Simulation and Modeling
Discrete System Simulation

Er. Narayan Sapkota, M.Sc.
narayan.sapkota@eemc.edu.np

Everest Engineering College (Senior Lecturer)
Kathmandu University (Visiting Faculty)

January 5, 2025

Syllabus

1. Discrete System Simulation
2. Representation of Time
3. Generation of Arrival Patterns
4. Simulation of Telephone System
5. Gathering Statistics
6. Counters and Summary Statistics
7. Measuring Utilization and Occupancy
8. Recording Distribution and Transit Time
9. Discrete Simulation Languages

Table of Contents

Discrete System Simulation (1)

- Discrete system simulation involves modeling systems where changes occur at specific, discrete points in time rather than continuously.
- These systems typically evolve through a series of discrete events or steps, which are often triggered by specific conditions or actions.
- In discrete event simulation, the system's state is updated at distinct time intervals based on events that occur in the system.

The primary steps involved in discrete system simulation are:

1. Event Generation: Define and simulate events that cause the system's state to change.
   - Event 1: A customer arrives at the bank at time 0.
   - Event 2: The customer is served by the cashier and leaves at time 5.
   - Event 3: A second customer arrives at time 3, and so on.
2. Event Scheduling: Events are scheduled to occur at future times based on the system's logic.

Discrete System Simulation (2)

- Event 1 (customer arrival at time 0) is scheduled for time 0.
- After Event 1, Event 2 (cashier serving the customer) is scheduled at time 5.
- Event 3 (second customer arrives at time 3) is scheduled for time 3.

3. State Changes: The system state is updated as each event occurs.
   - Initially, the system is in the state where the queue is empty and the cashier is free.
   - After Event 1 (customer arrival at time 0), the queue becomes non-empty (1 customer).
   - After Event 2 (service completion at time 5), the cashier becomes free again, and the queue reduces by 1.
   - New states keep updating based on the scheduled events.
4. Termination: The simulation continues until a predefined condition (e.g., a maximum simulation time or a specified event) is met. The simulation might run until a total of 8 hours have passed or until all customers have been served.

Discrete System Simulation (3)

Applications

Discrete system simulation is commonly used to model systems with queues(e.g service systems), manufacturing (e.g., simulate the production process, analyze bottlenecks, and optimize resources),Communication Networks (e.g., simulate data traffic and evaluate network performance).

Advantages

- Flexibility: It can handle complex, nonlinear, and stochastic behavior that is difficult to model analytically.
- Realistic Analysis: It allows for detailed modeling of time-dependent processes, making it suitable for real-world applications.
- Optimization: It can be used to find optimal configurations or strategies by experimenting with different scenarios.

Table of Contents

Representation of Time (1)

In discrete system simulation, time is represented as a sequence of discrete time points, typically denoted as $t_0, t_1, t_2, \ldots$, where changes in the system occur at specific points in time, not continuously. Time is represented in two main ways:

1. Simulation Clock
   The simulation clock track of the current time, which is updated during the simulation as events occur. The time progresses discretely, moving from one event to the next.
   - The clock starts at the initial simulation timet_0.
   - Each event has a scheduled time t_e when it is set to occur.
   - The simulation clock advances to the next event's time t_{next}.
2. Event-Driven Time Advancement
   In event-driven simulation, the system's time does not increment continuously. Instead, it advances from one event to another, with each event occurring at a specific point in time.
   - The simulation clock progresses to the time of the next scheduled event.

Representation of Time (2)

- The system's state changes only when an event occurs, and time remains unchanged between events.
- After each event, the simulation recalculates the next event time, ensuring that time moves forward based on the system's dynamics.

The concept of time is directly related to the events scheduling. The time step between events can be irregular, as as it depends on the system's dynamics and the event type.

- For each event, a time t_e is generated based on the system's behavior.
- Events are scheduled in advance and executed when their scheduled time is reached.
- The event list is continuously updated as new events are added.

Table of Contents

Generation of Arrival Patterns (1)

Arrival patterns refer to the distribution of arrival times or the occurrence of events in the system, such as customer arrivals in a queuing system, or packet arrivals in a network simulation. These arrival patterns are required to model the systems behavior and can be generated based on different probabilistic distributions.

1. **Poisson Process**
   One of the most common models for generating arrival patterns is the Poisson process. In this model-
   - Arrivals occur randomly and independently over time.
   - The time between consecutive arrivals, also known as inter-arrival time, follows an exponential distribution.
   - The average rate of arrival is constant over time, denoted as λ (arrivals per unit time).
   - The number of arrivals in a given time period follows a Poisson distribution.

Generation of Arrival Patterns (2)

The probability of k arrivals in a time period t is given by the Poisson distribution:

$$P(k \text{ arrivals}) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}$$

where λ is the rate of arrivals, and t is the time period.

2. **Exponential Distribution**

The inter-arrival times in the Poisson process follow an exponential distribution. The probability density function (PDF) of the inter-arrival time X is given by:

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0$$

where λ is the rate parameter, and x is the time between successive arrivals.

Generation of Arrival Patterns (3)

3. **Birth-Death Process**
   The birth-death process is another method for modeling arrival patterns, where arrivals (births) and departures (deaths) occur according to specific rates. This process can be used to model systems where both arrivals and departures happen dynamically, such as in queuing models.
   - Arrival rates λ_n depend on the state n (number of customers or entities in the system).
   - Departure rates μ_n also depend on the system's state.

   This model is commonly used in systems where the state of the system (e.g., the number of customers in a queue) affects the arrival and departure rates.
4. **Uniform and Normal Distributions**
   In some cases, arrivals may follow a uniform or normal distribution. For instance:

Generation of Arrival Patterns (4)

- Uniform Distribution: Arrival times are uniformly distributed over a given time interval, meaning each time point within the interval has an equal probability of an event occurring.
- Normal Distribution: Although less common in arrival modeling, a normal distribution might be used when arrival times tend to cluster around a mean time, with symmetrical variability around the mean.

Simulation of Arrival Patterns

In simulation, arrival patterns are generated using random number generation techniques. To simulate arrivals based on a Poisson process:

- Generate random numbers uniformly distributed between 0 and 1.
- Use the inverse transform method to convert these random numbers into exponential inter-arrival times.
- The resulting arrival times can then be used to simulate the events in the system.

Bootstrapping (1)

Bootstrapping is a statistical method used to estimate the distribution of a statistic (such as the mean or variance) by resampling with replacement from the original data. It is a non-parametric approach, meaning it does not assume any specific underlying distribution for the data.

1. Resampling: Multiple new samples (called "bootstrap samples") are drawn from the original dataset randomly, and each element may be selected more than once.
2. Sampling with Replacement: Each time a data point is selected, it is replaced back into the original dataset, meaning the same data point can be chosen multiple times in the new sample.

Bootstrapping (2)

3. Estimation: Repeat steps 1 and 2 many times (e.g., 1,000 or 10,000 times) to create a distribution of the statistic. The statistic of interest (e.g., mean, median, standard deviation) is calculated on each bootstrap sample.

   The distribution of these statistics provides an estimate of the statistic's variability. The variability in the computed statistics can be used to construct confidence intervals, assess the precision of the estimate, or test hypotheses.

Applications

- Model Evaluation: It can be used to assess the performance of a predictive model by generating multiple training and testing datasets.

Bootstrapping (3)

- Confidence Intervals: It is commonly used to construct confidence intervals for population parameters, especially when the sample size is small or the distribution is unknown.
- Hypothesis Testing: Bootstrapping can be applied in hypothesis testing, especially when the assumptions for traditional methods are violated.

Advantages

- No Distribution Assumptions: Bootstrapping is a non-parametric method, so it does not require assumptions about the data's underlying distribution.
- Versatility: It can be used for a wide range of statistics, including means, variances, regression coefficients, and more.

Bootstrapping (4)

- Works with Small Samples: It is particularly useful when dealing with small datasets, where traditional methods might not be applicable or reliable.

Limitations

- Computational Cost: Bootstrapping requires generating many resamples, which can be computationally expensive, especially for large datasets.
- Not Suitable for All Data Types: It assumes that the sample is representative of the population, so it might not be suitable for highly skewed or non-random data.

Table of Contents

Simulation of Telephone System

See Page no. 177 to 181 of *G. Gorden, System Simulation, Prentice Hall of India*

Simulation of Telephone System: Delayed Calls

See Page no. 181 to 185 of *G. Gorden, System Simulation, Prentice Hall of India*

Table of Contents

Gathering Statistics (1)

Gathering statistics refers to the process of collecting and analyzing data from a simulation experiment to compute important metrics, understand system performance, and derive conclusions based on the results. This is a important in validating and verifying simulation models and in making decisions based on the simulation output.

In discrete event simulation and other types of simulation models, various types of statistics collected to evaluate system performance are:

- Counts: These represent the number of occurrences of specific events or the number of entities of a particular type. For example, this could include counting the number of customers served or the number of times a machine fails. Counts are essential for tracking the volume of activity in the simulation.

Gathering Statistics (2)

- Summary Measures: These typically include extreme values (like minimum and maximum), mean values (averages), and standard deviations. These measures help provide a general understanding of the system's behavior over time. For example, the mean waiting time in a queue gives a good indication of the efficiency of the service process.
- Utilization: Utilization refers to the fraction or percentage of time an entity (such as a server or resource) is actively engaged in some task. For example, if a machine is used for 70% of the simulation time, its utilization is 70%. This is critical for understanding whether resources are being underused or overloaded.
- Occupancy: Occupancy is a measure of how often a group of entities (e.g., servers, machines) is in use during the simulation. It is typically calculated as the average fraction of time that the entities are occupied. This helps assess how efficiently resources are utilized in a system.

Gathering Statistics (3)

- Transit Times: Transit times measure the time taken for an entity to move from one part of the system to another. For example, in a manufacturing simulation, the transit time would be the time taken for a part to move from one workstation to the next. Understanding transit times is essential for identifying delays or inefficiencies in the flow of entities through the system.
- Queue Length: The number of entities in a queue at any given time.
- Waiting Time: The amount of time an entity spends in a queue before being served.
- Throughput: The number of entities processed by the system per unit of time.
- Response Time: The total time from when an entity enters the system until it is completed.

Table of Contents

Counters and Summary Statistics (1)

Counters and summary statistics are tools for collecting and analyzing data about the system's performance. They help quantify various aspects of the system's operation, such as the number of events, the performance of resources, and the variation in key variables over time.

Counters

- Counters are used to keep track of specific events or occurrences within the simulation.
- These events may include the arrival of entities, the completion of processes, or the number of times a certain condition is met during the simulation run.
- Counters are typically updated each time the corresponding event occurs and are used to generate statistical measures of system performance.

Counters and Summary Statistics (2)

- Counters provide useful data that can be aggregated to calculate overall performance metrics, such as throughput, average service times, and system utilization.
- They offer a simple yet powerful way to track key events and system states.

Types of Counters

- Entity Counters: These counters track the number of entities of a particular type that enter, exit, or are processed by the system. For example, the number of customers that arrive at a service station or the number of parts that pass through a production line.

Counters and Summary Statistics (3)

- Event Counters: These counters track the number of times specific events occur. For example, counting the number of machine breakdowns in a manufacturing system or the number of service requests in a queuing system.
- Resource Utilization Counters: These counters track how often and for how long resources (such as servers, machines, or workers) are in use during the simulation. This helps in understanding system efficiency and resource allocation.

Counters and Summary Statistics (4)

Summary Statistics

Summary statistics are used to provide a concise overview of the data gathered during the simulation. These statistics summarize important features of the system's behavior, such as central tendency, variation, and extreme values.

Types of Summary Statistics

1. Mean (Average): The mean is a measure of central tendency and represents the average value of a variable over the simulation run. For example, the mean waiting time for customers in a queue or the average time a machine is in operation.

$$\mu = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Counters and Summary Statistics (5)

2. Variance and Standard Deviation: These statistics measure the spread or variability of a variable. Variance is the average of squared deviations from the mean, and the standard deviation is the square root of the variance. For example, the standard deviation of queue lengths gives an idea of how much the queue fluctuates over time. Variance (σ^2) is calculated as,

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

3. Maximum and Minimum Values: These statistics capture the extreme values observed in the system. For example, the maximum queue length or the minimum system utilization during the simulation run.

$$\text{Maximum Value} = \max(x_1, x_2, ..., x_n)$$

$$\text{Minimum Value} = \min(x_1, x_2, ..., x_n)$$

Counters and Summary Statistics (6)

4. Percentiles and Quantiles: These statistics divide the data into intervals, with each interval containing a specific percentage of the total data. For example, the 90th percentile of waiting times might indicate that 90% of customers experience a waiting time shorter than this value.

$$P_k = \text{The value such that } k\% \text{ of data is less than or equal to } P_k$$

Where:

- k is the percentile (e.g., the 90th percentile for $k = 90$).

Counters and Summary Statistics (7)

5. Confidence Intervals: These provide a range of values within which the true value of a statistic is likely to fall, with a specified level of confidence. This is important for understanding the precision of the simulation results.

$$\mu \pm Z \cdot \frac{\sigma}{\sqrt{n}}$$

Where:

- μ is the sample mean.
- Z is the z-score corresponding to the desired confidence level (e.g., 1.96 for 95% confidence).
- σ is the sample standard deviation.
- n is the sample size.

Counters and Summary Statistics (8)

In a queuing simulation, counters may track the number of customers served, the number of customers waiting, and the number of events (such as service completions). Summary statistics, such as the average waiting time, the maximum queue length, and the standard deviation of service times, can be computed to understand the system's performance and identify areas for improvement.

- Counters provide raw data about specific events and system states.
- Summary statistics offer a way to analyze and interpret the collected data, providing insights into system performance and behavior.
- Together, they help to identify trends, patterns, and bottlenecks in the system and guide decision-making based on the simulation results.

Table of Contents

Measuring Utilization and Occupancy (1)

Utilization refers to the fraction (or percentage) of time that a particular resource (such as a machine, server, or operator) is actively engaged in some productive activity. It is a measure of how much time a resource is busy versus idle.

$$\text{Utilization} = \frac{\text{Total time the resource is busy}}{\text{Total time the resource is available}} \tag{1}$$

Utilization is an important indicator of whether resources are being used efficiently.

Example: In a call center simulation, the *utilization of an agent* would be the fraction of time that the agent is answering calls versus the total available time for the agent to take calls. If an agent is available for 8 hours in a day, but only spends 6 hours answering calls, the utilization would be:

Measuring Utilization and Occupancy (2)

$$\text{Utilization} = \frac{6\,\text{hours}}{8\,\text{hours}} = 0.75\,\text{or}\,75\%$$

Utilization values typically range from 0 to 1 (0% to 100%).

- High Utilization: May indicate a need for additional resources (e.g., more agents or machines) or that the resource is working near its capacity limit.
- Low Utilization: May indicate that the resource is underused, leading to excess capacity and inefficiency in the system.

Measuring Utilization and Occupancy (3)

Occupancy refers to the fraction (or percentage) of time that a *group of resources* (such as servers, machines, or workstations) is actively engaged during the simulation. While utilization is typically calculated for individual resources, occupancy is used for groups of resources and gives a broader view of how the system as a whole is operating.

Occupancy helps measure whether there is sufficient capacity in the system or if the system might be underutilized or overburdened. It looks at the overall usage of resources, rather than focusing on one particular resource.

$$\text{Occupancy} = \frac{\text{Total time the group of resources is in use}}{\text{Total available time for the group of resources}} \tag{2}$$

Occupancy can also be thought of as the average fraction of time that any resource in the group is being used.

Measuring Utilization and Occupancy (4)

Example: In a *queueing system* with 5 servers, if, over a certain period, 3 of the servers are busy for 6 hours and 2 servers are idle for the same time period, the occupancy for the group of 5 servers would be:

$$\text{Occupancy} = \frac{(3 \times 6)\,\text{hours}}{(5 \times 6)\,\text{hours}} = \frac{18}{30} = 0.60\,\text{or}\,60\%$$

Occupancy provides a measure of *how effectively a group of resources is being used*.

- High Occupancy: This may indicate that the system is operating at or near full capacity, and resources may be in short supply (potentially leading to longer waiting times or increased risk of overload).
- Low Occupancy: This may indicate that the system has excess capacity, leading to inefficiency or waste, where resources are not being fully utilized.

Measuring Utilization and Occupancy (5)

Metric	Utilization	Occupancy
Definition	Measures the time a *single resource* (server, machine) is busy.	Measures the time a *group of resources* (servers, workstations) is in use.
Focus	Individual resource usage.	Group of resources usage.
Usage	To determine if a specific resource is overused or underused.	To determine if the overall system is efficiently using available resources.
Key Idea	How much of the available time a resource is in use.	How much of the available time, on average, the resources in the system are being used.

Measuring Utilization and Occupancy (6)

Example: Utilization and Occupancy in Telephone System

A common requirement of a simulation is measuring the load on some entity, such as an item of equipment. The simplest measure is to determine *utilization*. Typically, the time history of the equipment usage might appear as shown in Figure 1.

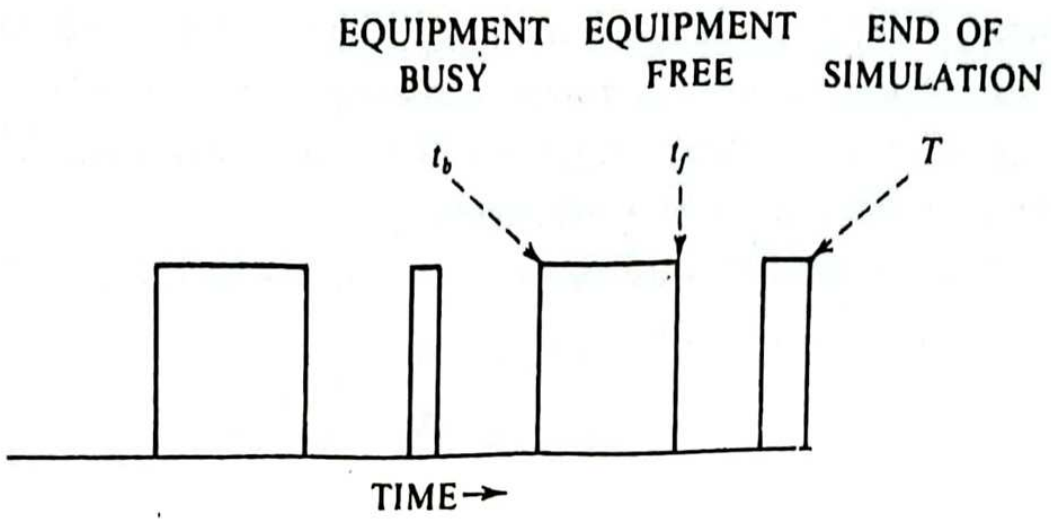

Figure 1: Utilization of Equipment

Measuring Utilization and Occupancy (7)

To measure the utilization, track the time t_b at which the item last became busy. When the item becomes free at time t_f, the interval $t_f - t_b$ is calculated and added to a counter. At the end of the simulation, the utilization U is calculated by dividing the accumulated total by the total time T

$$U = \frac{1}{T} \sum_{i=1}^{N} (t_{f_i} - t_{b_i}), \tag{3}$$

where N is the number of times the entity is used.

It is important to check whether the item is busy at the end of the run and, if so, add to the counter a quantity representing the engagement from the last time it became busy to the end of the simulation. Correspondingly, it is also important to check the initial conditions to see if the entity is busy at the beginning of the run.

Measuring Utilization and Occupancy (8)

For groups of entities, the calculation involves tracking the number of entities in use and the time of the last change. For example, in a telephone system, the number of busy links is recorded (see Figure 2). When the number of links changes at time t_i to n_i, the quantity $n_i(t_{i+1} - t_i)$ is calculated and added to the accumulated total. The average number of links in use, A, is then calculated by dividing the total by the total simulation time T:

$$A = \frac{1}{T} \sum_{i=1}^{N} n_i(t_{i+1} - t_i).$$

Measuring Utilization and Occupancy (9)

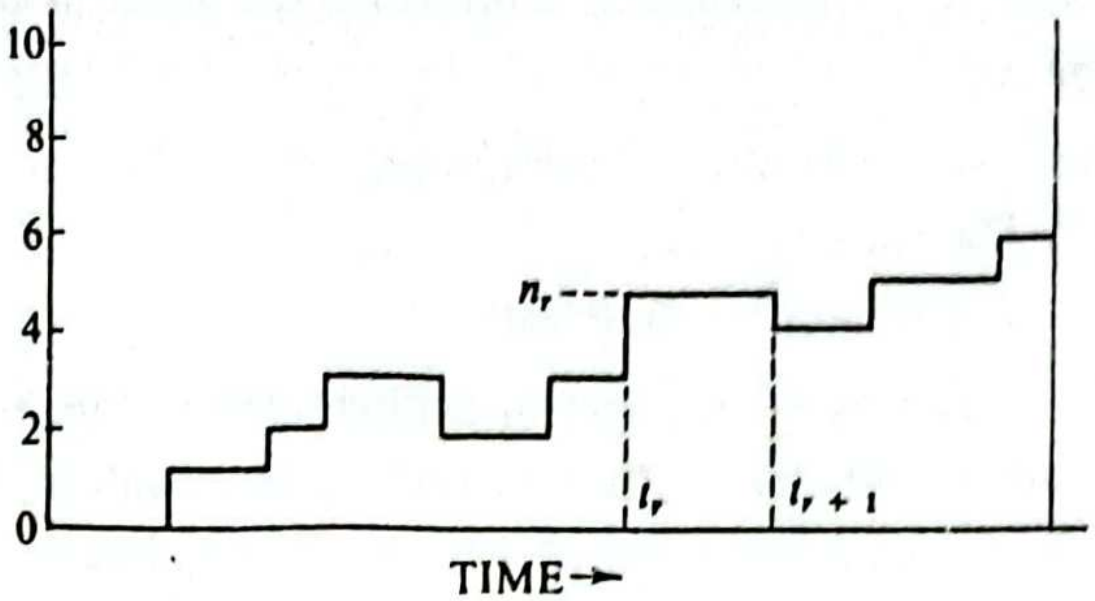

Figure 2: Time History of busy telephone links

Measuring Utilization and Occupancy (10)

If there is an upper limit on the number of entities, as there was a limit on the number of links in the telephone system, the term *occupancy* is often used to describe the average number in use as a ratio to the maximum. Thus, if there are M links in a telephone exchange and the quantity n_i is the number busy in the interval t_i to t_{i+1}, the average occupancy, assuming the number n_i changes N times, is:

$$B = \frac{1}{TM} \sum_{i=1}^{N} n_i(t_{i+1} - t_i).$$

Table of Contents

Recording Distribution and Transit Time (1)

In simulation, **distributions** refer to the statistical patterns or behaviors of variables over time, such as the arrival times of entities, service times, or waiting times in a queue. These distributions can describe how a variable behaves and help predict future system performance. Accurately recording and analyzing these distributions is essential for building realistic models of a system.

Determining the distribution of a variable requires counting how many times the value of the variable falls within specific intervals. A table sets aside locations in which to record the values defining the intervals and to accumulate each count. As each new observation is made, its value is compared with the limits established for the intervals, and 1 is added to the counter for one interval.

Recording Distribution and Transit Time (2)

Normally, the tabulation intervals are uniform in size and the specifications will be for:

1. The lower limit of tabulation,
2. The interval size,
3. The number of intervals.

The meaning of these terms is illustrated in Figure 3.

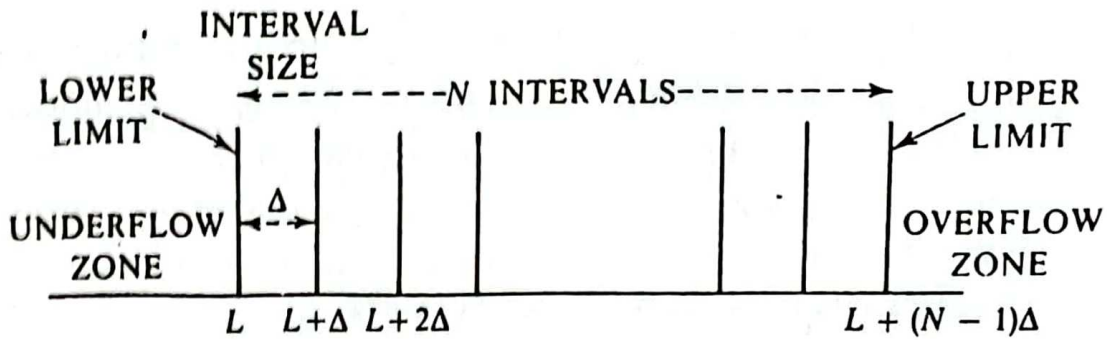

Figure 3: Definition of a distribution table

Recording Distribution and Transit Time (3)

Types of distributions recorded in simulations include:

- Arrival Distributions: The pattern of time between the arrivals of entities into the system. For example, in a call center, the inter-arrival time between calls might follow an exponential distribution (Poisson process).
- Service Time Distributions: The time taken to serve an entity, which might vary depending on the resource or type of service. Common service time distributions include exponential, normal, or triangular distributions.
- Waiting Time Distributions: The time an entity waits before it starts receiving service, which can be influenced by the number of entities in the system.

When recording distributions, it's important to track the following:

Recording Distribution and Transit Time (4)

- The type of distribution used (e.g., exponential, normal, uniform, etc.).
- The parametersof the distribution (e.g., mean and variance for normal distribution).
- The realized values over time during the simulation (e.g., actual service times or inter-arrival times).

Example: In a simulation modeling a customer service system:

- The arrival distribution could follow an exponential distribution with a mean inter-arrival time of 5 minutes. This would mean that on average, customers arrive every 5 minutes, but the actual arrival time could vary around this average.
- The service time distribution might follow a normal distribution with a mean of 10 minutes and a standard deviation of 2 minutes, meaning that most customers will be served within 10 minutes, with some variability.

Recording Distribution and Transit Time (5)

The nature of the random variable being measured determines when the observation would be taken as each entity starts to receive service, so that the times at which the observations are tabulated, or recorded for tabulation, are randomly spaced. To measure the distribution of the number of entities waiting, however, observations would be taken at uniform intervals of time.

Transit time refers to the amount of time an entity spends moving from one part of the system to another. This includes all travel time between stages, processing time at different stations, and any waiting time encountered during the transition. Transit times are particularly useful in systems where entities move between different locations or resources, such as a manufacturing process, logistics network, or supply chain simulation.

To measure transit times, the clock is used in the manner of a time stamp. When an entity reaches a point from which a measurement of transit time is

Recording Distribution and Transit Time (6)

to start, a note of the time of arrival is made. Later, when the entity reaches the point at which the measurement ends, a note of the clock time upon arrival is made and compared with the first time to derive the elapsed interval.

Transit time is the total time an entity spends from the moment it enters the system to the point it reaches its destination. This can include several components:

- Queueing time: The time an entity waits in a queue before it is processed or moves forward.
- Processing time: The time an entity spends being processed at a particular resource (e.g., a server or machine).
- Movement time: The time it takes for an entity to physically move between locations (e.g., moving from one workstation to the next in a manufacturing system).

Recording Distribution and Transit Time (7)

$$\text{Transit Time} = \text{Queueing Time} + \text{Processing Time} + \text{Movement Time}$$

Recording transit times helps to assess the efficiency of the system's flow. Long transit times may indicate bottlenecks, delays, or inefficiencies in the process that could be improved.

Example: In a simulation of a warehouse, the transit time for a package moving through different stages might consist of:

- Queueing time: The time spent waiting for a worker to pick the package, which might vary depending on the workload.
- Processing time: The time spent picking and packing the package, which could be affected by worker efficiency or package size.

Recording Distribution and Transit Time (8)

- Movement time: The time it takes for the package to travel from one stage (e.g., picking) to the next (e.g., packing or shipping).

By recording the transit times of multiple packages, the simulation can identify which stage of the process is causing delays and where improvements can be made to speed up the flow.

Table of Contents

Discrete Simulation Languages (1)

- Discrete simulation languages are specialized programming languages used for modeling and simulating systems that change discretely over time.
- Specifically designed to handle events that occur at distinct points in time, which is typical in discrete-event simulation (DES).
- Discrete event simulations involve systems where the state changes at separate, distinct times, often triggered by specific events (e.g., arrivals of customers, machine breakdowns, etc.).
- In discrete simulation, time progresses in discrete steps, and the system is updated only when an event occurs.
- These simulation languages help in modeling such systems by providing the necessary constructs to define entities, events, queues, resources, and the state transitions.

Discrete Simulation Languages (2)

Features of Discrete Simulation Languages

- Event Scheduling: Discrete simulation languages allow the simulation of events that occur at specific times. These events can trigger changes in the state of the system,, and the language must have the ability to schedule these events.
- Entity Representation: These languages provide features to define entities in the system (e.g., customers, products, machines). Entities are objects that interact within the simulation, and each has specific attributes and states.
- Time Management: The language handles the progression of time by updating the system only when events occur. It helps track time steps, and the simulation proceeds by advancing to the next event.

Discrete Simulation Languages (3)

- Queueing and Resource Management: Discrete simulation languages typically include constructs to model queues (waiting lines) and resources (e.g., servers or machines). These constructs help in managing how entities interact with resources and wait for service.
- Random Variables: These languages include the ability to generate random variables, which are essential for modeling uncertainty and variability in the system, such as arrival times, service times, or processing times.
- Output and Reporting: The language includes methods to track and record statistics and data during the simulation, allowing the user to analyze system performance after running the model.
- Modularity and Reusability: Good simulation languages allow modularity, meaning users can define functions, procedures, and objects that can be reused across different parts of the model.

Discrete Simulation Languages (4)

Examples of Discrete Simulation Languages

- SIMSCRIPT: One of the earliest and most widely used discrete-event simulation languages. SIMSCRIPT allows the simulation of complex systems using a simulation program that defines events, entities, and processes. It has evolved through different versions over the years.
- GPSS (General Purpose Simulation System): GPSS is a popular language for discrete-event simulation. It is a block-structured language designed for simulating systems with queues and resources. It is easy to use for modeling industrial systems such as manufacturing lines and communication networks.
- SLAM II:SLAM (Simulation Language for Alternative Modeling) is another discrete-event simulation language that provides a modeling framework for simulating systems. SLAM II is a later version of the language, known for its flexibility in representing systems and providing useful built-in features for statistical analysis.

Discrete Simulation Languages (5)

- Arena: Arena is a modern discrete-event simulation software that provides a graphical interface for building simulation models. It uses a combination of graphical blocks to represent different system components and is often used for modeling manufacturing, healthcare, and logistics systems.
- Simula: Simula is considered the first object-oriented programming language and is used for both discrete-event and continuous simulation. Simula is powerful for modeling systems that involve objects, processes, and events. It laid the groundwork for later object-oriented languages like C++ and Java.
- AnyLogic: AnyLogic is a simulation software platform that supports discrete-event simulation, agent-based modeling, and system dynamics. It allows users to simulate real-world systems and processes in an interactive and flexible manner.

Discrete Simulation Languages (6)

- QNS (Queue Network Simulation): A discrete-event simulation language used for modeling queueing networks. It provides tools for analyzing performance measures of queueing systems and is commonly used in telecommunications and computer network modeling.

Advantages

- Flexibility: Discrete simulation languages offer flexibility in modeling complex systems and dynamic processes.
- Accuracy: These languages simulate real-world systems accurately by incorporating random variations and detailed event handling.
- Performance Evaluation: They allow for the evaluation of system performance, such as identifying bottlenecks and resource utilization.
- Optimization: They can be used to test different configurations of systems and optimize them for better efficiency.

Discrete Simulation Languages (7)

- User-friendly: Many simulation languages offer graphical user interfaces (GUIs), making it easier for users to build and run models.