluate plant capacity, balance packaging and manufacturing lines, manage bottlenecks.
- The results of each simulation can be analyzed:
  - Graphically through 3D animation, and
  - Through statistical reports and graphs.

Micro Saint [Simulation Software]

- Micro Saint is a general-purpose, discrete-event, network simulation-software package for building models that simulate real-life processes.
- It does not use the terminology or graphic representations of a specific industry.
- Model can be built for any process that can be represented by a flowchart diagram.
- It provides two views of the simulation model:
  - Network diagram view: the process flowchart in action.
  - Actionview provides a realistic 2-D picture of the process.
- OptQuest optimization is included:
  - Automatically search for and find optimal or near-optimal solutions.

ProModel [Simulation Software]

- A simulation and animation tool designed to model manufacturing systems.
  - Has manufacturing-oriented modeling elements and rule-based decision logic.
  - The modeling elements in ProModel are parts (entities), locations, resources, path networks, routing and processing logic, and arrivals.
- Includes logic for automatically generating cost data associated with a process.
- Also comes with an output viewer.
- Its runtime interface allows a user to define multiple scenarios for experiments.
- It offers 2-D animation with an optional 3-D like perspective view.
- The company also offers MedModel for healthcare systems and ServiceModel for service systems.

Delmia/QUEST [Simulation Software]

- Delmia/QUEST is a manufacturing-oriented simulation package.
  - Combines an object-based, true 3-D simulation environment with a graphical user interface for material-flow modules.
  - Incorporates 2-D and 3-D CAD geometry to create a virtual factory environment.
- The company also offers a number of workcell simulators:
  - IGRIP® for robotic simulation and programming.
  - ERGO™ for ergonomic analyses.
  - PROCESS ENGINEER™ for process-planning.
- Simulation Control Language (SCL): allows expert users to define customer behavior and to gain control over simulation.
- Batch Control Language (SCL): open architecture allows the advanced user to perform batch simulation runs to automatically collect and tabulate data.
- Output is available both numerically and visually.

Slide 64

B45 Delmia/QUEST
(Check this throughout for consistency)
Brian; 2005/03/11

SIMUL8

[Simulation Software]

- SIMUL8 models are created by drawing the flow of work with the computer mouse, using a series of icons and arrows to represent the resources and queues in the system.
- Main focus is service industries where people are processing transactions.
- The company's goal is to spread simulation very widely across businesses.
  - Have very different pricing and support policies.
  - Contains features that watch how the product is being used.
- Simulation models and data are saved in SML format.
- SIMUL8 has a VBA interface and supports ActiveX/COM so that external applications can build and control SIMUL8 simulations.
- The product is available at two levels: Standard and Professional.

WITNESS [Simulation Software]

- WITNESS has separate versions for manufacturing and service industries.
- WITNESS models are based on template elements.
  - Elements may be customized and combined into module elements and templates for reuse.
  - Displayed in a 2-D layout animation with multiple windows and display layers.
- WITNESS has object-model and ActiveX control for simulation embedding and includes direct data links to Microsoft Excel, MINITAB and any OLEDB database source.

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.

OptQuest

[Experimental and Analysis Tools]

- An optimization tool.
- OptQuest is based on a combination of methods: scatter search, tabu search, linear-integer programming, and neural networks.
  - The combination of methods allows the search process to escape local optimality in the quest for the best solution.
    - Scatter search: Population_based approach-
      Creates new solutions with combining existing solutions
    - Tabu search: is then superimposed to prohibit the search from reinvestigating previous solutions
    - Neural Network: Screens out solutions likely to be poor

AutoStat

[Experimental and Analysis Tools]

- AutoStat is the run manager and statistical-analysis product in the AutoMod product family.
- It provides a number of analyses
  - Including warm-up determination, absolute and comparison confidence intervals, design of experiments, sensitivity analysis.
- The evolutionary-strategies algorithm used is well suited to find a near-optimal solution without getting trapped at a local optimum.
- An end user can define any number of scenarios by defining factors and their range or values.
- AutoStat supports correlated sampling using common random numbers.
- AutoStat is capable of distributing simulation runs across a local area network and pulling back all results to the user's computer.

SimRunner [Experimental and Analysis Tools]

- SimRunner was developed by PROMODEL Corporation.
  - Available for ProModel, MedModel and ServiceModel.
- Uses genetic algorithms and evolutionary strategies.
- Manipulates the input factors within boundaries specified by the user seeking to optimize the objective function.
- Also has a utility for helping users estimate:
  - The end of the warm-up phase
  - The number of replications needed to obtain an estimate of the objective function's mean value to within a specified percentage error and confidence level.

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.