**Chapter 01: Basic Simulation Modelling**

Table of Contents

[Performance Evaluation 4](#_Toc87177814)

[Actual System 4](#_Toc87177815)

[Physical and Mathematical Models 5](#_Toc87177816)

[Analytical Solutions and Simulations 6](#_Toc87177817)

[Using Simulations 7](#_Toc87177818)

[Appropriateness of Simulations 7](#_Toc87177819)

[Advantages and Disadvantages of Simulations 8](#_Toc87177820)

[Areas of Simulation Applications 8](#_Toc87177821)

[Systems 9](#_Toc87177822)

[Input-Output Modelling 9](#_Toc87177823)

[Static and Dynamic Systems 10](#_Toc87177824)

[States 11](#_Toc87177825)

[State Space Modelling 11](#_Toc87177826)

[Input-Output Modelling vs State Space Modelling 12](#_Toc87177827)

[Time-Varying and Time-Invariant Systems 13](#_Toc87177828)

[Continuous and Discrete-State Systems 14](#_Toc87177829)

[Deterministic and Stochastic Systems 16](#_Toc87177830)

[Discrete-Time Systems 16](#_Toc87177831)

[Discrete-Event Systems 17](#_Toc87177832)

[Time-Driven and Event-Driven Systems 17](#_Toc87177833)

[Properties of Discrete Event Systems 18](#_Toc87177834)

[Models 19](#_Toc87177835)

[State-Space Models 19](#_Toc87177836)

[Simulation Development Life Cycle 20](#_Toc87177837)

A simulation is the imitation of the operations of a system or process. By running a simulation, we are generating an artificial history of running the system.

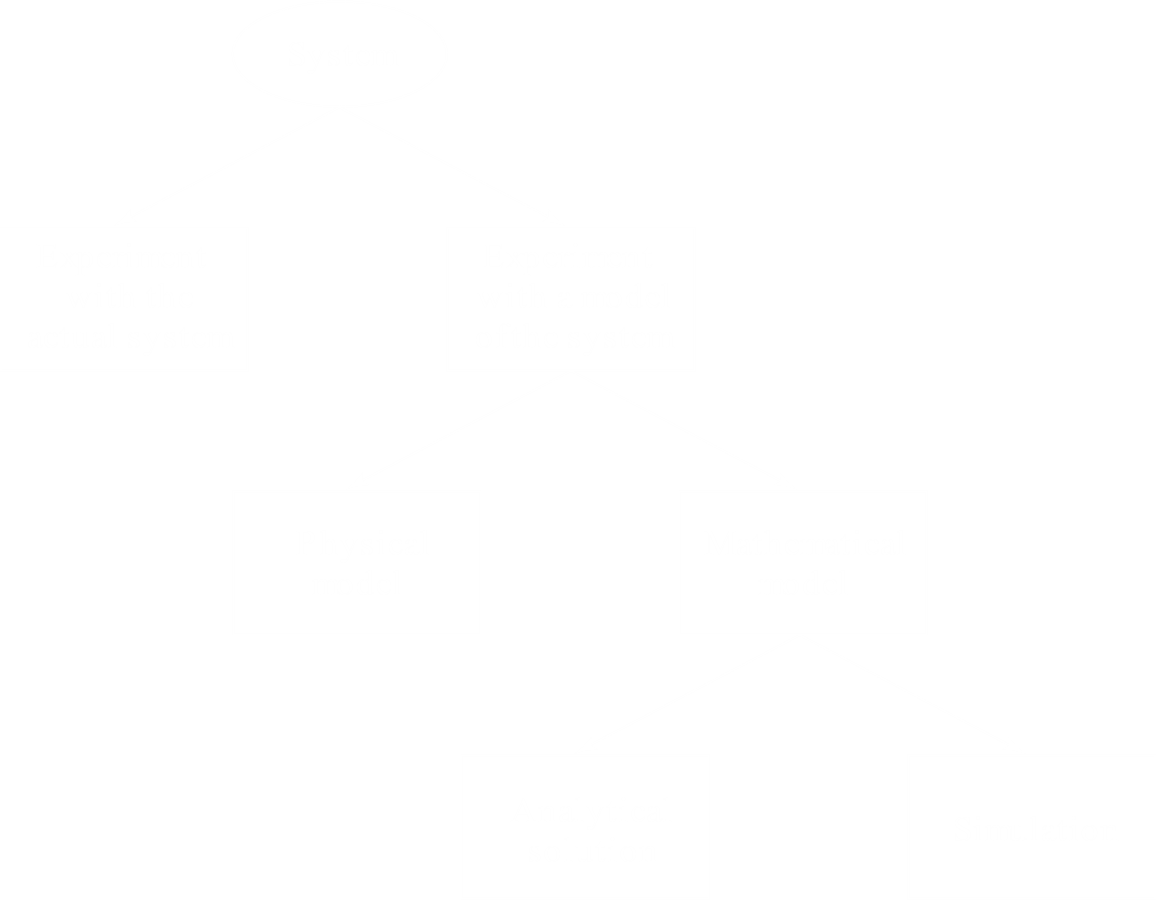
Modelling is the process of creating an alternate representation of the system, either physically or mathematically.

Performance evaluation is the process of understanding the behaviour of a system by analysing its artificial history. By doing this, we can infer its operational characteristics.

Essentially, for a particular model of a system, we need to create a simulation for the model and then analyse the data from the simulation. Then we can determine information about it, such as whether it is fast enough or accurate enough etc.

## Performance Evaluation

Performance evaluation can be done in several ways, which is shown in the diagram below:



### Actual System

Consider that we decide to perform our experiment on the actual system. The good part of this is that no one can question the validity of our results. However, it is not usually feasible to work with the actual system. Perhaps it is too expensive or too disruptive. Consider a situation where we want to check if we can reduce the number of ATMs in the country for a particular bank. Actually, shutting down ATMs could lead to disastrous consequences for the bank. In many cases, the actual system may not even exist yet, or may be difficult to get access to for long enough to experiment with.

### Physical and Mathematical Models

A **physical model** is a replica of the **real system**. It contains at least the **necessary components** of the system related to the actual evaluation. For example, a physical model used to train pilots would contain the cockpit, not the entire plane.

Sometimes however, it can be costly to build a physical model. For example, we cannot create a single IC just to test it. In these cases, we can use **mathematical models**.

Each component of the physical model can be defined using a set of **variables**. The **relationships** between the components can be described with **equations**. The set of equations we obtain will describe the system. A simple example could be , where represents the velocity, represents the time and gives us the distance travelled.

The solution of the mathematical model produces the desired **characteristics** of the system. However, we will need to **assume** a lot of things. Even a very simple real-world system will produce an extremely complicated mathematical model. For example, in the equation we saw above, , we are assuming that friction and air resistance are not present. If we take those into account, the equations get more and more complicated. However, when making assumptions, we still need to do so in a way such that the equations are **very close** to the real-world system.

### Analytical Solutions and Simulations

If we choose to use a mathematical model, we can solve the system **analytically**, or by using a **simulation**.

Solving the mathematical model analytically is easy for **simple models**, but complex ones can require **vast computing resources**. Additionally, a **closed-form solution** may not even exist, meaning we cannot find an equation. However, if an analytical solution is available and is computationally efficient, it is preferred over simulations, since they are more reliable.

A simulation on the other hand, needs to be created **very carefully**. It needs to be **validated** repeatedly to ensure there are no mistakes, since it is very easy to come up with a simulation that seems to mimic the real-world system perfectly but gives **imperfect results** due to faulty assumptions or poor choices. It is best to have an **expert** on the system nearby when developing the simulation.

**Numerical techniques** are used to evaluate the system. Based on the equations created for the model, **inputs** are provided to the simulation and **outputs** are generated. The **behaviour** of the system is studied based on the output. One issue here is that a **huge amount** of data must be given as input, which means we cannot have **real** data. We must **mathematically generate** the input data.

It can sometimes be difficult to tell if the output occurs due to the way the **input** was given, or due to actual **behaviour** of the system. To overcome this, we must run the simulation a **large number** of times.

The funny part is, even though analytical solutions are considered more reliable than simulations, experts suggest using a **simulation** to **validate** the results once an analytical solution has been found.

## Using Simulations

### Appropriateness of Simulations

Simulations are appropriate when we want:

* To know the **internal interaction** of a complex system
* To know the **information**, **organizational** and **environmental** changes
* To experiment with a **new system**
* To verify the **analytical solution**

Simulations are **not** appropriate when:

* The problem can be solved with **common sense**
* The problem can be solved **analytically**
* **Direct experiments** can be easily performed
* The **cost** of developing the simulation exceeds the savings
* **Resources** and **time** are not available
* **Sufficient data** is not available
* The system’s **behaviour** is too **complex**

### Advantages and Disadvantages of Simulations

The advantages of using simulations include:

* **Exploring** new policies, operating procedures, decision rules, information rules, etc. **without disrupting** the existing system
* **Testing** new hardware designs, physical layouts, transportation systems, communication protocols, etc.
* **Understanding hypothesis** about how and why certain phenomena occur
* **Answering** ‘What if…’ questions, especially for new systems.

Disadvantages include:

* Building models requires **special training**
* Simulation results can be **difficult to interpret**
* Modelling and analysing can be **time consuming**
* Many people **insist on using simulations** instead of analytical solutions

### Areas of Simulation Applications

* Manufacturing applications
* Semiconductor manufacturing
* Construction of engineering and project management
* Military applications
* Logistics, supply chain and distribution applications
* Transportation modes and traffic
* Business process simulation
* Healthcare

## Systems

A system is a **collection of components** that each have a specific function. The components regularly **interact** with each other in order to **achieve a goal** that none of the components could have achieved individually. A system is associated with the **specific function** that it is designed to perform.

The above definition of a system is a qualitative definition. Quantitatively, a system can be defined by associating a set of **measurable variables** with the system. By measuring these variables over time, we can describe the **behaviour** of the system. Thus, we have some input variables and some output variables.

### Input-Output Modelling

The quantitative definition describes the behaviour of the system, which means it can **relate** the inputs and outputs. For simplicity, consider that we can create a function that relates the input variables and the output variables. This is the simplest possible modelling process, **input-output modelling**.

The problem here is that input-output modelling does not explore the **internal details** of the system, which limits what we can do with the model. In fact, this model is so simple that we could even create it with trial-and-error.

### Static and Dynamic Systems

A **static** system is one in which **time** has no role. If we consider a system where an input voltage is given and the circuit produces an output voltage, then time has no effect on the system.

A **dynamic** system is one where **time** does have a role. If we hang a mass from a spring and measure the extension of the spring, the extension will vary over time from the moment the mass was hanged. For a dynamic system, **time** is one of the **inputs**.

The problem with having a dynamic system is that the simple **input-output model** we just learnt cannot give us the results. This is because the output depends on **previous inputs** as well, which the input-output model does not take into account. It can only give us solutions where the output is a direct result of the input and nothing else.

For example, if we know the position of the mass hung from a spring at a particular moment, we cannot tell where it will be in the next moment because we also needed to know whether the mass was moving upwards or downwards, i.e. the velocity of the mass. Using just the input, the time, we were unable to determine the output.

To be able to solve this problem, we need to introduce the concept of **states**.

### States

The **state** of a system at any time describes the **behaviour** of the system at that time in some measurable way. The state of a system at time is the information required at to **uniquely determine** the output.

For dynamic systems, we need to have a **variable** or a set of variables that **captures the information** required to determine the state of the system. In the example we used earlier, velocity would be this variable. Such a variable is called the **state variable**. By **updating** the state variable over time, we can determine the output of the dynamic system using just the input.

### State Space Modelling

In **State Space Modelling**, we have **input** and **output** **variables**, but we also have a set of **state variables**, . The models themselves define the **relationship** between , the inputs, , the outputs and .

The set of **equations** required to specify the state, , for all , given , the **initial state**, and the function , where , are called the **state equations**.

State equations of a system usually depend on the system. However, it might have the following form:

The set of **all possible values** that the state can take is called the **state space** of the system, .

The state space model needs to specify the following set of equations:

### Input-Output Modelling vs State Space Modelling

In Input-Output Modelling:

* We take a **black-box** approach
* System **behaviour** is captured by **observing** the output
* The **internal structure** is **unspecified**

In State Space Modelling:

* **States** of the system are defined
* A set of **state equations** are derived
* Output variables are chosen based on **interest**
* Output variables are calculated for different **input conditions** and **system states**

Example

Consider a water tank where water is entering at a rate and exiting at a rate . The state of the system tells us the amount of water stored in the system, measured by its height, .

, where is the maximum height of the tank.

This is because the water level will be if it is already and water goes out faster than it comes in, or if the tank is full and water is coming in faster than it can go out (the tank overflows). If neither of these conditions are true, the water level will change at a rate equal to the difference between the input rate and the output rate.

The complete model is:

Input Variables: and

Output Variables:

### Time-Varying and Time-Invariant Systems

A **time-invariant** system is one in which the input-output relationship is independent of time. The same input always produces the same output, meaning the properties of the system do not change over time.

A **time-variant** system is one in which the output is a function of time, meaning the properties of the system change over time.

Time-variant systems make life difficult for us, even though most real-world systems are time-variant. Thus, we will often assume that a system is time-invariant even though it is not.

### Continuous and Discrete-State Systems

A **continuous-state system** is one in which the **state space**, , is a **continuous** set. The **state equations** are represented using **differential equations**. There are mathematical tools and solution techniques available for continuous-state systems.

A **discrete-state system** is one in which the **state space**, , is a **discrete** set. The **state equations** are usually **logical statements**, e.g. if event occurs and the current state is , the next state will be . Mathematical models for discrete-state systems are usually complex.

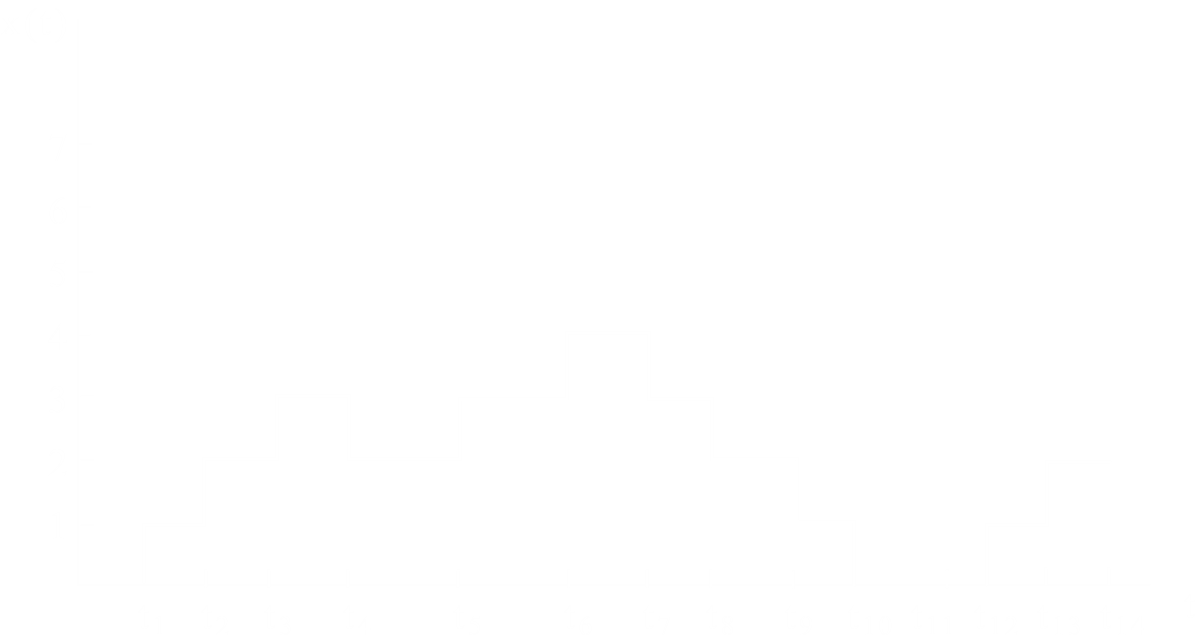
Consider a situation where we have a state variable , which measures the number of products in a warehouse and two input variables, , which is if a product has arrived at time and otherwise, and , which is is a product departs at time and otherwise.

In this scenario, we can have a few logical statements to describe how the state variable is changing:

* If and ,
* If and ,
* Otherwise,

As can be seen, it is not possible to **derive equations** to describe . We can only describe it **logically**. This is because this is a discrete-state system.

Here, denotes the moment just after .



The diagram above shows the values of over time for a particular set of inputs. The path created is called a **sample path**. In a simulation, as different sets of inputs are given, different sample paths will be seen. We can also have sample paths for continuous-state systems.

Discrete-state systems are difficult to solve **mathematically**, but they are very easy to solve **computationally**. In the above example, we could just write a program with a few if-else statements.

### Deterministic and Stochastic Systems

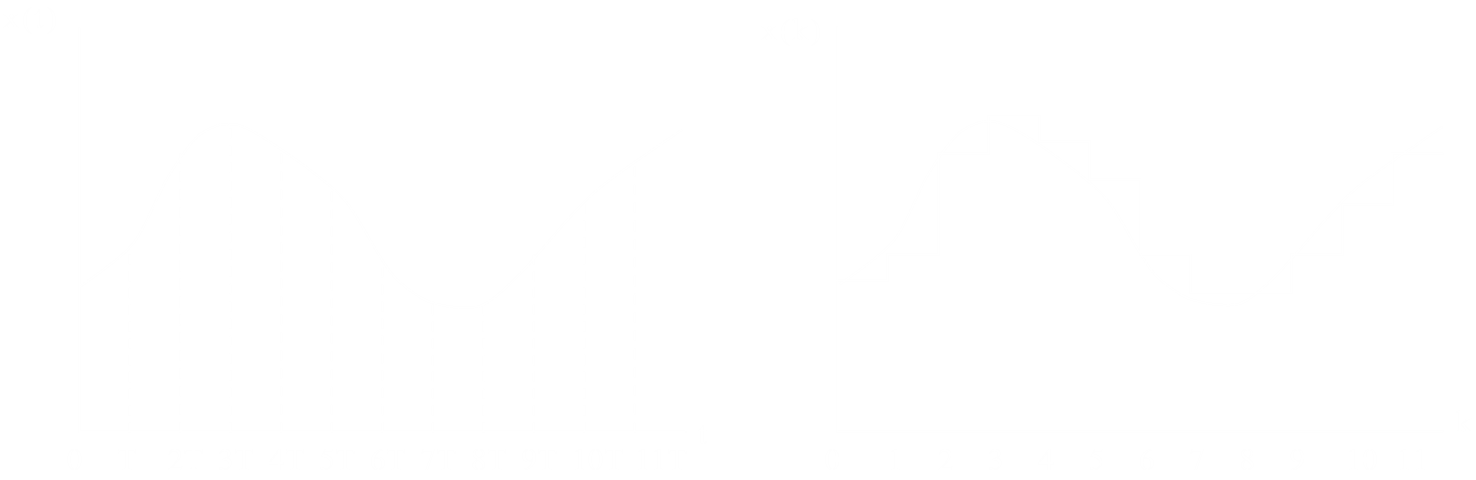
A **stochastic system** is one in which at least one of the **output** variables is a **random variable**. This causes the **state** of the system to be a random or stochastic process. The state at time is a random vector and its **probability distribution** function can be evaluated.

A **deterministic system** is one in which none of the output variables are random. Given , for all , can be evaluated.

### Discrete-Time Systems

In a **discrete-time model**, the time line is a sequence of **intervals** or a sequence of **points**. All the intervals are of **equal length**. Each interval is called a **slot**.

A discrete-time system can still have a **state variable** that is **continuous or discrete**. If it is continuous, we assume that it changes smoothly from one value to another. If it is discrete, we only change the values at the end of each time-slot.



## Discrete-Event Systems

Consider a **state space** that can be described by a **discrete set** like , and **state transitions** are only observed at **discrete** points of **time**. In this scenario, each transition can be associated with special actions, called **events**. For example, pressing a button could trigger an event.

A **discrete-event system** is one which is **stochastic**, meaning at least some of the system state variables are stochastic, **dynamic**, meaning the system state evolves over time, and have system state changes associated with **events** that occur at discrete times.

### Time-Driven and Event-Driven Systems

Continuous-state systems are usually **time-driven**. This means the state changes continuous over time. Discrete-state systems change states are certain points of time, when an **event** occurs.

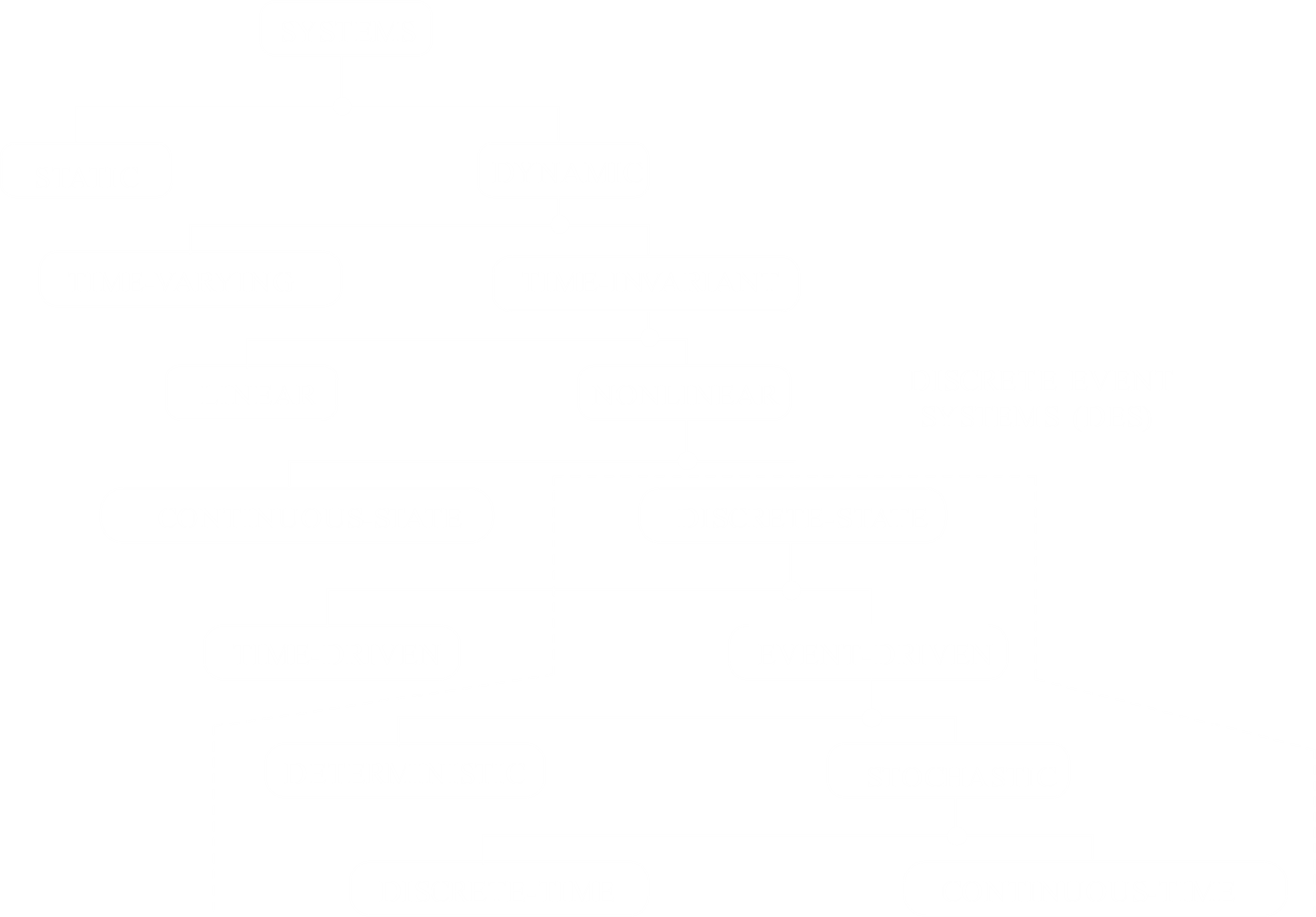
If the system is **time-driven**, then at each clock-tick, either an event occurs, or it does not, i.e. transitions are synchronized with the **clock**. For example, a computer is time-driven. Even if a user clicks a button, the actual event still occurs with the clock cycles.

If the system is **event-driven**, at certain time instances, an event occurs and the state changes. The transitions are **asynchronous** and the events define a time instance of its occurrence. For example, the bell in a shop ringing to indicate a customer has arrived is an event-driven system, since the event does not wait for a clock-tick.

### Properties of Discrete Event Systems

* The **state space** is a **discrete** set.
* The **state transition** mechanism is **event-driven**.
* **State evolutions** depends entirely on the occurrence of **asynchronous discrete events**.
* The **sample path** of the system **jumps** from one state to another whenever an event occurs.
* The **system status** can be represented as a **sequence of timed events**. For example, tells us what event occurs and what time they occur at. In between, nothing happens.

**Discrete-event systems** are a small part of the taxonomy of system classifications.



## Models

A **model** is a simplified representation of some real object of physical situation which serves a particular purpose. It is an **abstract representation** of a system with a number of **approximations** and a **mathematical description** of the system. We have already seen two types of models, **input-output** models and **state-space** models.

### State-Space Models

A **state-space model** is described by a set of **attributes**, . Here,

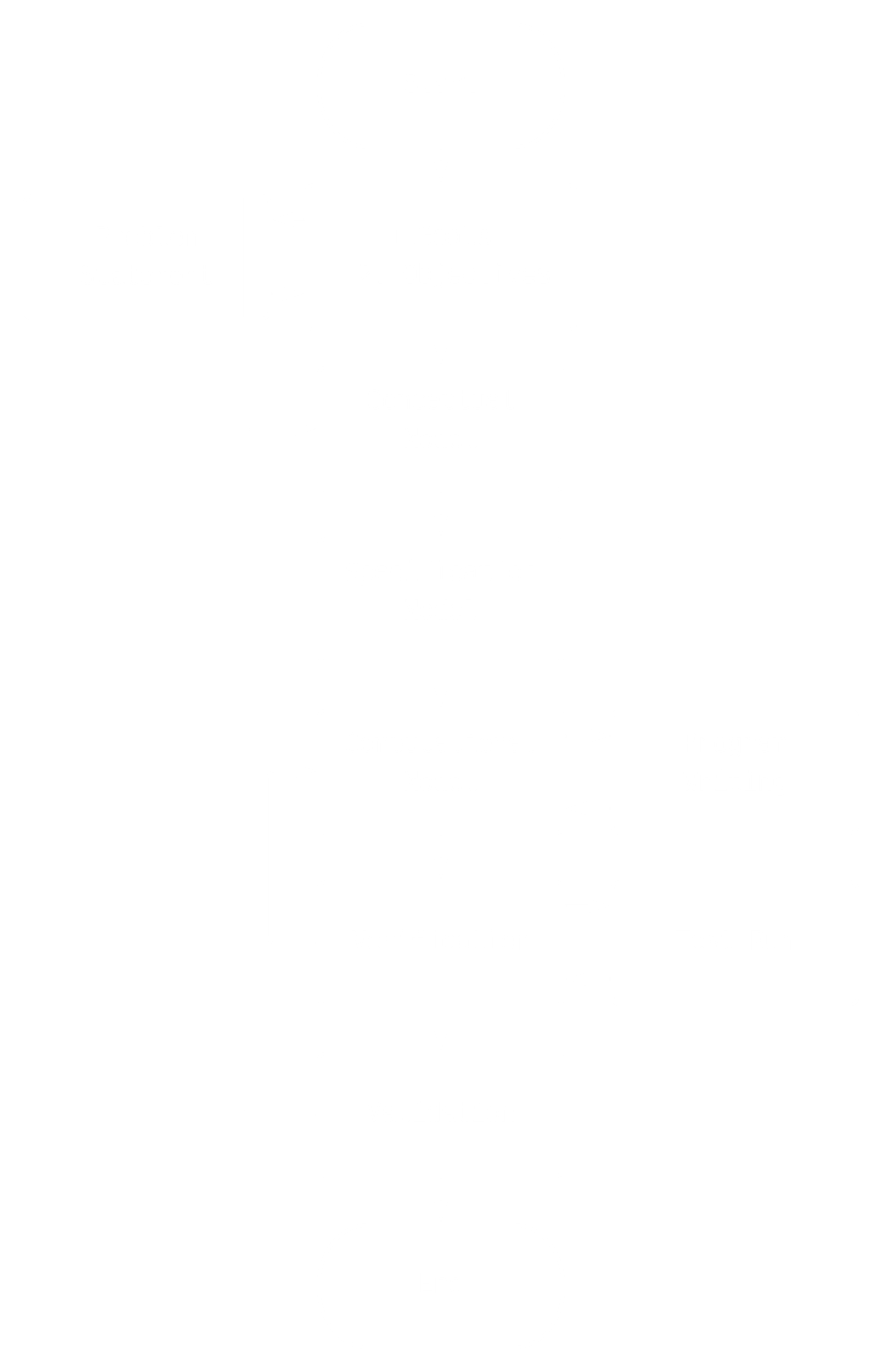
* is the **state space** of the system
* is the set of **events** of the system
* is the **feasible set of events** at state
* is the **initial system state**
* are the **input variables**
* are the **output variables**

We also need

* The set of **state equations**
* The set of **output equations**

In state-space modelling, we need **input data modelling**. We need to **collect** the data from a real system, identify its **statistical properties** and define the **distribution** of the input data. Finally, using the state-space model, we can perform **output data analysis** and make **decisions**.

## Simulation Development Life Cycle



The diagram above describes the **Simulation Development Life Cycle**.

The **Problem Statement** tells us the **purpose** behind the simulation.

The **Goals and Objectives** tells us what we are trying to determine using the simulation. These are usually found by asking **questions**. For example, how many servers are required for satisfactory performance. There may be some **qualitative decisions** here, such as what exactly is ‘satisfactory’. This stage usually helps us identify the **output variables**.

The **Conceptual Model** is informal. We define the system in an **abstract manner**. We can still identify the **state variables**, **input variables** and **events** though, as well as how they are related to one another. We can also remove variables that we think are **loosely connected** or **not connected**, since the fewer variables we have, the easier our work will be. We could even draw a **diagram**.

The **Specification Model** is formal. We write the **state equations** and the **output equations**, identify the **state space** and the **feasible events** at each state and we also do the **input modelling**. Essentially, we have the **state-space model**.

The **Computational Model** consists of the **algorithms**. We convert the model we created into a computer program. There will be a **simulation clock** and we need to have a procedure to **update the clock**. There also needs to be some **initialization procedure**.

We also have a **scheduler**, which maintains the **feasible events**. It **triggers** the next event, runs the procedure to **update the clock** and calls the **event handler** of the next event.

Associated with each event are a set of **actions**, all of which are executed by the **event handler**.

During development of the Computational Model, we need to make decisions about whether to use an **object-oriented design** or a **structured design** and create **diagrams** like UML Diagrams, Use-Case Diagrams, Sequences Diagrams and Flow-Charts for the routines.

The **Verification** stage is where we check whether the **Computational** **Model** we created is consistent with the **Specification Model**. We can test the program with **known data**.

The **Validation** stage is where we check if the model we created is consistent with the **actual system** we were studying. We need to **collect output data** from the real system for **known inputs** and check them against the simulation. This check could be done by an expert. If an expert cannot differentiate between the data, then the program works correctly.