African Virtual University

Applied Computer Science: ITI 4103

# MODELING AND SIMULATION

Luc Einstein Ngend,CSM, M.Sc.,Ph.D

# Foreword

The African Virtual University (AVU) is proud to participate in increasing access to education in African countries through the production of quality learning materials. We are also proud to contribute to global knowledge as our Open Educational Resources are mostly accessed from outside the African continent.

This module was developed as part of a diploma and degree program in Applied Computer Science, in collaboration with 18 African partner institutions from 16 countries. A total of 156 modules were developed or translated to ensure availability in English, French and Portuguese. These modules have also been made available as open education resources (OER) on oer.avu.org.

On behalf of the African Virtual University and our patron, our partner institutions, the African Development Bank, I invite you to use this module in your institution, for your own education, to share it as widely as possible and to participate actively in the AVU communities of practice of your interest. We are committed to be on the frontline of developing and sharing Open Educational Resources.

The African Virtual University (AVU) is a Pan African Intergovernmental Organization established by charter with the mandate of significantly increasing access to quality higher education and training through the innovative use of information communication technologies. A Charter, establishing the AVU as an Intergovernmental Organization, has been signed so far by nineteen (19) African Governments - Kenya, Senegal, Mauritania, Mali, Cote d'Ivoire, Tanzania, Mozambique, Democratic Republic of Congo, Benin, Ghana, Republic of Guinea, Burkina Faso, Niger, South Sudan, Sudan, The Gambia, Guinea-Bissau, Ethiopia and Cape Verde.

The following institutions participated in the Applied Computer Science Program: (1) Université d'Abomey Calavi in Benin; (2) Université de Ougagadougou in Burkina Faso; (3) Université Lumière de Bujumbura in Burundi; (4) Université de Douala in Cameroon; (5) Université de Nouakchott in Mauritania; (6) Université Gaston Berger in Senegal; (7) Université des Sciences, des Techniques et Technologies de Bamako in Mali (8) Ghana Institute of Management and Public Administration; (9) Kwame Nkrumah University of Science and Technology in Ghana; (10) Kenyatta University in Kenya; (11) Egerton University in Kenya; (12) Addis Ababa University in Ethiopia (13) University of Rwanda; (14) University of Dar es Salaam in Tanzania; (15) Universite Abdou Moumouni de Niamey in Niger; (16) Université Cheikh Anta Diop in Senegal; (17) Universidade Pedagógica in Mozambique; and (18) The University of the Gambia in The Gambia.

Bakary Diallo

The Rector

African Virtual University

# Production Credits

## Author

Luc Ngend

## Peer Reviewer

Robert Oboko

## AVU - Academic Coordination

Dr. Marilena Cabral

## Overall Coordinator Applied Computer Science Program

Prof Tim Mwololo Waema

## Module Coordinator

Florence Tushabe

## Instructional Designers

Elizabeth Mbasu

Benta Ochola

Diana Tuel

## Media Team

Sidney McGregor

Barry Savala

Edwin Kiprono

Kelvin Muriithi

Victor Oluoch Otieno

Michal Abigael Koyier

Mercy Tabi Ojwang

Josiah Mutsogu

Kefa Murimi

Gerisson Mulongo

# Copyright Notice



# Supported By

AVU Multinational Project II funded by the African Development Bank.

# Table of Contents

# Course Overview

## Welcome to Modelling and Simulation

This module examines three modern computer modelling and simulation methods, namely Agent-based Modelling (ABM), System Dynamics Modelling (SDM), and Discrete Event Modelling (DEV) utilized to simulate complex systems in health, business, science, and industry environments.

The content has been thought to provide students with the thrust of theoretical background needed to later practically develop, verify, and validate computer simulation models using the software AnyLogic 7 that is one of the rare simulation software tools that support three simulation modeling methods: system dynamics, discrete event, and agent-based modeling and allows you to create multi-method models.

The four laboratories of the module will be developed using the book AnyLogic 7 in Three Days of Ilya Grigoryev (2015). The book is structured around four examples: a model of a consumer market, an epidemic model, a model of a small job shop, and an airport model. It also gives some theory on different modeling methods.

Students are required to attend theoretical activities before taking practical assignments.

## Prerequisites

Students are recommended to have taken the following modules before attending this module:

- Statistics and Probability;
- Linear Algebra;
- Differential Equation and Integral Calculus;
- Data Structures and Algorithms;
- Object-Oriented Analysis, Design and Programming.

## Materials

- **Core Text**

  Grigoryev, L. (2015). AnyLogic 7 in Three Days, A Quick Course in Simulation Modelling, ISBN-13: 978-1508933748

- **Background Text**

  Singh. V, P. (2009). Simulation and Modelling. New Age International

  (P) Ltd., Publishers John D. Sterman. (2000). Business Dynamics:

  Systems thinking and modeling for a Complex world, , McGraw-Hill

## Course Goals

The goal of this module is to equip students with practical knowledge that will enable them to develop real world simulation models from the start to the end. The software used is a modern one that supports different approaches of modelling and simulation, enabling the student to:

- Acquire a working knowledge regarding the art of systems simulation;
- Characterize a system in term of its essential components that are its purpose, stakeholders, constraints, performance requirements, sub-systems, interconnections and environmental context;
- Gain hands-on experience on simulation software;
- Develop skills to use simulation software to construct and execute goal-driven system models;
- Gain the ability to interpret the model and apply the results to resolve critical issues in a real world environment.

## Units

### Unit 0: Pre-Assessment

This Unit aims at testing the prerequisite level of the student. It is a series of questions in the subjects listed as prerequisites of this module.

### Unit 1: Introduction to Modelling and Simulation

This Unit provides the student with a broad understanding of modelling and simulation. It explores the different types of models and applications used in simulation, and details the steps of the simulation model lifecycle. The laboratory of this section will consist of installing AnyLogic.

### Unit 2: Agent-Based Modelling

In this Unit we are going to study computational models that simulate the actions and interactions of autonomous agents (such as people, organizations or groups) with a view to assessing their effects on the system as a whole. In our laboratory, we'll build an agent-based model of a consumer market – one where each consumer will be an agent – to help us understand how a product enters the market.

### Unit 3 : System Dynamics Modelling

This Unit examines a perspective and set of conceptual tools that enable the modeller to understand the nonlinear behaviour of complex systems over time using stocks, flows, internal feedback loops, and time delays and use them to design more effective policies and organizations. In our laboratory, we will build a model that simulates the spread of a contagious disease among a large population. Our sample model will have a population of 10,000 people.

### Unit 4: Discrete Event Modeling

In this Unit we a going to study the process of codifying the behavior of complex systems with highly scholastic variable parameters and constrained resources in order to improve the organization of delivered services. The laboratory of this section will create a discrete-event model that will simulate a small job shop's manufacturing and shipping processes.

## Assessment

Formative assessments, used to check learner progress, are included in each unit.

Summative assessments, such as final tests and assignments, are provided at the end of each module and cover knowledge and skills from the entire module.

Summative assessments are administered at the discretion of the institution offering the course. The suggested assessment plan is as follows:

| # | Designation of Assignments | % of Marks |
|---|---|---|
| 1 | Continuous Assignments (Quizzes and CATs) | 30 |
| 2 | Laboratories | 30 |
| 3 | Final Exam | 40 |
| | | |

## Schedule

| Units | Activities | Estimated time |
|---|---|---|
| Unit 0: Pre-Assessment | Pre-Assessment Test | 5 H |
| Unit 1: Introduction to Modelling and Simulation | Theory Learning and Laboratory | 25 H |
| Unit 2: Agent-Based Modelling | Theory Learning and Laboratory | 30 H |
| Unit 3: System Dynamics Modelling | Theory Learning and Laboratory | 30 H |
| Unit 4: Discrete Event Modeling | Theory Learning and Laboratory | 30 H |

## Readings and Other Resources

### Core Text

- Grigoryev, L. (2015). AnyLogic 7 in Three Days, A Quick Course in Simulation Modelling, ISBN-13: 978-1508933748
- Background Text

- Singh. V, P. (2009). Simulation and Modelling. New Age International (P) Ltd., Publishers
- John D. Sterman. (2000). Business Dynamics: Systems thinking and modeling for a Complex world, , McGraw-Hill
- Fishwick. P, A. (1995). Simulation Model Design and Execution: Building Digital Worlds, Prentice-Hall
- Roberts et al. (1983). Introduction to Computer Simulation: A System Dynamics Modeling Approach, Addison-Wesley
- Banks, J. & Carson, J.C. & Nelson, B. L. (19996). Discrete Event System Simulation," 2nd Edition, Prentice Hall

# Unit 0. Pre-Assessment

## Unit Introduction

This Unit aims at testing your current knowledge and judge whether you are up for taking this module. There are five prerequisites for this module, namely (1) Statistics and Probability; Linear Algebra; (3); Differential Equation and Integral Calculus; (4) Data Structures and Algorithms; (5) Object-Oriented Analysis, Design and Programming. You will have one challenging question for each subject.

## Unit Objectives

Upon completion of this Unit you should be able to:

Know whether you will be up for taking the module of System Modelling and Simulation

## Unit Assessment

Instructions:

- Attend all the questions below
- Grading:
- Maximum marks for each questions are in brackets
- The maximum overall mark is 40.

The Acme Light Bulb Company has found that an average light bulb lasts 1000 hours with a standard deviation of 100 hours. Assume that bulb life is normally distributed. What is the probability that a randomly selected light bulb will burn out in 1200 hours or less? [5]

Determine whether the following matrices are invertible. If the matrix is invertible, compute the inverse. [10]

(a) $A_1 = \begin{bmatrix} 3 & 4 \\ 1 & 2 \end{bmatrix}$

(b) $A_2 = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 5 & 2 \\ 4 & 7 & 4 \end{bmatrix}$

(c) $A_3 = \begin{bmatrix} 1 & 0 & 4 \\ -1 & 1 & -1 \\ -1 & 0 & -3 \end{bmatrix}$

**Solve the ordinary differential equation (ODE)**

Write a program in Java called Fibonacci to display the first 20 Fibonacci numbers F(n), where F(n)=F(n–1)+F(n–2) and F(1)=F(2)=1. Also compute their average. The output shall look like: [10]

- Draw a use case diagram for a ticket distributor for a train system. The system includes two actors: a traveller, who purchases different types of tickets, and a central computer system, which maintains a reference database for the tariff. Use cases should include: BuyOneWayTicket, BuyWeeklyCard, BuyMonthlyCard, UpdateTariff. Also include the following exceptional cases: Time-Out (i.e., traveller took too long to insert the right amount), TransactionAborted (i.e., traveller selected the cancel button without completing the transaction), DistributorOutOfChange, and DistributorOutOfPaper. [10]

# Unit 1. Introduction to Modelling and Simulation

## Unit Introduction

Simulation modelling has been used in a wide range of physical, social sciences and engineering fields, ranging from nuclear fusion to economic forecast to space shuttle design. For different types of situations and systems, different types of models are used. In classifying simulations, there are important distinctions among the types of models that are being simulated, and among the types of program structures that are used to carry out the simulation.

## Unit Objectives

Upon completion of this unit you should be able to:

- Understand what is modelling and simulation;
- Understand why it is important to simulate;
- Differentiate the types of models used in simulation;
- Know different applications of modelling and simulation;
- Know how to develop, verify, and validate computational models of simulation.

### Key Terms

**Model:** A (usually miniature) representation of something; an example for imitation or emulation.

A description of observed behavior, simplified by ignoring certain details. Models allow complex systems to be understood and their behavior predicted within the scope of the model, but may give incorrect descriptions and predictions for situations outside the realm of their intended use.

**Simulation:**

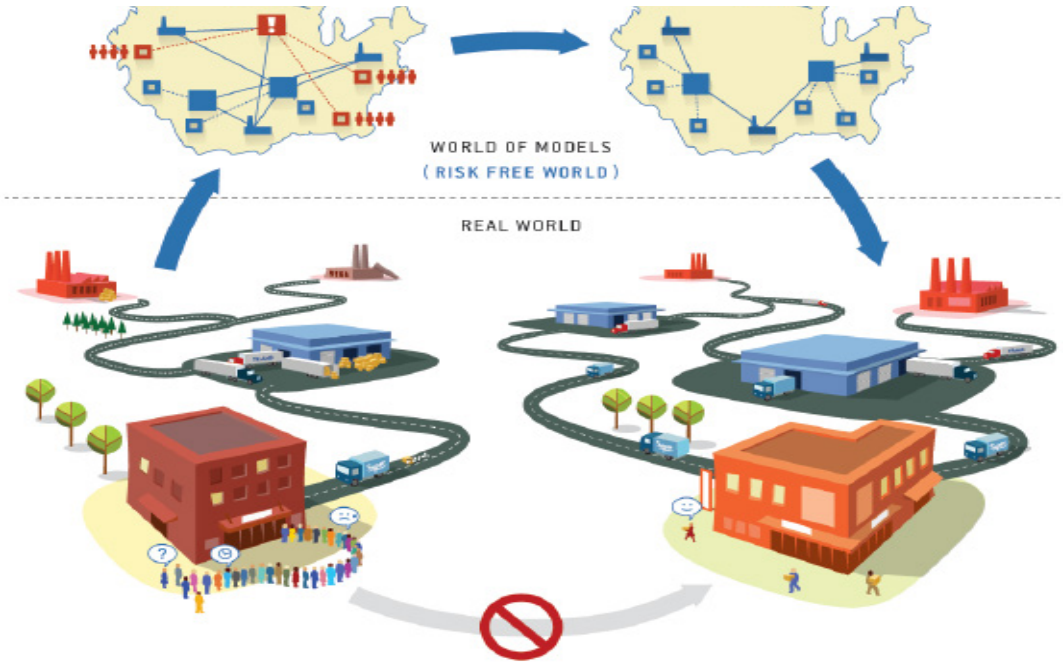
The imitative representation of the functioning of one system or process by means of the functioning of another.

The imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process.

**Modelling :**

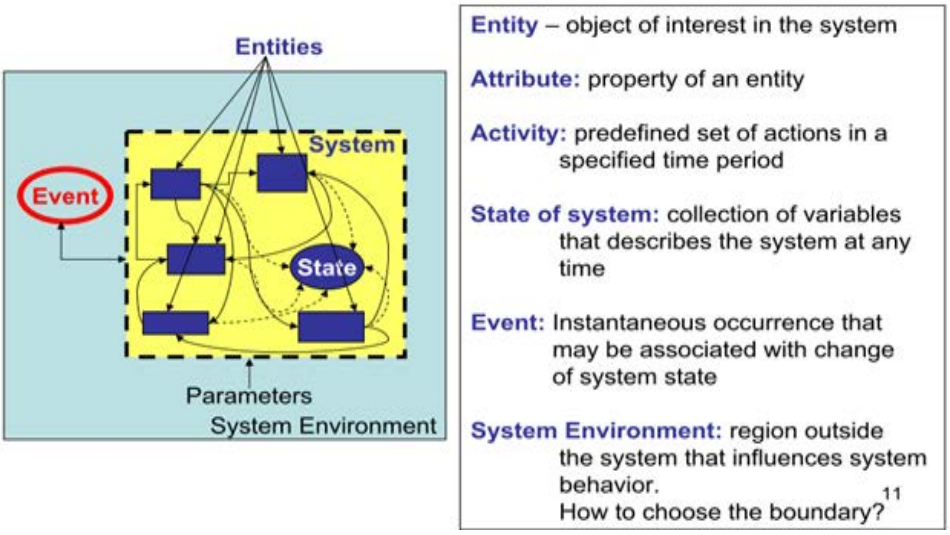
Modeling is a way we can solve real-world problems. In many cases, we can't afford to experiment with real objects to find the right solutions: building, destroying, and making changes may be too expensive, dangerous, or just impossible. If that's the case, we can build a model that uses a modeling language to represent the real system. This process assumes abstraction: we include the details we believe are important and leave aside those we think aren't important. The model is always less complex than the original system.

Figure 1.1: World of Models vs. Real World




Depiction of Other Useful Concepts:

Figure 1.2: Useful Concepts




## Learning Activities

Activity 1: Getting Started!

### Why Simulate?

- It may be too difficult, hazardous, or expensive to observe a real, operational system;
- Parts of the system may not be observable (e.g., internals of a silicon chip or biological system);
- Analyze systems before they are built:
- Reduce number of design mistakes;
- Optimize design;
- Analyze operational systems;
- Create virtual environments for training, entertainment.

### When is simulation appropriate?

- Allows access to system internals that may otherwise not be observable;
- Informational, organizational, and environmental changes can be simulated, and the effect of these changes on the model's behavior can be observed;
- Observations based on simulations give great insight into the system behavior, and it can be determined which variables are most important and how they interact;
- Analytic solutions can be verified;
- Simulation allows to experiment with new designs or policies prior to implementation;

- Can be used for training without the cost and disruption of on-the-job learning;
- The simulated system is so complex, that its interactions can be treated only through simulation.

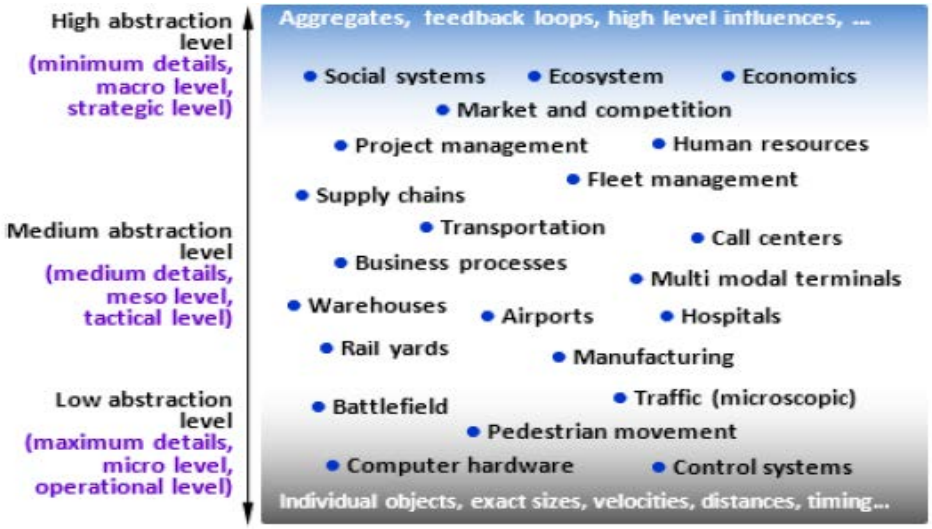
### When simulation is not appropriate?

- Common sense suffices;
- There is no analytical solution;
- It easier to perform direct measurements on a physical system;
- There is a shortage of resources for implementing the simulation;
- There is a shortage of time for getting the desired results;
- Data is lacking for modeling the system and beginning a simulation study;
- There is enough time and personnel to verify and validate the model;
- Managers' expectations are unrealistic;
- The system is too complex to be modeled.

### Modelling and Simulation (M&S) Applications

Simulation modeling has accumulated a large number of success stories in a wide and diverse range of application areas. As new modeling methods and technologies emerge and computer power grows, you can expect simulation modeling to enter an ever-larger number of areas

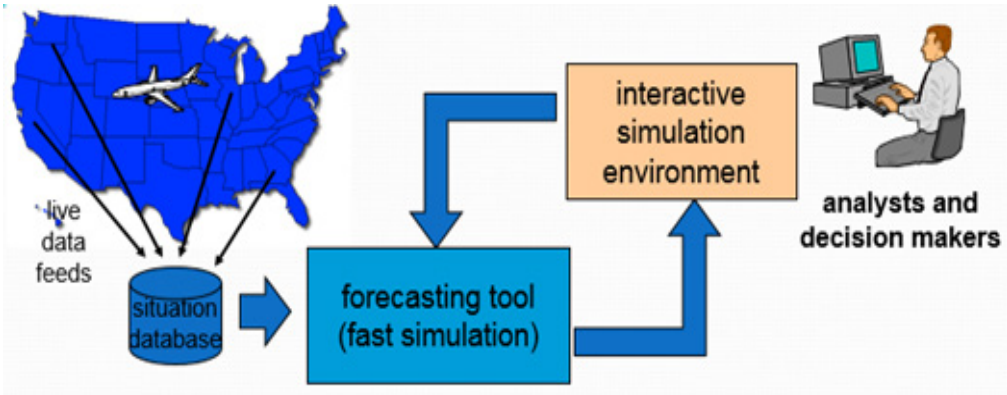
Figure 1.3: Areas of Applications of M&S




The figure above shows a number of simulation applications, all sorted by the abstraction level of the corresponding models.

## M&S Applications in Critical Decision Making

Figure 1.4: Use of M&S in Online Decision Aids




Simulation tool is used for fast analysis of alternate courses of action in time critical situations

- Initialize simulation from situation database
- Faster-than-real-time execution to evaluate effect of decisions

Applications: air traffic control, battle management

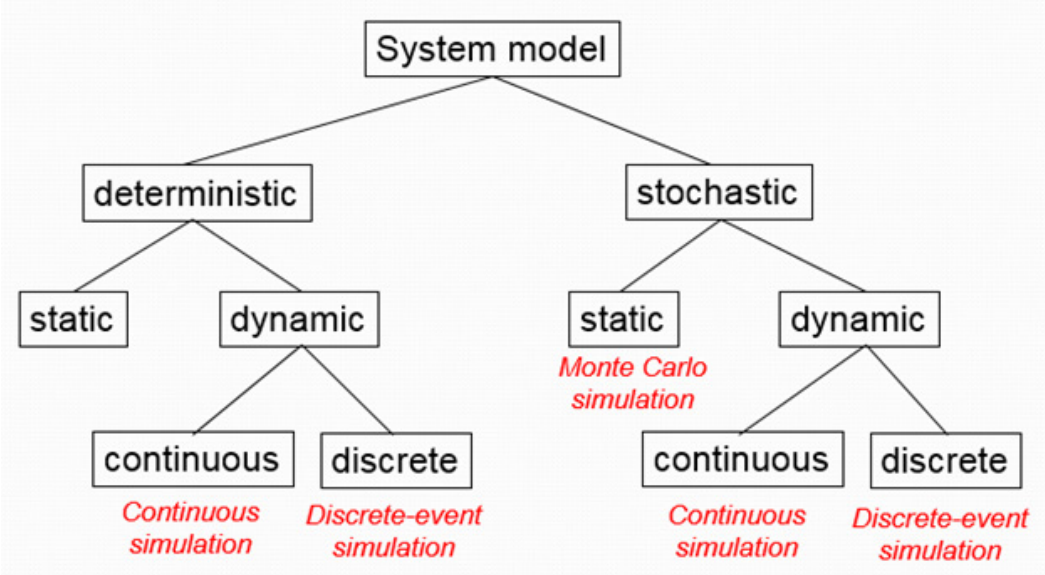
Simulation results may be needed in only seconds

## M&S Applications in Virtual Environments

- Trainings (e.g., military, pilot, medicine, emergency planning),
- Defense: Computer generated forces (CGF):
- Automated forces;
- Semi-automated forces;
- Physical phenomena:
- Trajectory of projectiles;
- Buildings "blowing up";
- Environmental effects on environment (e.g., rain washing out terrain).

### Types of Simulation Models

Figure 1.5: Types of Models Used in M&S




### Stochastic vs. Deterministic Models

Stochastic simulation: a simulation that contains random (probabilistic) elements.

Examples

- Inter-arrival time or service time of customers at a restaurant or store
- Amount of time required to service a customer
- Output is a random quantity (multiple runs required analyze output)

Deterministic simulation: a simulation containing no random elements.

Examples

- Simulation of a digital circuit
- Simulation of a chemical reaction based on differential equations
- Output is deterministic for a given set of inputs

### Static vs. Dynamic Models

Static models: Model where time is not a significant variable

Examples

- Determine the probability of a winning solitaire hand
- Statistical sampling to develop approximate solutions to numerical problems

Note: Static + Stochastic = Monte Carlo Simulation

Dynamic models: Model focusing on the evolution of the system under investigation over time

**Continuous vs. Discrete Models**

Discrete Models:

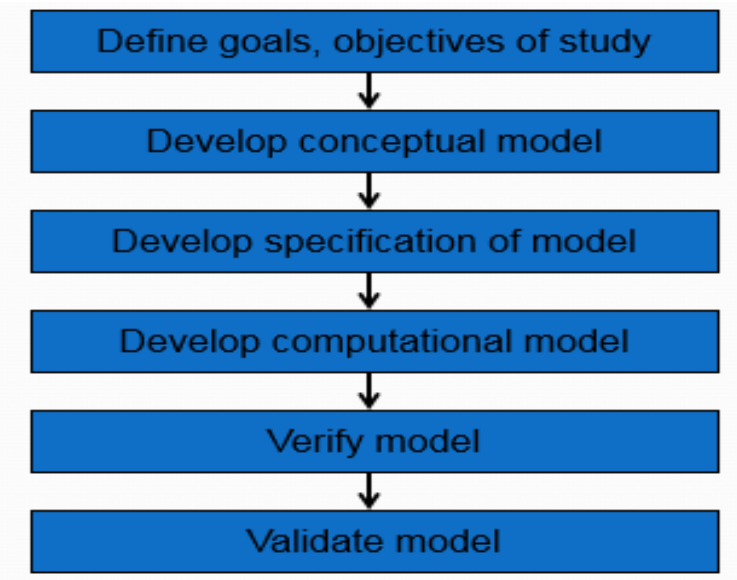
- State of the system is viewed as changing at discrete points in time
- An event is associated with each state transition
- Events contain time stamp

**Continuous Models**

- State of the system is viewed as changing continuously across time
- System typically described by a set of differential equations

**Model Development Lifecycle**

Figure 1.6: Steps of Models Development




**Define Goals and Objectives of Study**

This step answers the question: What does you (or the customer) hope to accomplish with the model?

- It might be:
- Predict the weather
- Train personnel to develop certain skills (e.g., driving)
- Optimize a manufacturing process or develop the most cost effective means to reduce traffic congestion in some part of a city

**Note**: Goals may not be known when you start the project! (One often learns things along the way)

### Develop Conceptual Model

This step involves creating an abstract (i.e., not directly executable) representation of the system:

- What should be included in model?  What can be left out?
- What abstractions should be used
- What metrics will be produced by the model?
- Appropriate choice depends on the purpose of the model

### Develop Specification of Model

Detailed specifications of the model are developed:

- Collect data to populate model
  *Traffic example: Road geometry, signal timing, expected traffic demand, driver behavior*
- Development of algorithms necessary to include in the model
- Example: Path planning for vehicles

### Develop Computational Model

At this stage, the model is implemented in the system:

- Executable simulation model
- Software approach:
- General purpose programming language
- Special purpose simulation language
- Simulation package
- Other (non-functional) requirements are taken into consideration:
- Performance
- Interoperability with other models/tools/data

### Verify Model

- Did I build the model right?
- Does the computational model match the specification model?
- Largely a software engineering activity (debugging)
- Not to be confused with correctness (see model validation)!

### Validate Model

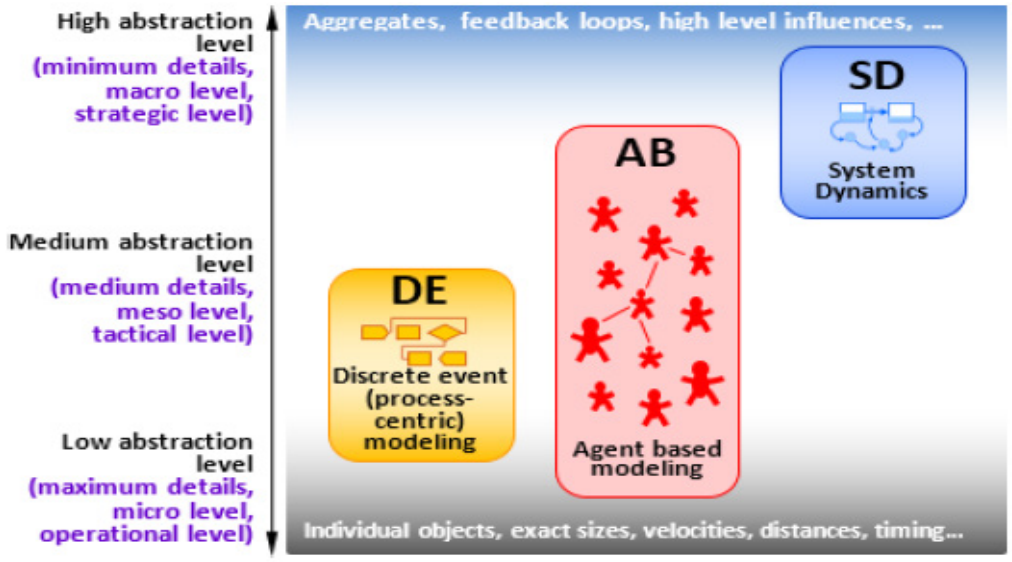
- Did I build the right model?
- Does the computational model match the actual (or envisioned) system?
- Typically, compare against
- Measurements of actual system
- An analytic (mathematical) model of the system
- Another simulation model
- By necessity, always an incomplete activity!
- Often can only validate portions of the model
- If you can validate the simulation with 100% certainty, why build the simulation?

### M&S Methods

In modelling and simulation, a method is a framework we use to map a real world system to its model. You can think of a method as a type of language or a sort of "terms and conditions" for model building. There are many methods used to model and simulate systems; the modern ones we have chosen for this module are the following:

- Systems Dynamics;
- Discrete Event Simulation;
- Agent Based Modelling.

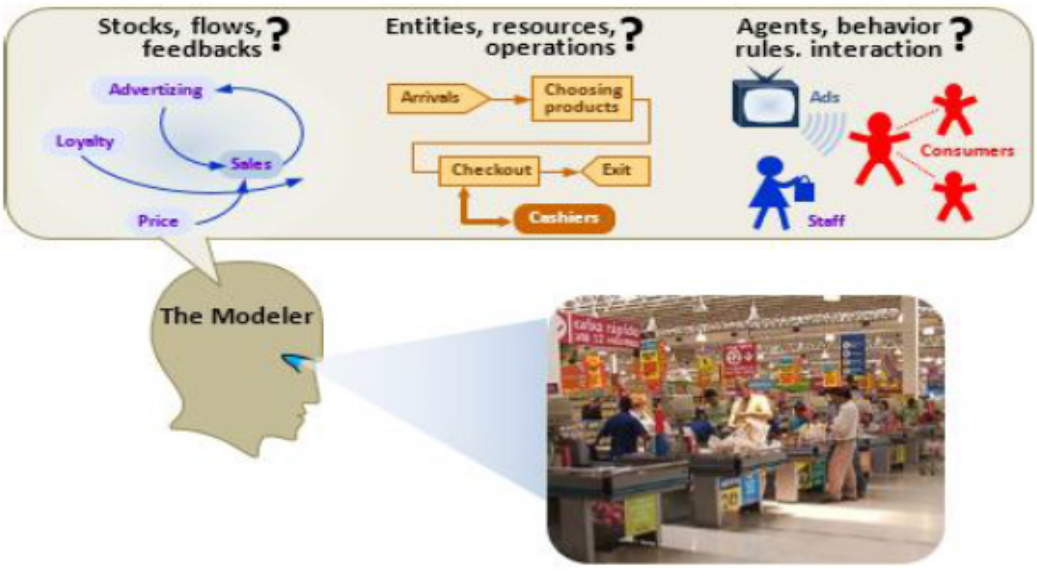
*Figure 1.6: Modern Methods in M&S*




Each method serves a specific range of abstraction levels. System dynamics assumes very high abstraction, and it's typically used for strategic modeling. Discrete event modelling supports medium and medium-low abstraction. In the middle are agent based models, which can vary from very detailed models where agents represent physical objects to the highly abstract models where agents represent competing companies or governments.

You should always select your method after you've carefully considered the system you want to model and your goals. In the figure below, the modeller's problem will largely determine how they model a supermarket. They could build a process flowchart where customers are entities and employees are resources, an agent based model where consumers are agents who are affected by advertising, communication, and their interactions with agents and employees, or a feedback structure where sales are in the loop with ads, quality of service, pricing, and customer loyalty.

Figure 1.7: Abstraction Objects of a Supermarket



You may also find that the best way to model the different parts of a system is to use different methods, and in these situations a multi-method model will best meet your needs (Borshchev, 2013).

## Activity 2: Laboratory 1: Installing and Activating AnyLogic

### Laboratory Objective:

We will use AnyLogic 7 software for all our laboratories. The objective of this lab is to have it install in your machine and get it for use.

### Instructions:

- Download AnyLogic and install it on your machine

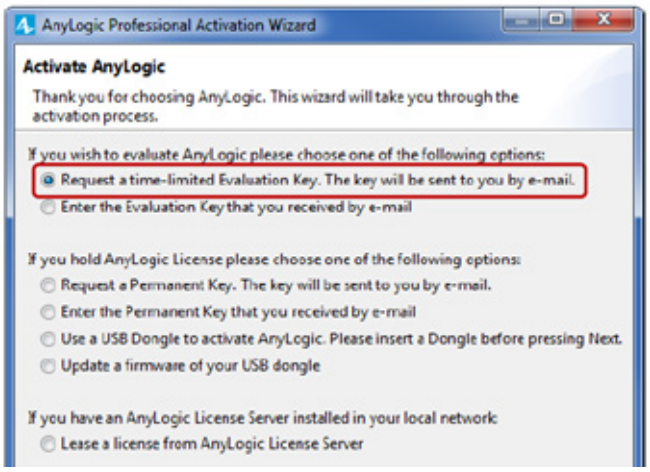
### Grading:

- The maximum mark for this laboratory is 10

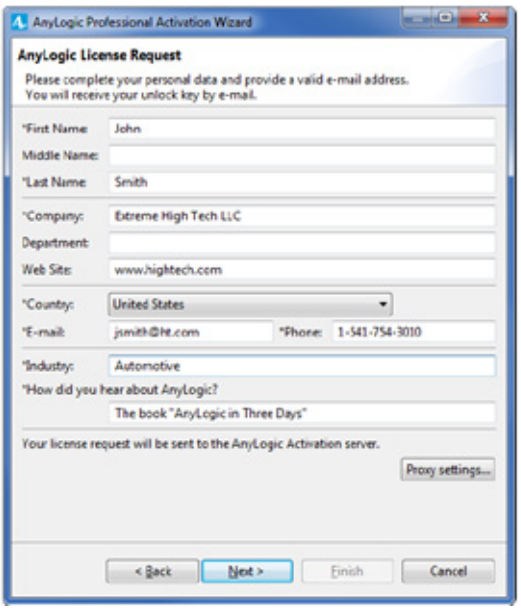
**Time Required**: 3 Hours

AnyLogic 7 Professional's wizard-driven installation process is simple and straightforward. Download AnyLogic 7 from www.anylogic.com, and then use the following steps to install it:

1. Start AnyLogic. If it is not activated with a personal unlock key yet, the AnyLogic Activation Wizard will be displayed automatically.
2. On the Activate AnyLogic page, select Request a time-limited Evaluation Key. The key will be sent to you by e-mail, and then click Next.

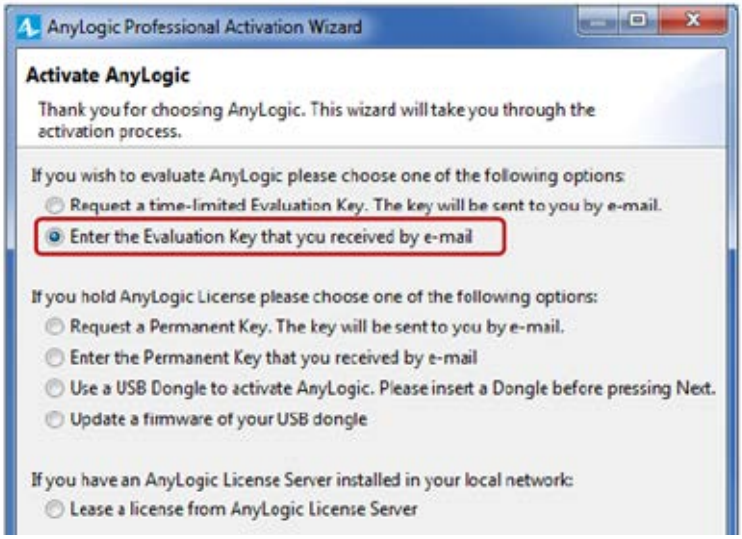



3. On the **AnyLogic License Request** page, provide your personal information and then click **Next**.

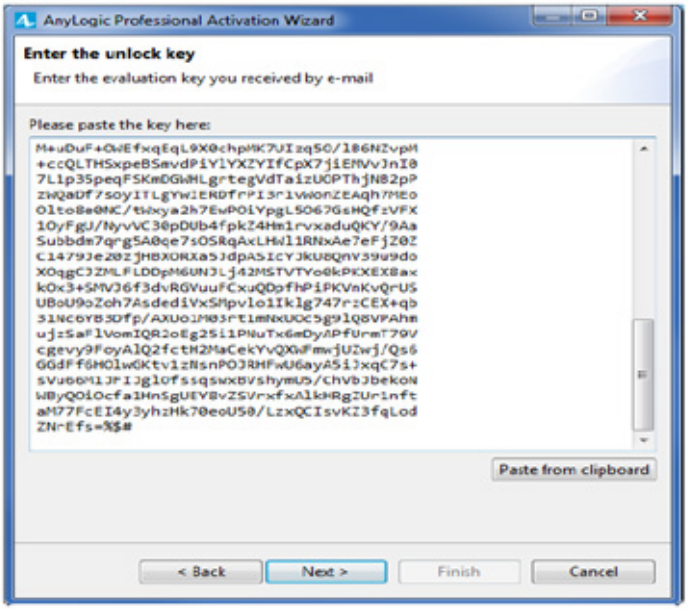



You'll receive a confirmation shortly after you send your request, and you'll receive your evaluation key in a separate e-mail.

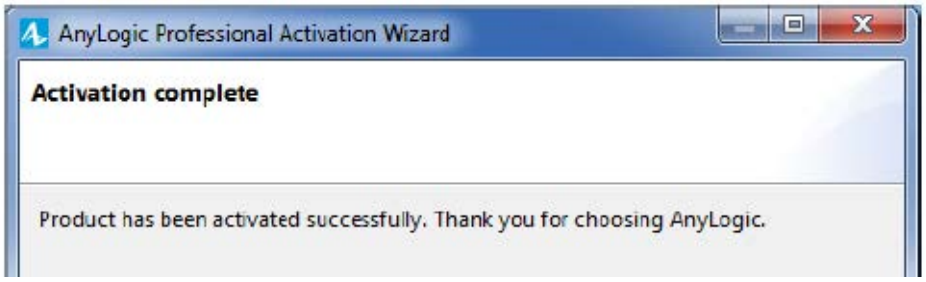
4. After you receive your activation key, open the AnyLogic activation wizard, select **Enter the Evaluation Key that you received by email** on the first page, and then click **Next**.




5. Copy the received activation key from the email message you received, paste it into the **Please paste the key here** field, and then click **Next**.




6. You should see a message that informs you the product has been activated successfully.




7. Click **Finish**.

You've completed AnyLogic's activation process, and you can start developing your first model.

## Unit Summary

Modeling and simulation (M&S) is an important, widely used technique with a wide range of applications. It is sometimes too difficult or expensive to observe a real and operational system, so transforming it into a model a simulating its behavior can be easier and cost cutting. It is also important to analyze in-depth some complex systems before they are built. This helps foresee all the facets of the future system before engaging resources (money, human resource, time, effort, etc.) to implement it. M&S helps assure and reassure that the system will successfully be built and meet the expected deliverables.

It is however important to note that M&S is not a panacea to all problems, i.e. there are situations when it is not appropriate to simulate. Before you engage into simulation, make sure that an analytical solution is feasible; it is more difficult to perform analysis on the physical system as compared to the model; data, time, and resources are available for modelling and simulation.

When you have decided to model and simulate, make sure that you choose the right techniques and methods, and that you thoroughly follow the development lifecycle.

## Unit Assessment

**Instructions:**

- Attend all the questions below

## Grading

- Maximum marks for each questions are in brackets
- The maximum overall mark is 40.

1. Briefly discuss why the model cannot be separated from the simulation. [5]
2. How does the model differ with the reality? [5]
3. When is it important or not to simulate on models? [5]
4. Give an example (different from the ones given in the Activity) for each sub type of probabilistic and deterministic models of simulation. [5]
5. What is the difference between model verification and model validation? [5]
6. List 3 software used in M&S and give their strengths and weaknesses. [5]
7. Briefly explain the modelling approaches used by Systems Dynamics, Agent Based and Discrete Event Modelling. [5]
8. Considering these processes: Account Opening, Deposit on Account, and Withdrawal on Account. With the help of a Use Case Diagram, model the use cases of each process, the communications between their use cases, and the interactions of the system with the environment (actors). [5]

## Unit Readings and Other Resources

### Core Text

- Grigoryev, L. (2015). AnyLogic 7 in Three Days, A Quick Course in Simulation Modelling, ISBN-13: 978-1508933748

## Background Text


- Singh. V, P. (2009). Simulation and Modelling. New Age International (P) Ltd., Publishers
- John D. Sterman. (2000). Business Dynamics: Systems thinking and modeling for a Complex world, , McGraw-Hill
- Fishwick. P, A. (1995). Simulation Model Design and Execution: Building Digital Worlds, Prentice-Hall
- Roberts et al. (1983). Introduction to Computer Simulation: A System Dynamics Modeling Approach, Addison-Wesley
- Banks, J. & Carson, J.C. & Nelson, B. L. (19996). Discrete Event System Simulation," 2nd Edition, Prentice Hall

# Unit 2. Agent-Based Modelling and Simulation

## Unit Introduction

Agent-based modeling is a relatively new method compared to system dynamics and discrete event modeling. In fact, agent-based modeling was largely an academic topic until simulation practitioners began using it some 15 years ago.

It was triggered by:

- A desire to gain deeper insights into systems that traditional modeling approaches don't capture well
- Advances in modeling technology made possible by computer science, such as object oriented modeling, UML, and state charts

The rapid growth of CPU power and memory. Agent-based models are more demanding than system dynamics and discrete event models.

Agent-based modeling offers a modeller another way to look at the system:

You may not know how a system behaves, be able to identify its key variables and their dependencies, or recognize a process flow, but you may have insights into how the system's objects behave. If that's the case, you can start building your model by identifying the objects (agents) and defining their behaviours. Afterward, you may connect the agents you've created and allow them to interact or put them in an environment which has its own dynamics. The system's global behaviour emerges from many (tens, hundreds, thousands, millions) concurrent individual behaviours.

There's no standard language for agent-based modeling, and an agent-based model's structure comes from graphical editors or scripts. There are many ways to specify an agent's behaviour. Frequently agent has a notion of state and its actions and reactions depend on the state; then behaviour is best defined with state charts. Sometimes behaviour is defined in rules executed upon special events.

In many cases, the best way to capture the agent's internal dynamics is to use system dynamics or a discrete event approach, and then place a stock and flow diagram or a process flowchart inside an agent. Similarly, outside agents the dynamics of the environment where they live is often naturally modelled using traditional methods. It's why many agent-based models are multi-method models.

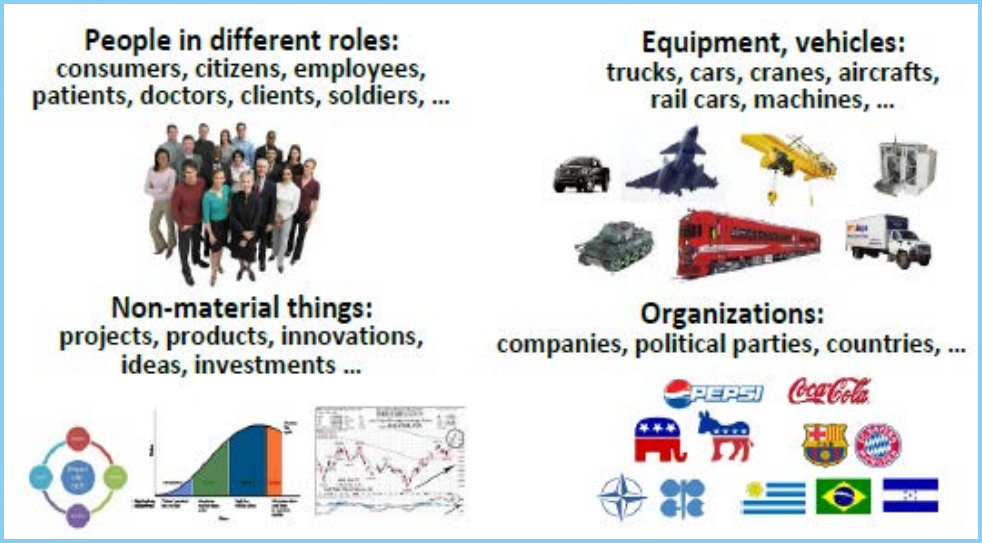
## Unit Objectives

Upon completion of this unit you should be able to:

- Understand what is Agent-based Modelling (ABM)
- Know in which situations the use of ABM is appropriate
- Model and simulate a scenario using ABM techniques and algorithms

### Key Terms

**Agent:** Agents in an agent-based model may represent very diverse things: vehicles, units of equipment, projects, products, ideas, organizations, investments, pieces of land, people in different roles, etc




Academics still debate which properties an object should have to be an "agent": proactive and reactive qualities, a spatial awareness, an ability to learn, social ability, "intellect", etc. In applied agent-based modeling, however, you'll find all kinds of agents: some communicate while others live in total isolation, some live in a space while others live without a space, and some learn and adapt while others never change their behaviour patterns.

Here are some useful facts to ensure you aren't misguided by academic literature or the various theories of agent-based modeling:

Agents aren't cellular automata. Agents don't have to live in discrete space (like the grid in The Game of Life, ("The Game of Life", n.d.)), and space isn't part of many agent-based models. When you need to represent space, it's typically continuous such as a geographical map or a facility floor plan.

Agents aren't necessarily people. Anything can be an agent: a vehicle, a piece of equipment, a project, an idea, an organization, or even an investment. A model of a steel converter plant where each machine is modelled as an agent and their interactions produce steel is an agent-based model.

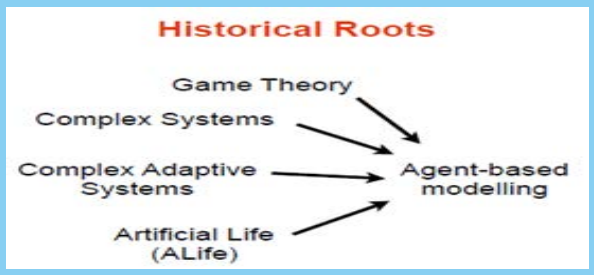
An object that seems to be absolutely passive can be an agent. You could model a single pipe segment in a larger water supply network as an agent and then associate maintenance and replacement schedules, costs, and breakdown events with it.

An agent-based model can have many or few agents. The model can also have one or many types of agents.

There are agent-based models where agents don't interact. Health economics, as an example, uses alcohol use, obesity, and chronic disease models where individual dynamics depend only on personal parameters and, sometimes, on the environment

## 1. Historical Roots of ABM

*Figure 3.2: Historical Roots of ABM*



## Learning Activities

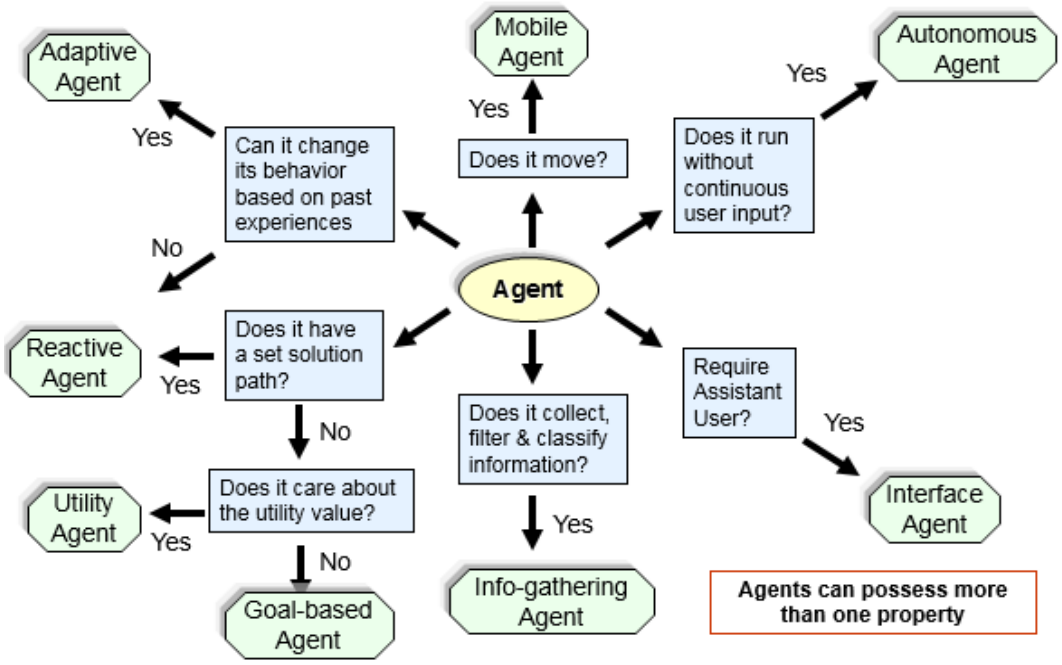
Activity 1: Modelling Autonomous and Interacting Agents

### What is ABM?

- Simulation modeling technique where a system is modeled as a collection of agents and the relationships between them.
- Agents individually asses its situation in the environment and make decisions on the basis of a set of rules.

### Agents Types

*Figure 3.3: Types of Agents*




Types of Agents in Action

### Traffic Control:

- **Reactive**: Police (enforce laws of road)
- **Info-gathering**: Media (informs the public of traffic and accidents in major areas)**Autonomous**: Disruptors (weather / accidents)
- **Goal-based**: City Planners (would like the least number of accidents and greatest amount of flow through parts of town)
- **Adaptive**: Drivers (may avoid roads that are known to be overcrowded during certain times of day)
- **Utility**: Drivers (would like to minimize drive time / distance)

## Common Characteristics of Agents

Self-contained, modular, and uniquely identifiable individual;

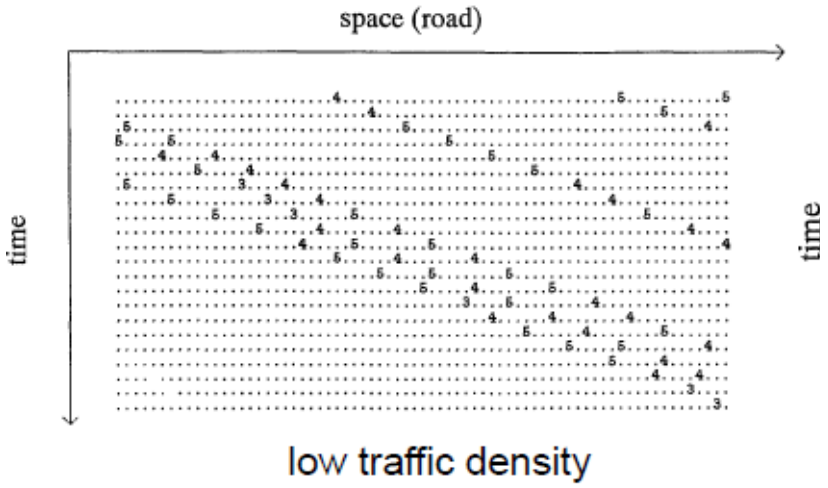
- Autonomous and self-directed; Have dynamic state (that varies over time);
- Are social, i.e., has dynamic interactions with other agents that influence their behaviors;
- Adaptive: adapt behavior based on previous experiences (learning and memory); learning can also occur in a population through (evolutionary) selection;
- Goal-directed: try to achieve goals or maximize objectives;
- Heterogeneous: not like indistinguishable atoms but with different properties, goals, experiences, etc.

## Examples of Agent-Based Systems

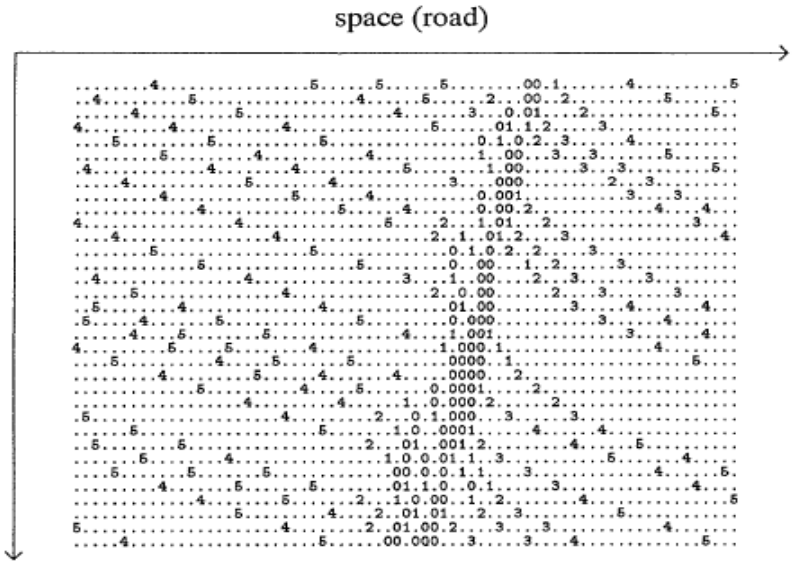
### *Example 1: A Cellular automation model for freeway traffic (Nagel & Schreckenberg,1992)*

Abstract: We introduce a scholastic discrete automation model to simulate freeway traffic. Monte-Carlo simulations of the model show a transition from laminar traffic flow to start-stop-waves with increasing vehicle density, as is observed in real freeway traffic. For special cases, analytical results can be obtained (Nagel & Schreckenberg, 1992)

- Acceleration: If the velocity v of a vehicle is lower than $[\![ v ]\!]\_max$ and if the distance to the next car ahead is larger than v+1, the speed is advanced by one [v→v+1].
- Slowing down (due to other cars): if a vehicle a site i sees the next vehicle at site
- i+j (with j≤v), it reduces its speed to j-1[v→j-1].
- Randomization: with the probability p, the velocity of each vehicle (if greater than zero) is decreased by [v→j-1].
- Car motion: each car is advanced v sites




low traffic density

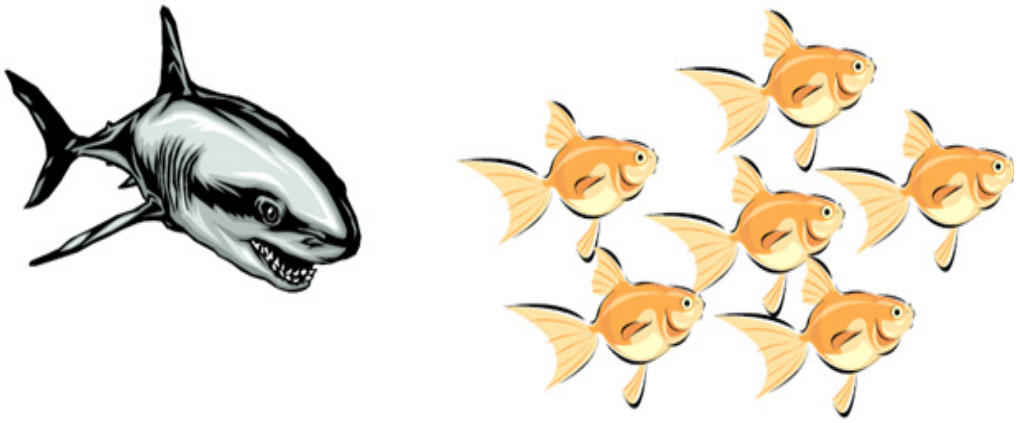



high traffic density

### Advantage of schooling and flocking

To Richard Dawkins (The Selfish Gene): A simple strategy for a predator animal is to chase the closest prey animal in its vicinity. This is reasonable because it expends the least amount of effort for the predator.

Also: fish in school may be harder to find

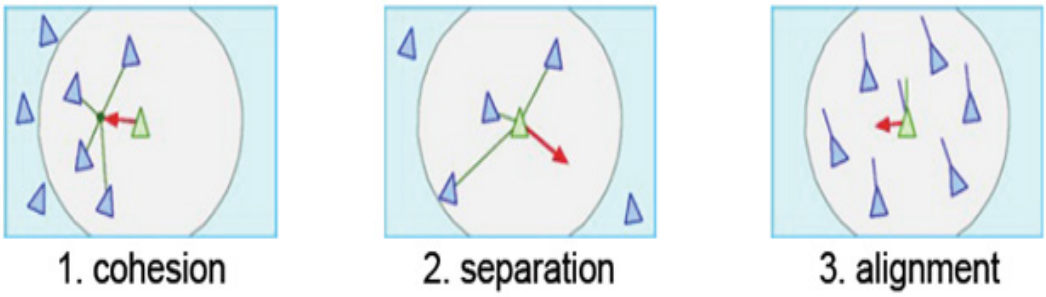


If you have buddies all around you, you are pretty save!

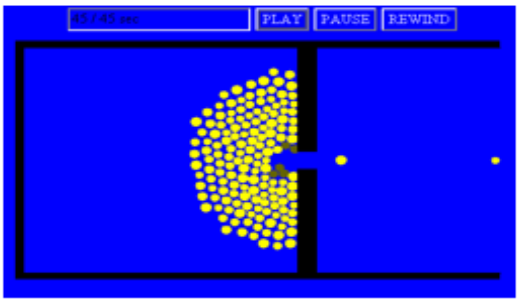
To Craig Reynolds (1986), there are a few local rules sufficient to produce birds' flocking.

Each individual follows these (informal) rules:

- Fly towards the centre of mass of neighbours
- Keep small distance away from other objects (including other boids).
- Match velocity with near boids.

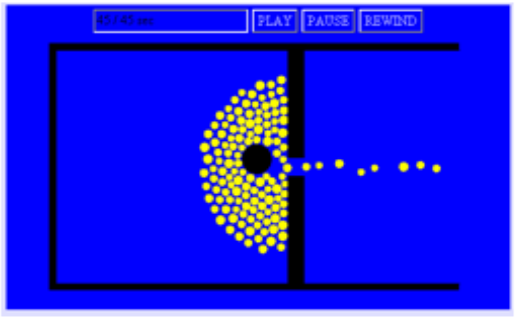



*Example 3: Evacuation*




**Stampede Situation**

- People become injured when they collide at a certain speed
- As a consequence, leaving the room becomes difficult.

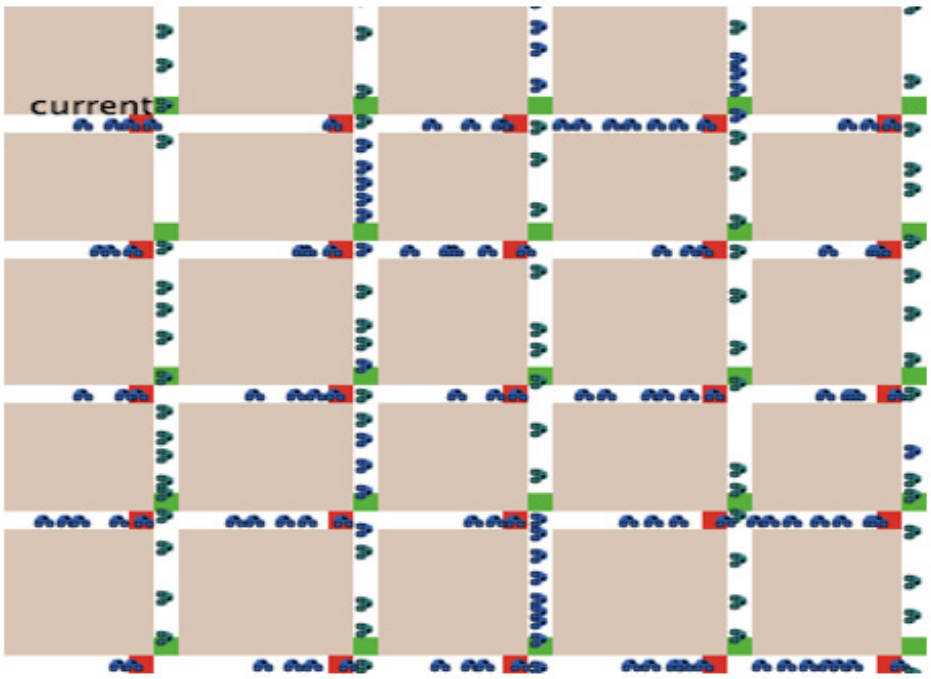



**Stampede Situation w/ Column**

- A column in front of the door can avoid injuries.
- It can increase the outflow well with less / no injured people

Helbing, Farkas, Vicsek

*Example 4: Traffic Control*



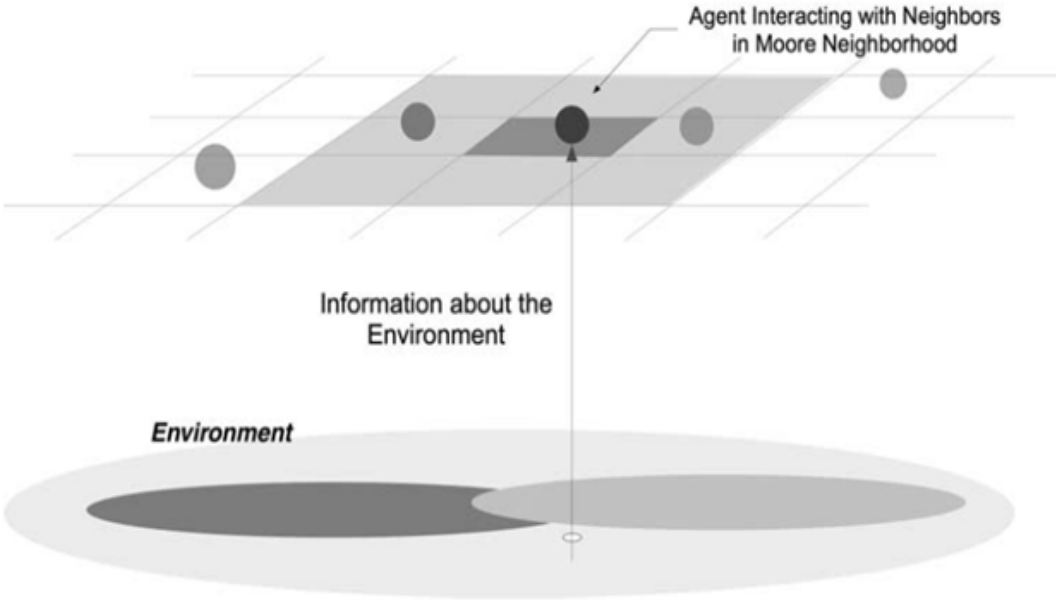

**Modern Applications of ABM**

- Economics (stock markets, supply chains, consumer purchasing behaviour, agent-based computational economics", ...)
- Biology/medicine (tissue growth, modelling the immune system, spread of epidemics, ...)
- Social sciences (traffic simulations, crowd behaviour, tourism, fall of ancient civilizations, modeling engagement of forces on the battle field, ...)
- Ecology (interactions between individuals of different species, predator-prey dynamics, etc.
- Physics (self-assembly of nano-materials, ...)

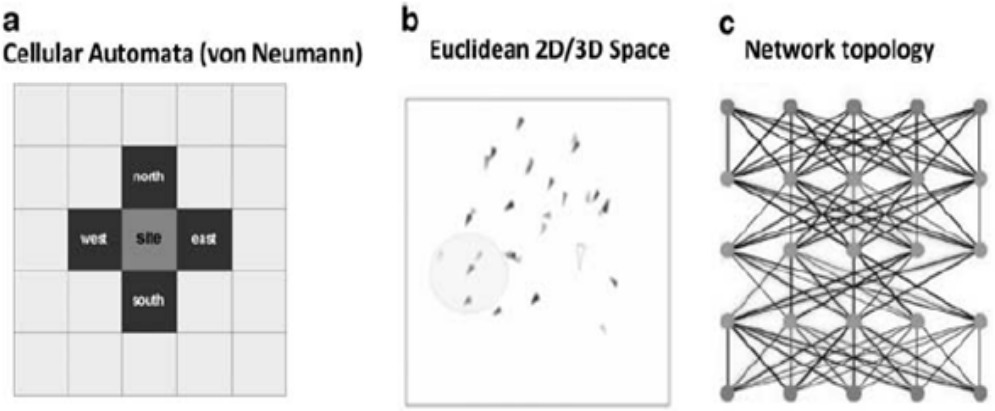
## Structure of Agent-based Models

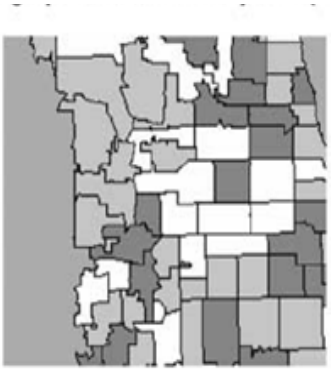
- A set of agents, their attributes and behaviours.
- A set of agent relationships and methods of interaction:
- An underlying topology of connectedness defines how and with whom agents interact.
- The agents' environment: Agents interact with their environment in addition to other agents.

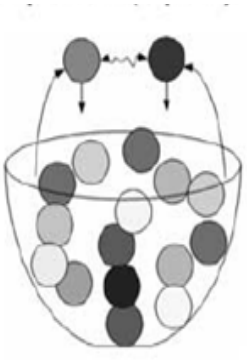
*Figure 2.4: Agent interacting to each other and their environment*




## Interaction Topologies







Activity 2: Laboratory 2: Market Model

**Laboratory Objective**

We'll build an agent-based model of a consumer market – one where each consumer will be an agent – to help us understand how a product enters the market. Since human decisions always include stochastics, agent-based modeling is ideal for modeling market simulations.

Let's assume the following:

The model includes 5000 people who don't use the product, but a combination of advertising and word of mouth will eventually lead them to purchase it.

**Instructions:**

*This laboratory consists of 8 Phases. The first phase is depicted in this activity. Attend it and continue the 7 remaining phases in our practical book (AnyLogic 7 in 3 Days)*

**Grading:**

- The maximum mark of this lab is 70

**Time Required**: 5 Days

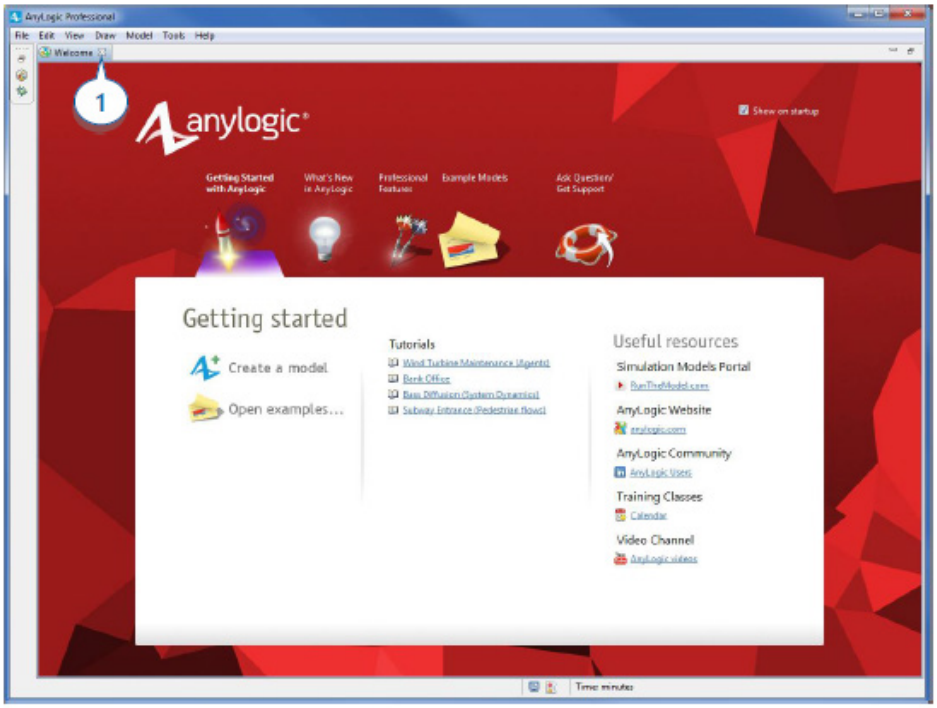
**Phase 1. Creating the agent population**

We'll start by creating a simple model that depicts how advertising leads consumers to purchase our product.

Our model's consumers won't use the product at first, but they are all potentially interested in using it. We'll also represent advertising's influence on consumer demand by allowing a specific percentage of them to become interested in purchasing the product during a given day. For our purposes, Advertising effectiveness = 0.1 determines the percentage of potential users that become ready to buy the product during a given day.

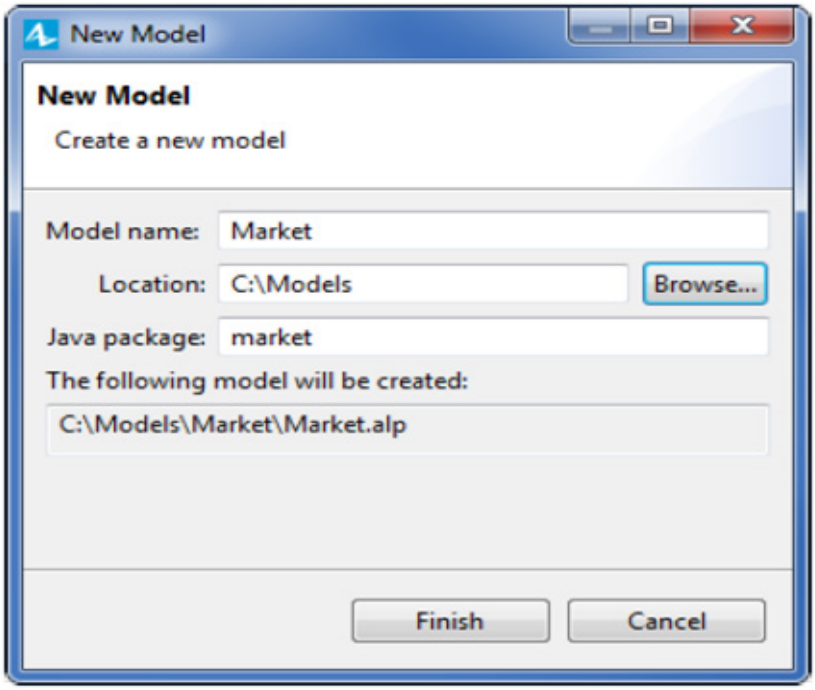
Start AnyLogic and the Welcome page displays.

The Welcome page introduces you to AnyLogic, offers a helpful overview of the program and its features, and allows you to open the example models.




**Welcome page**

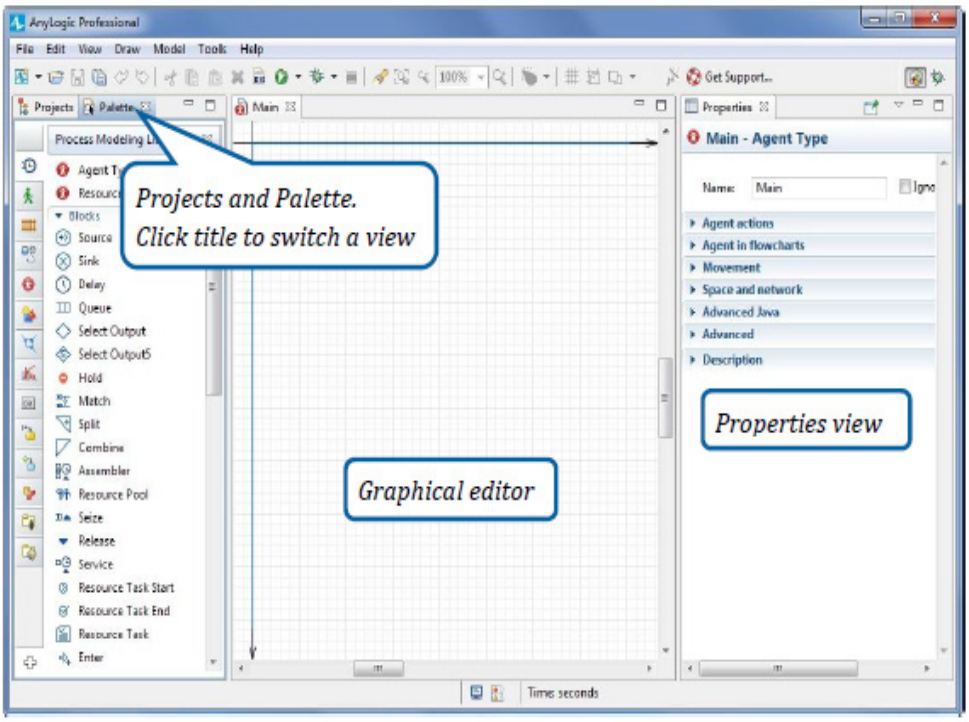
1. Close the Welcome page, and create a new model by selecting **File > New > Model** from AnyLogic's main menu. The **New Model** wizard will open.



2. In the **Model name** box, enter the new model's name: Market.

3. In the **Location** box, select the folder where you want to create the model. You can browse for a folder by clicking **Browse** or type the name of the folder you want to create in the Location box.

4. Click **Finish.**
   Now, let's briefly review AnyLogic's interface.




*AnyLogic workspace*

The graphical editor allows you to edit the agent type's diagram, and you can add model elements by dragging them from the Palette on to the diagram and placing them on the editor's canvas. The elements you place inside the blue frame will appear inside the model window when you run it.

The **Projects** view allows you to access the AnyLogic models you have open in the workspace, and the workspace tree helps you easily navigate them.

The **Palette** view lists the objects grouped in palettes. To add an element to your model, drag the element from the palette on to the graphical editor.

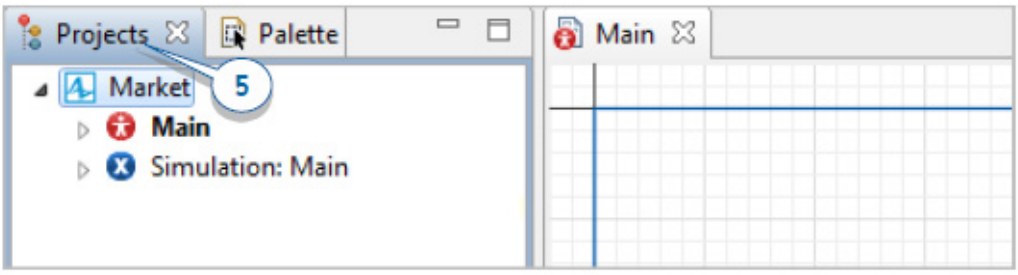
The **Properties** view allows you to view and modify the selected item's properties.

To open/close a view, choose the corresponding item from the View menu. If the item is selected, the corresponding view will be visible.

To resize a view, use your mouse to drag the view's edge.

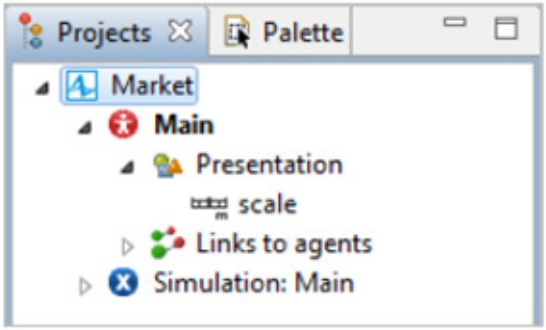
You can always use the option **Reset perspective** in the **Tools** menu to return the views to their default positions.

5. Let's open the Projects view to examine the model's structure. You'll find the Palette and Projects views in the workspace's left section, and you can switch from the Palette view to the Projects view by clicking the Projects tab.




The **Projects** view allows you to access the AnyLogic projects you have open in the workspace, and you can use the workspace tree to quickly and easily navigate them.

AnyLogic uses a tree structure to display your model. The top level displays the model, the level below displays agent types and experiments, and the lower-level branches organize the elements that make up the agent structure.




By default, a model has one agent type - Main - and one experiment Simulation. Double-clicking the agent type or the experiment opens its diagram in the graphical editor.

Clicking the model element in the tree selects the element and centers it in the graphical editor. This may be helpful when you can't find an element on the graphical diagram.

In the graphical editor, you'll see the empty diagram of the model's Main agent type.

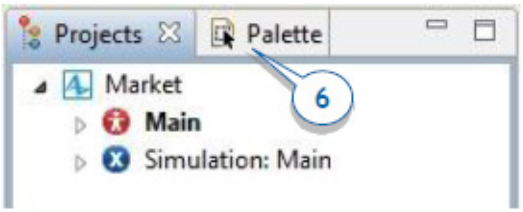
### Agents

Agents are a model's building blocks, and you can use them to model all kinds of real-world objects, including organizations, companies, trucks, processing stations, resources, cities, retailers, physical objects, controllers, and so on.

Each agent typically represents one of the model's logical sections. This allows you to decompose a model into many levels of detail.

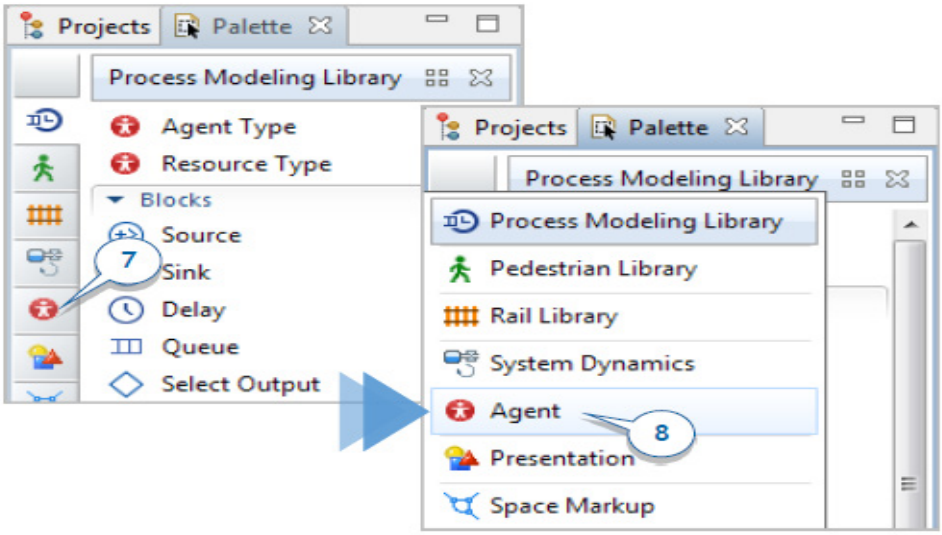
Our model has one agent type, Main. To add consumers, we'll need to create an agent type to represent consumers, and then create an agent population made up of instances of this consumer agent type. In AnyLogic 7, you can use the helpful **New agent** wizard to create agents.

6. We want to add a new model element, but we first need to switch to the **Palette** view by clicking the **Palette** tab.



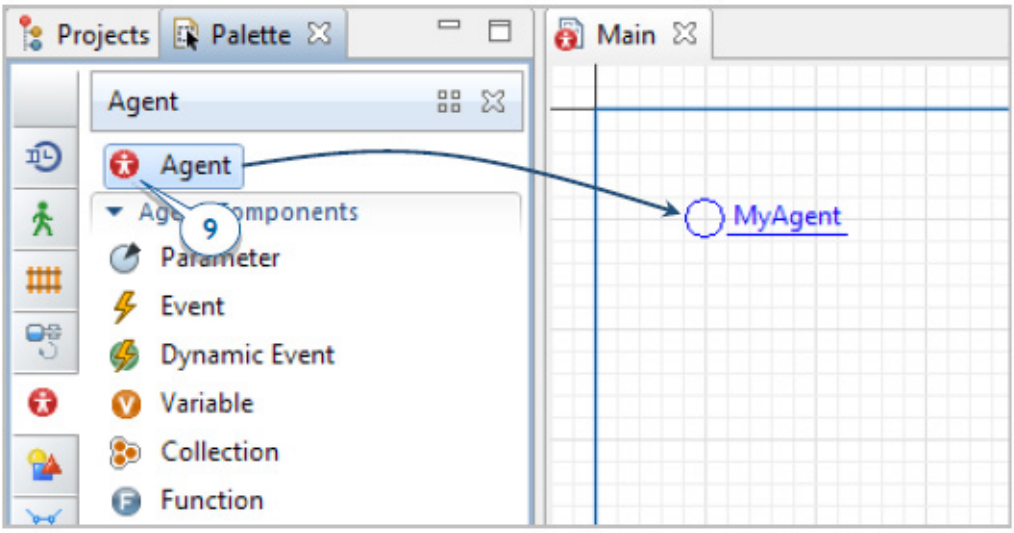

7. Open the Agent palette. To open a specific palette, go to the Palette view and hover your mouse over the view's vertical navigation panel.

8. It will expand to show the names of all palettes so you can select the one you need. Click the Agent palette in the list to select it.

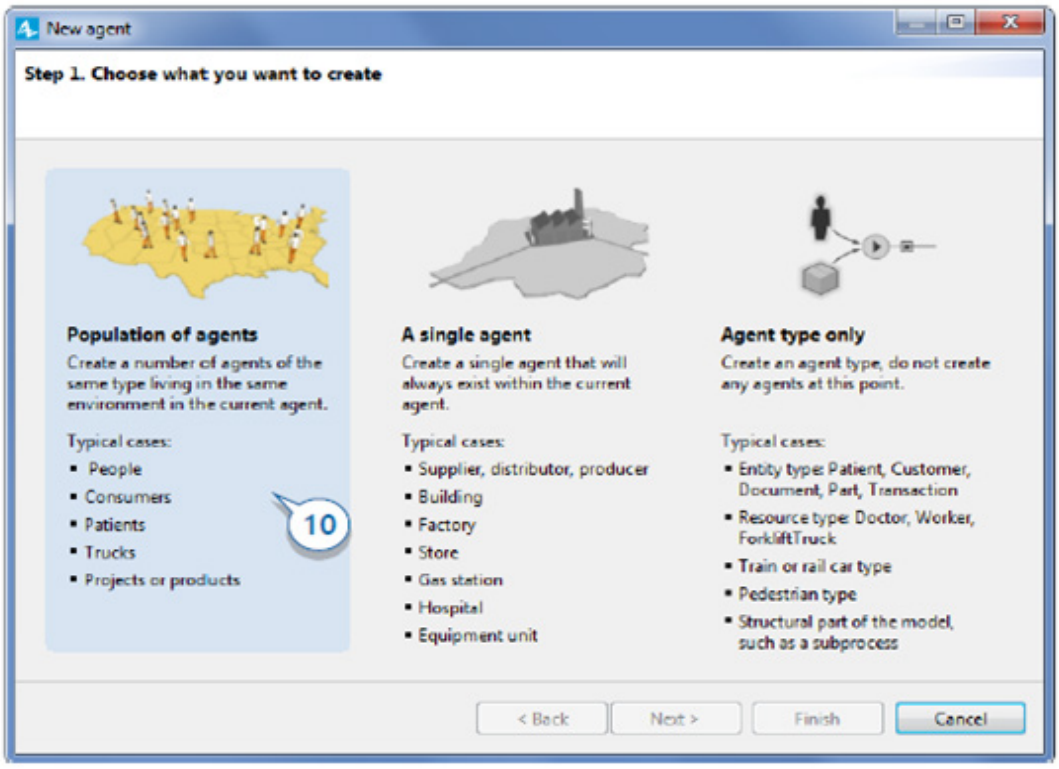



Once you're familiar with the icons, you can click the palette icon you want in the navigation bar.

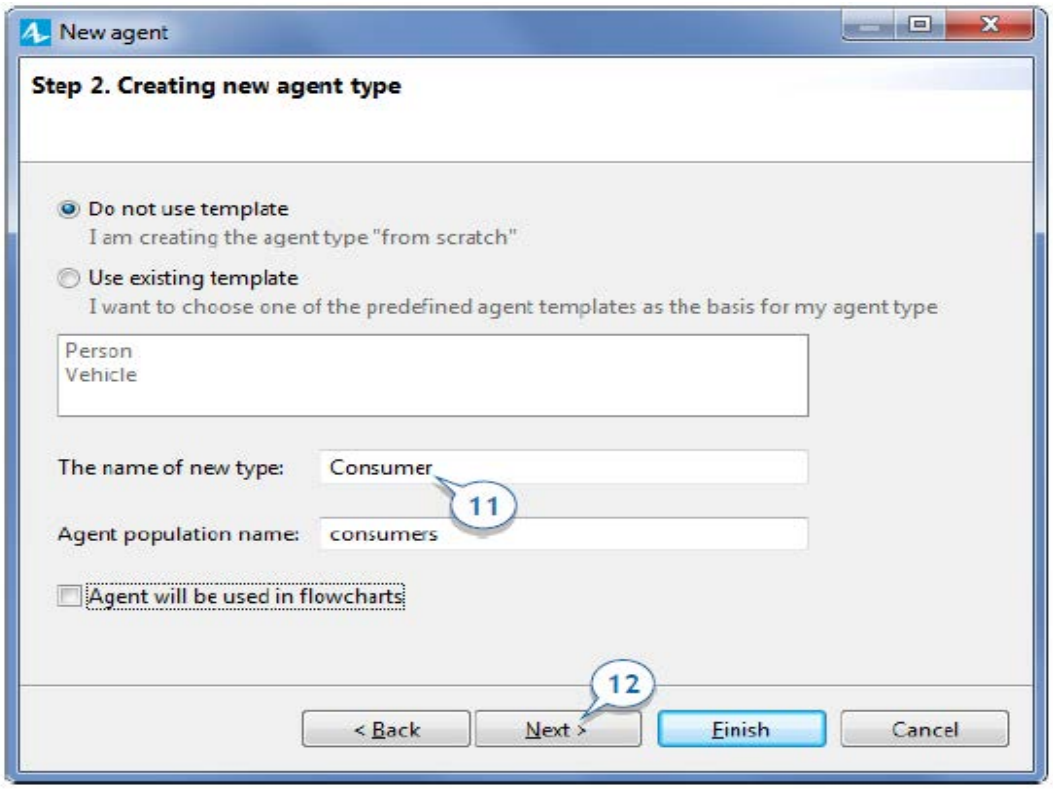
9. Drag the Agent from the Agent palette on to the Main diagram, and the New agent wizard will open.



10. On the Step 1. Choose what you want to create page, select the option that best meets your needs. Since we want to create multiple agents of the same type, select Population of agents and click Next.

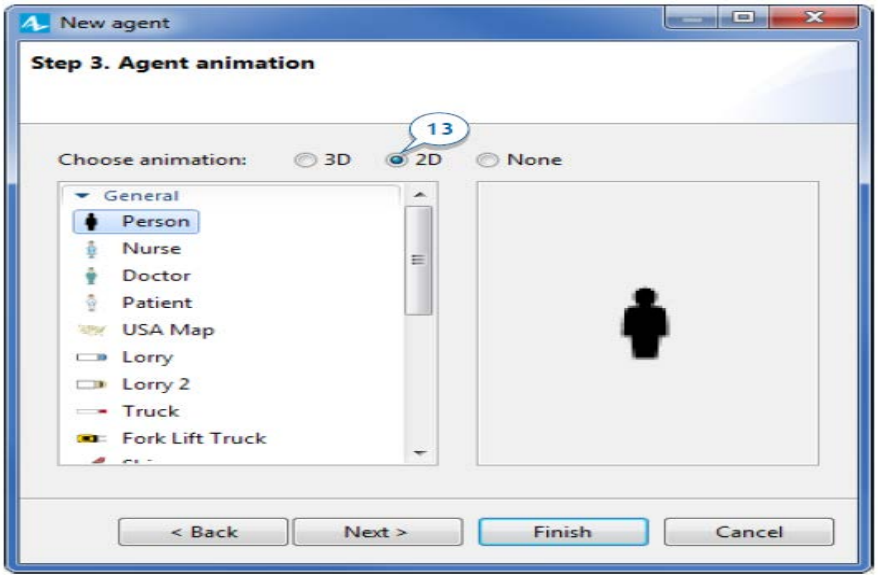



11. On the Step 2. Creating new agent type page, in The name of new type box, type Consumer. The information in the Agent population name box will automatically change to consumers

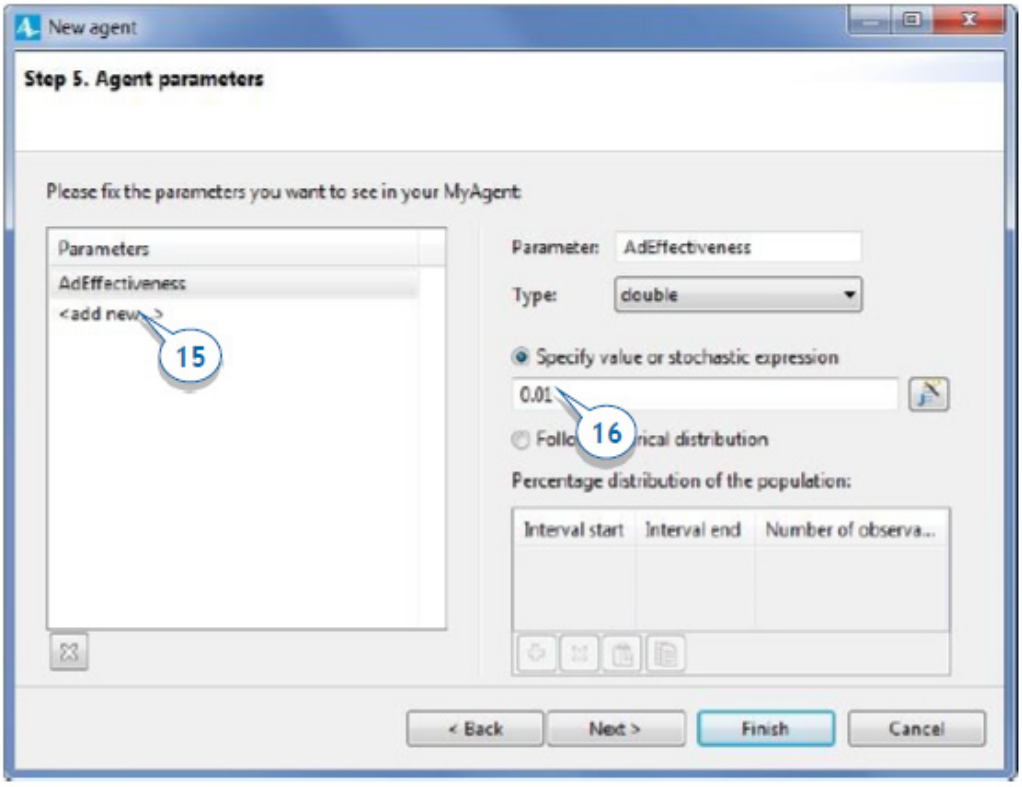


12. Click **Next.**

13. On the **Agent animation** page, choose the agent's animation shape. Since we're creating a simple model that uses **2D** animation, choose 2D, select the **General** list's first item: Person, and click Next.




14. On the **Agent Parameters** page, define the agent's parameters or characteristics. Since our model only considers advertising-related product purchases, we'll add a parameter – AdEffectiveness – to define the percentage of potential users who become ready to buy the product during a given day.



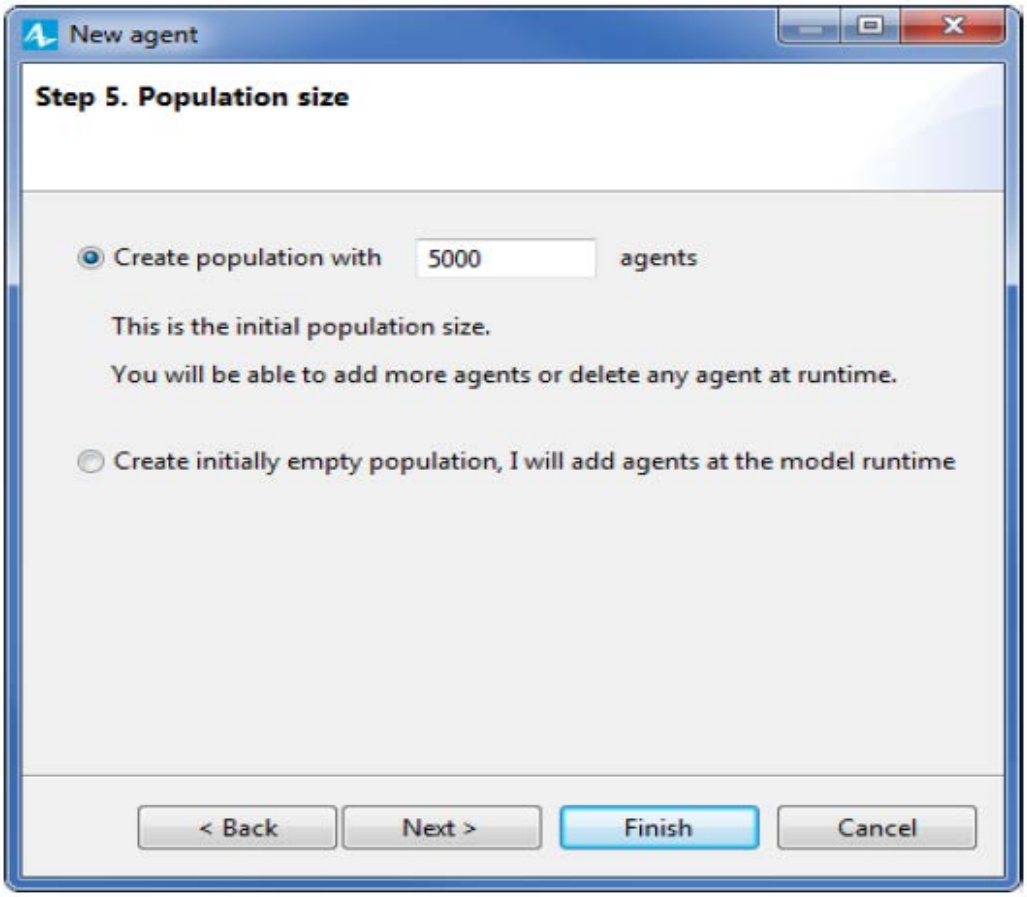
15. On the left section, in the **Parameters** table, click **<add new...>** to create a parameter.

16. In the **Parameter** box, change the default parameter's name to AdEffectiveness, and choose **double** as the parameter **Type**. We'll assume an average of 1% of our model's potential users will want to buy the product during a given day, so specify 0.01 as the parameter's value.

17. Click **Next.**

18. On the **Population size** page, type 5000 in the **Create population with** … agents box to create 5000 instances of the Consumer type. Each instance in the population will model a specific agent-consumer.

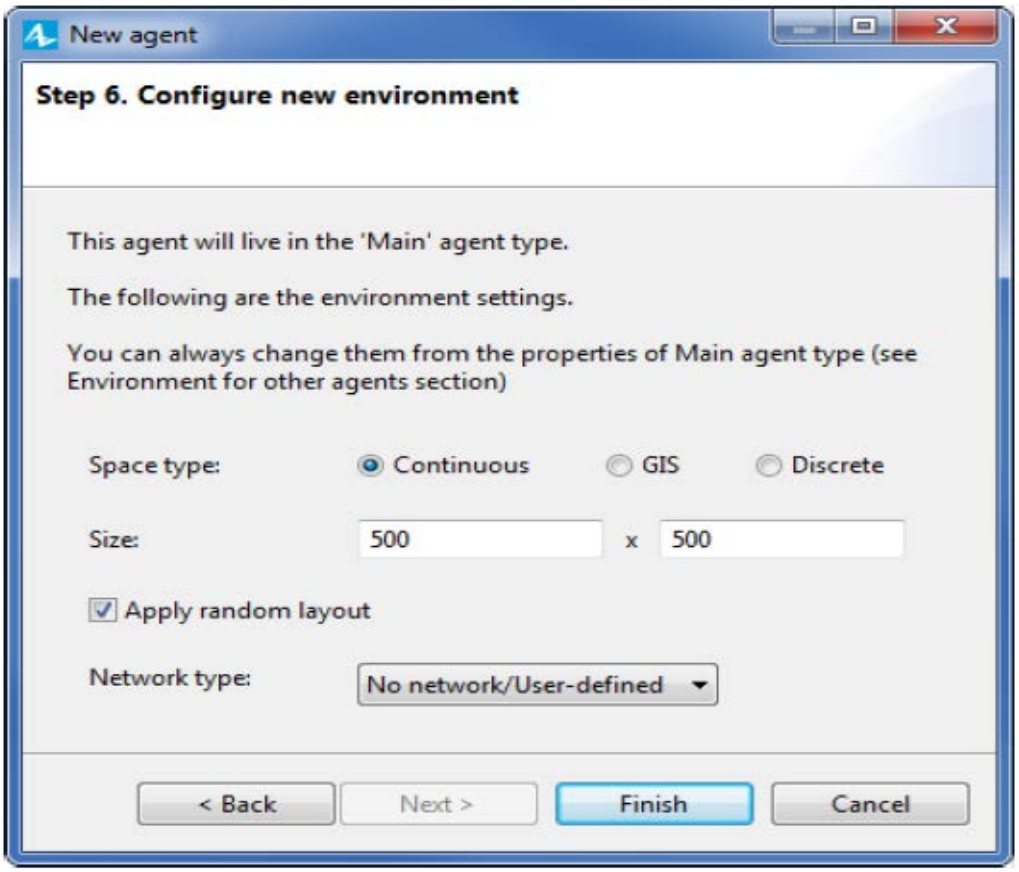
    While we've created our agent population, we won't see 5,000 Person animation figures on Main diagram. Instead, AnyLogic will use the 5000 agents in the population we've called consumers to simulate the market when we run our model.




19. Click **Next.**

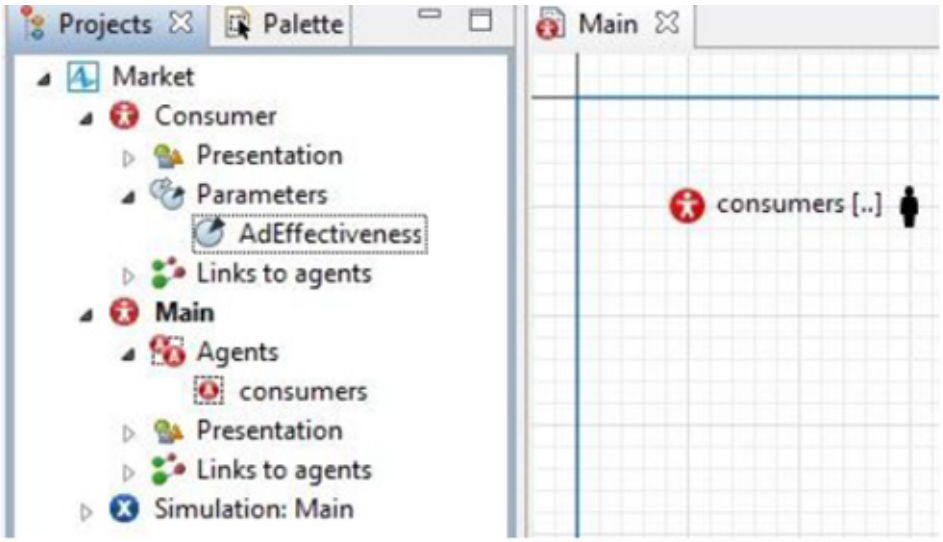
20. 20. On the **Configure new environment** page, accept the default values for the environment's space type (**Continuous**) and both its **Width** and **Height** values (500). AnyLogic will display the agents in a 500x500 pixel rectangle.

21. Select the **Apply random layout** box to randomly distribute the agents across the 500 pixel width and height we've defined. Since we don't want to create an agent network, we'll accept the default **No network/User-defined** network type.




22. Click **Finish.**

23. Let's use the **Projects** view to see the new elements that the wizard created. Expand the model tree branches to see the internals



Our model now has two agent types: Main and Consumer.

- The Consumer agent type has the agent's animation shape (person, in the **Presentation** branch) and the parameter AdEffectiveness.
- The Main agent type contains the agent population consumers (a set of 5000 agents of type Consumer).

### Agent's environment

The Main agent acts as the environment for the consumers population. Since the environment defines the space, layout, network, and communication that our agents use, we'll need an environment to arrange our agent presentations and model the "word of mouth" advertising that occurs when our agents interact.

24. Click Main in the Projects to open its properties in the Properties view (you'll find Properties in the AnyLogic window's right half).

In the Space and network section of Main properties, you can adjust the environment settings for the consumers' agent population.

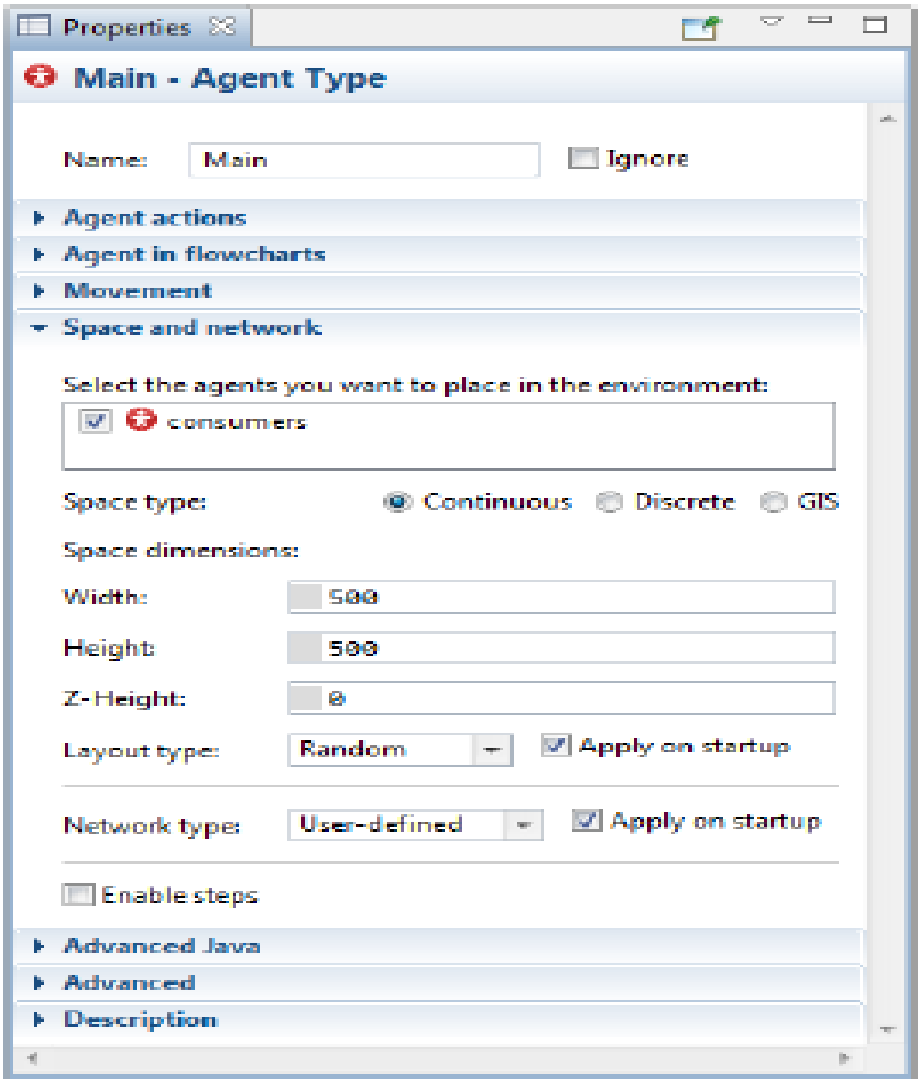
*The Properties view*

The **Properties** view is a context-sensitive view of the element's properties.

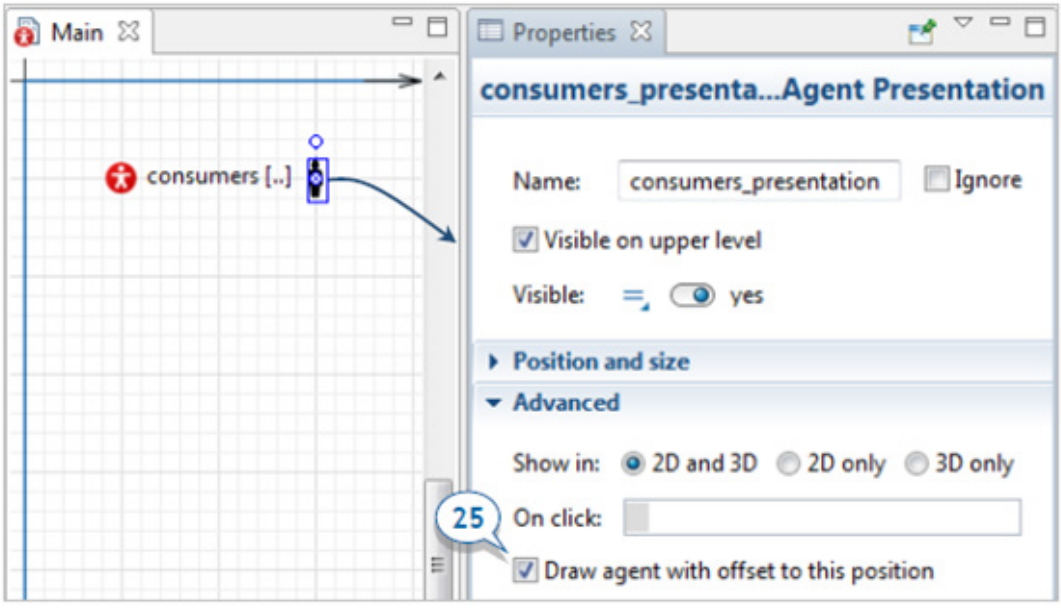
To modify an element's properties, select the element by clicking it in the graphical editor or in the **Projects** view, and then use the **Properties** view to modify the properties.

The **Properties** view has several sections. To expand or collapse a section, click its title.

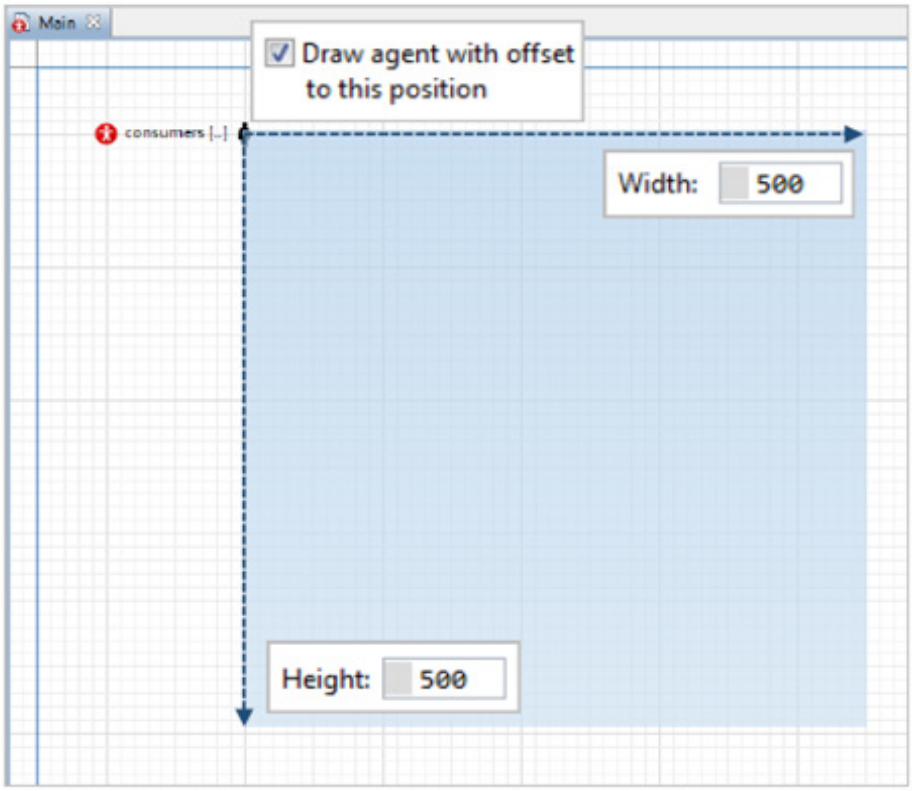
The selected element's name and type display at the top of the view




25. On Main diagram, select the agent population's non-editable embedded animation shape , open the Advanced properties section, and select the Draw agent with offset to this position option.



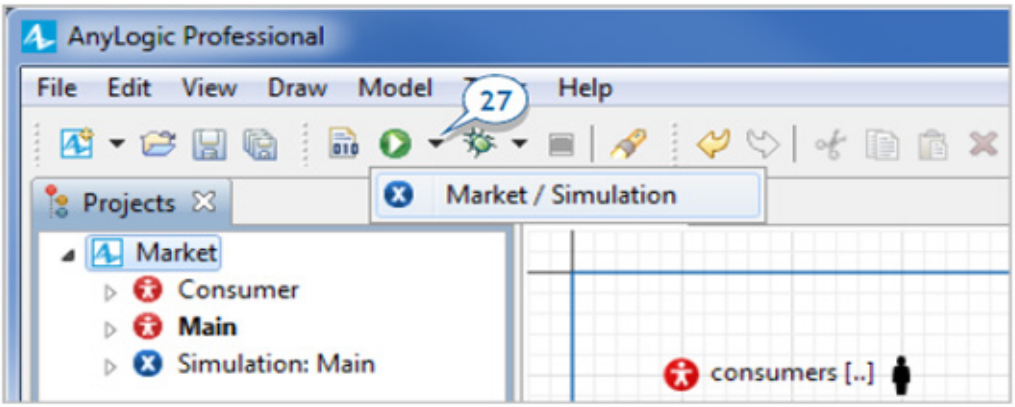
As you can see in the following figure, the animation shape defines the upper-left corner of the 500x500 pixel space where the individual agents will reside when we run the model.




We've finished building this very simple model, and you can now run it and observe its behaviour.

26. 26. On the toolbar, click the **Build** button to build the model and check it for errors.

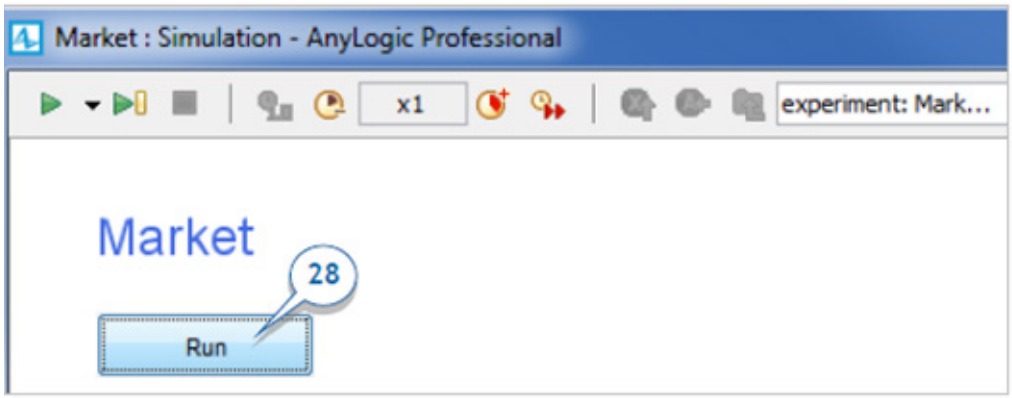
27. 27. Locate the **Run** button, and click the small triangle to the right. Select the experiment you want to run. Choose **Market / Simulation** from the list.



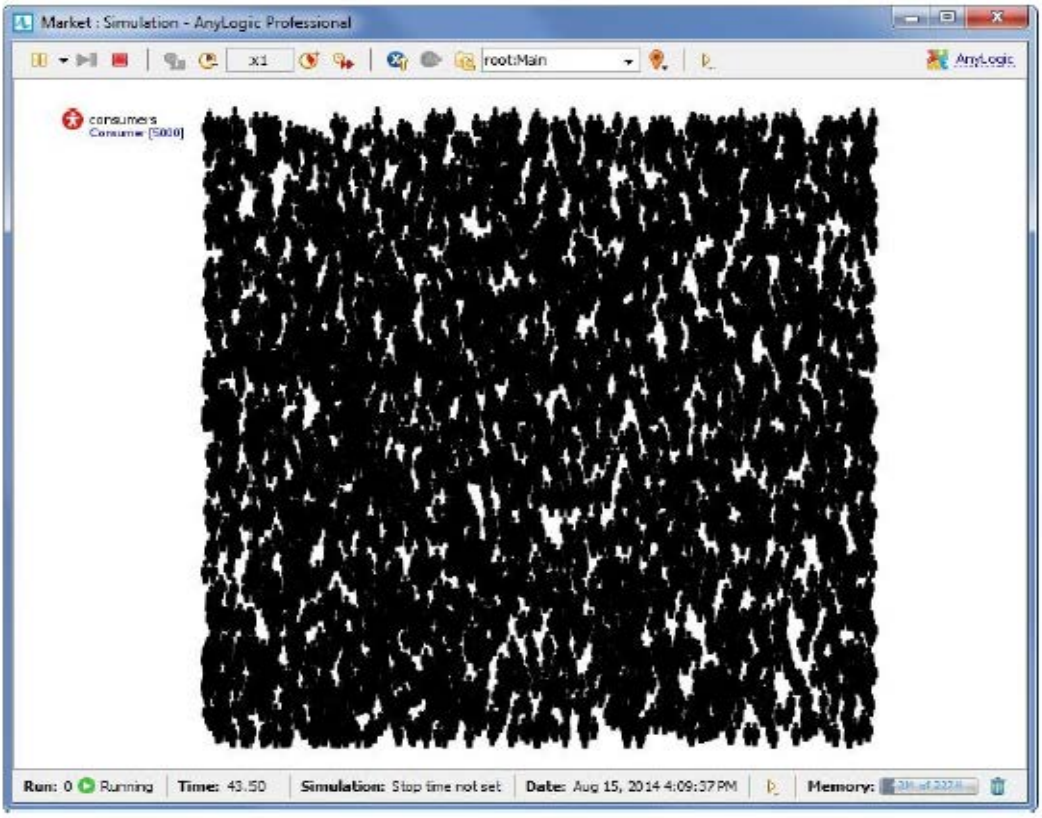

Since you can have several models open at the same time - and each model may have several experiments – you must select the correct experiment.

After you start the model, the presentation window displays the launched experiment Simulation. By default, the presentation window displays the model's name and the Run button.

28. Click the Run button to run the mod




You'll see the model's presentation (the presentation you created for Main agent) that shows 5000 animations for the agents that comprise the consumers' population. Since we didn't create any behavior for our agents, the animation appears still.




*Model window's status bar*

To ensure the model is running, look at the status bar at the bottom of the model window:


- The status bar displays the model's simulation status (Running, Paused, Idle, or Finished), current model time, model date, etc.
- You can customize the status bar by clicking the button, and choosing the required sections from the pop-up list.

You can use the toolbar at the top of AnyLogic's presentation window to control the model's execution.

*Controlling the model execution*

**Run from the current state**

[Visible when the model is not running] Starts the simulation or, if the simulation was paused, resumes it.

**Step**

Executes one model event and then pauses the model execution.

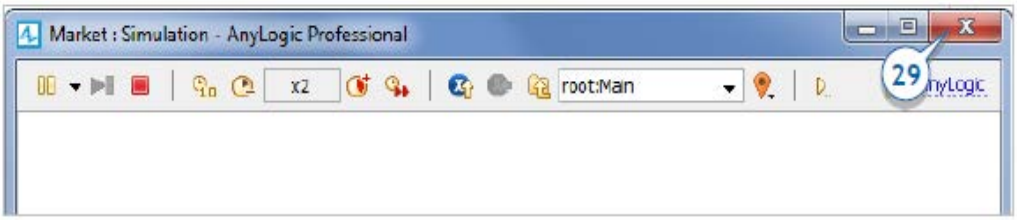
**Pause**

[Visible when the model is running] Pauses the simulation. You can resume a paused simulation at any time.

**Terminate execution**

Terminates the current model execution.

29. We're ready to define the consumer's logic. To continue developing our model, close the presentation window.




## Unit Summary

Agent-based modelling (ABM) enables to build models that simulate the actions and interactions of agents that can be diverse things (vehicles, units of equipment, projects, products, ideas, organizations, investments, pieces of land, people in different roles, etc.). These agents have common characteristics, they are autonomous, self-organized, social, adaptive, and goal-directed. Agent-based models enable to simulate behaviours in highly abstract environments where agents represent competing companies or governments.

## Unit Assessment

Instructions:

Attend all the questions below

## Grading:

Maximum marks for each questions are in brackets

The maximum overall mark is 35.

1. What specific problem should be solved by the model? What specific questions should the model answer? What value-added would agent-based modelling bring to the problem that other modelling approaches cannot bring? [5]
2. What should the agents be in the model? Who are the decision makers in the system? What are the entities that have behaviours? What data on agents are simply descriptive (static attributes)? What agent attributes would be calculated endogenously by the model and updated in the agents (dynamic attributes)? [5]
3. What is the agents' environment? How do the agents interact with the environment? Is an agent's mobility through space an important consideration? [5]
4. What agent behaviours are of interest? What decisions do the agents make? What behaviours are being acted upon? What actions are being taken by the agents? [5]
5. How do the agents interact with each other? With the environment? How expansive or focused are agent interactions? [5]
6. Where might the data come from, especially on agent behaviours, for such a model?
7. How might you validate the model, especially the agent behaviours? [5]

## Unit Readings and Other Resources

### Core Text


- Grigoryev, L. (2015). AnyLogic 7 in Three Days, A Quick Course in Simulation Modelling, ISBN-13: 978-1508933748


### Background Text


- Singh. V, P. (2009). Simulation and Modelling. New Age International (P) Ltd., Publishers
- John D. Sterman. (2000). Business Dynamics: Systems thinking and modeling for a Complex world, , McGraw-Hill

# Unit 3. System Dynamics Modelling

## Introduction

The *system dynamics (SD)* method was created in 1950s by MIT Professor Jay Forrester. Drawing on his science and engineering background, Forrester sought to use the laws of physics, in particular the laws of electrical circuits, to investigate economic and social systems.

Today, system dynamics is typically used in long-term, strategic models, and it assumes high levels of object aggregation: SD models represent people, products, events, and other discrete items by their quantities.

System dynamics is a methodology to study dynamic systems. It suggests you:

Model the system as a causally closed structure that defines its own behaviour.

Discover the system's feedback loops (circular causality) balancing or reinforcing. Feedback loops are the heart of system dynamics.

Identify stocks (accumulations) and flows that affect them.

Stocks are accumulations and characterize the system state. They are the memory of the system and sources of disequilibrium. The model works only with aggregates - the stock's items are indistinguishable. Flows are the rates at which these system states change.

If you're having difficulty distinguishing between stocks and flows, consider how we measure them. Stocks are usually expressed in quantities such as people, inventory levels, money, or knowledge, while flows are typically measurements of quantities in a given time period such as clients per month or dollars per year.

This unit will to teach you how to use AnyLogic to develop system dynamics models. If you want more information about the system dynamics approach, we recommend Business Dynamics: Systems Thinking and Modeling for a Complex World by John Sterman.

## Unit Objectives

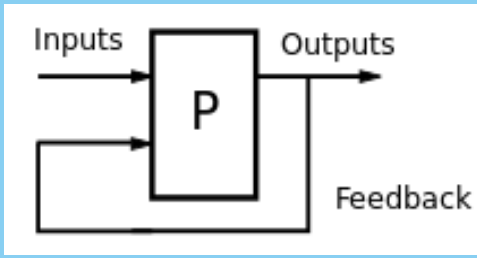
Upon completion of this unit you should be able to:

- Understand what is system dynamics;
- Know the difference between SD and the other two methods used;
- Model and simulate a problem using the SD approach

## Key Terms

**System:** In general, a system is a collection of interacting elements that function together for some purpose. Here, feedback is the differentiating descriptor. Feedback occurs when outputs of a system are routed back as inputs as part of a chain of cause-and-effect that forms a circuit or loop. The system can then be said to feed back into itself.

*Figure 3.1: Depiction of a Feedback*




**Properties of Dynamic Problems**

Contain quantities that vary over time;

Variability can be described causally;

Important causal influences can be contained within a closed system of feedback loops.

What is System Dynamics?

Computer simulation modeling for studying and managing complex feedback systems, such as business and other social systems

**System dynamics:** is a perspective and set of conceptual tools that enable us to understand the structure and dynamics of complex systems.

System dynamics is also a rigorous modeling method that enables us to build formal computer simulations of complex systems and use them to design more effective policies and organizations (John Sterman, 2000)

1. **History of SD**

   Cybernetics (Wiener, 1948): the study of how biological, engineering, social, and economic systems are controlled and regulated;

   Industrial Dynamics (Forrester, 1961): applied principles of cybernetics to industrial systems;

   System Dynamics: Forrester's work has been broadened to include other social and economic systems;

   Relying on computer, System Dynamics provides a framework in which to apply the idea of systems theory to social and economic problems.

## Learning Activities

Activity 1: System Dynamics Causal Graphs

**System Dynamics: How Does It Work?**

1. Identify a problem;
2. Develop a dynamic hypothesis explaining the cause of the problem;
3. Create a basic structure of a causal graph;
4. Augment the causal graph with more information;
5. Convert the augmented causal graph to a system dynamics flow graph;
6. Translate a system dynamics flow graph into DYNAMO programs or equations.

**Critical Steps**

- Thinking in terms of cause-and-effect relationships;
- Focusing on the feedback linkages among components of a system;
- Determining the appropriate boundaries for defining what is to be included within a system.

## Cause & Effects

- Causal thinking is the key to organizing ideas in a system dynamics study
- Instead of 'cause', 'affect' or 'influence' can be used to describe the related components in the system
- Some are logical (e.g. physics)
- Food intake→ weight
- Money → happiness
- Fire → smoke
- Some are not (e.g. sociology, economics)
- Use of seatbelts → reduced highway fatalities
- Shortened daylight hours → increased suicide rates

## Answers

- Thinking in terms of "cause and effect" is not enough;
  ocean → evaporation → cloud → rain → ocean → …

- Feedback: an initial cause ripples through a chain of causation ultimately to re-affect itself;
- Search to identify closed, causal feedback loops is one key element of System Dynamics;
- The most important causal influences will be exactly those that are enclosed within feedback loop.

## Causal Loop Diagram (CLD)

- Represent the feedback structure of systems
- Capture :

  The hypotheses about the causes of dynamics

  The important feedbacks

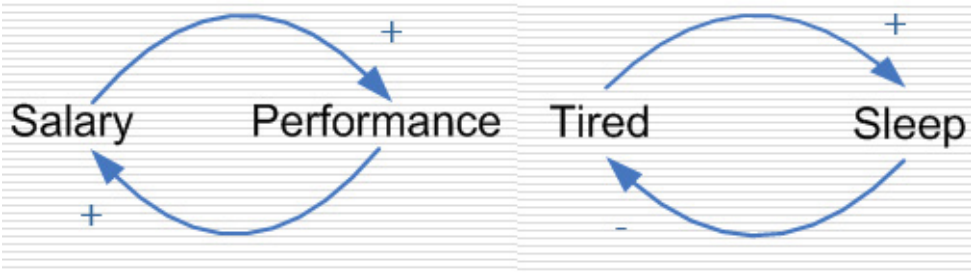
**Example of CLD:**


**Augmenting CLD 1 (Labelling Link Polarity)**

Signing: Add a '+' or a '–' sign at each arrowhead to convey more information

A '+' is used if the cause increase, the effect increases and if the cause decrease, the effect decreases

A '-' is used if the cause increases, the effect decreases and if the cause decreases, the effect increases




Augmenting CLD 2 (Determining Loop Polarity)

1. Positive feedback loops

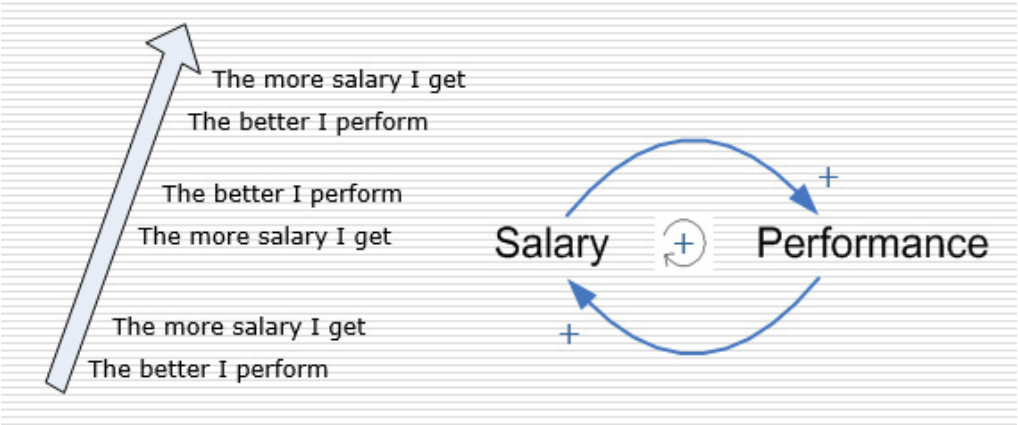
- Have an even number of '–' signs
- Some quantity increase, a "snowball" effect takes over and that quantity continues to increase
- The "snowball" effect can also work in reverse
- Generate behaviors of growth, amplify, deviation, and reinforce
- Notation: place symbol in the center of the loop

2. Negative feedback loops

- Have an odd number of "–" signs
- Tend to produce "stable", "balance", "equilibrium" and "goal-seeking" behavior over time
- Notation: place  symbol in the center of the loop

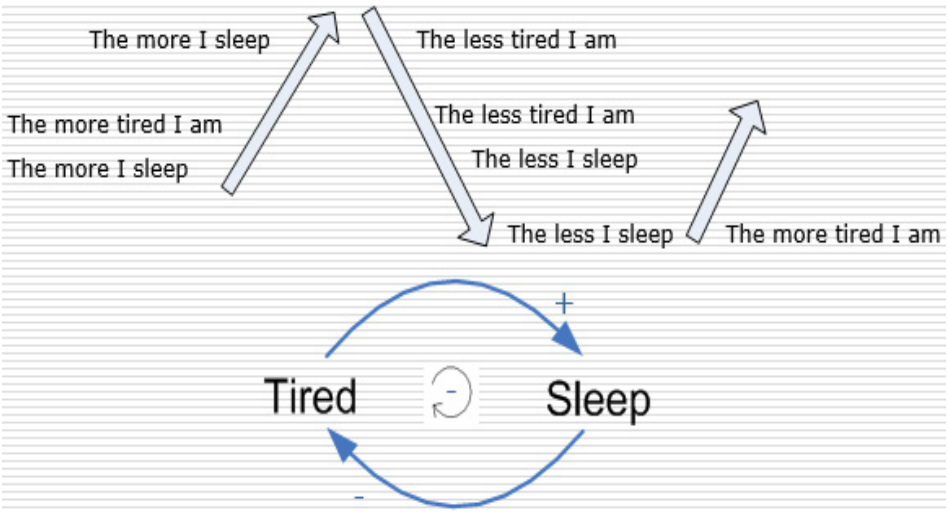
**Example of CLD with Positive Feedback Loop**

Salary → Performance, Performance → Salary




**Example of CLD with Negative Feedback Loop**

Tired → Sleep, Sleep → Tired



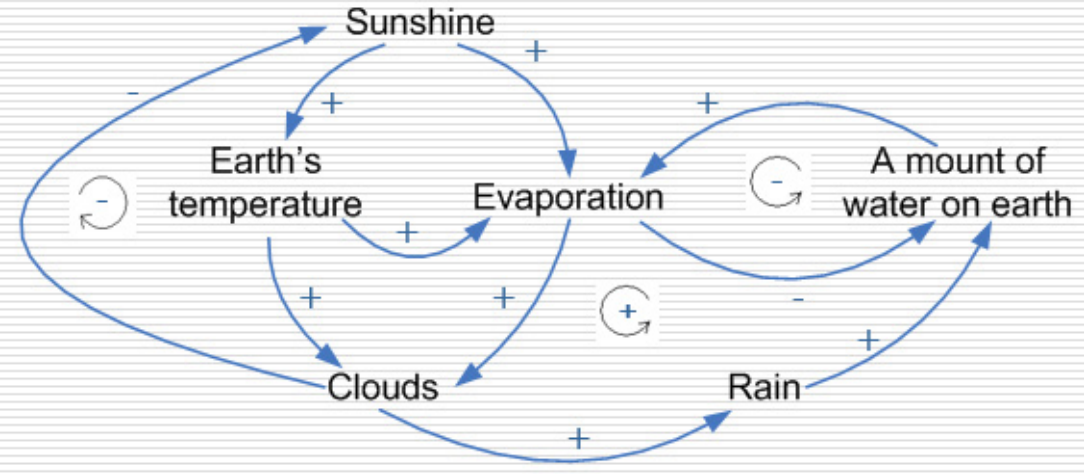
### Loop Dominance

- There are systems which have more than one feedback loop within them;
- A particular loop in a system of more than one loop is most responsible for the overall behavior of that system;
- The dominating loop might shift over time;
- When a feedback loop is within another, one loop must dominate;
- Stable conditions will exist when negative loops dominate positive loops.

### CLD with Combined Feedback Loops (Population Growth


### CLD with Nested Feedback Loops (Self-Regulating Biosphere)

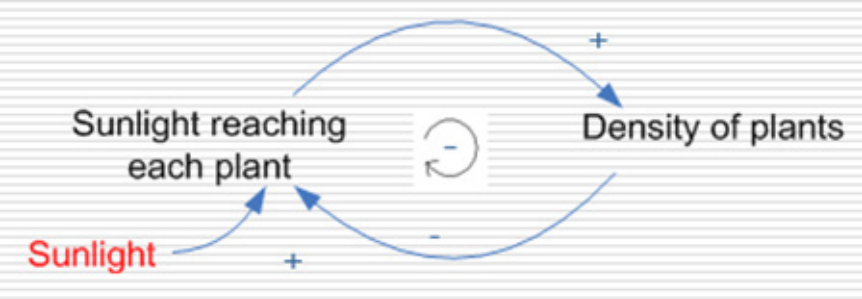
Evaporation → clouds → rain → amount of water → evaporation → …




### Exogenous Items

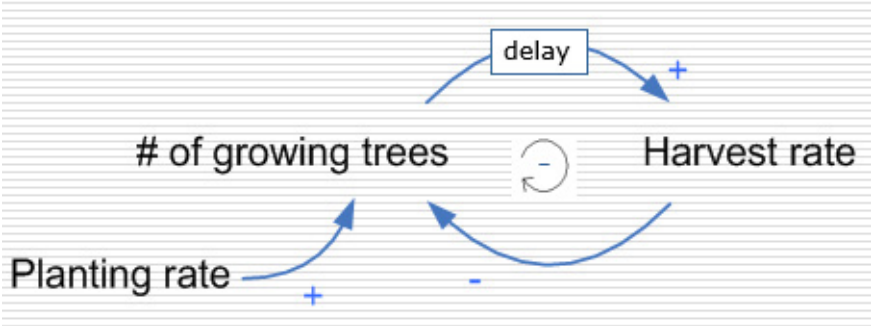
Items that affect other items in the system but are not themselves affected by anything in the system;

Arrows are drawn from these items but there are no arrows drawn to these items.




**Delays**

- Systems often respond sluggishly;
- From the example below, once the trees are planted, the harvest rate can be '0' until the trees grow enough to harvest.




Activity 2: Translating a System Dynamics Flow Graph into DYNAMO Programs or Equations

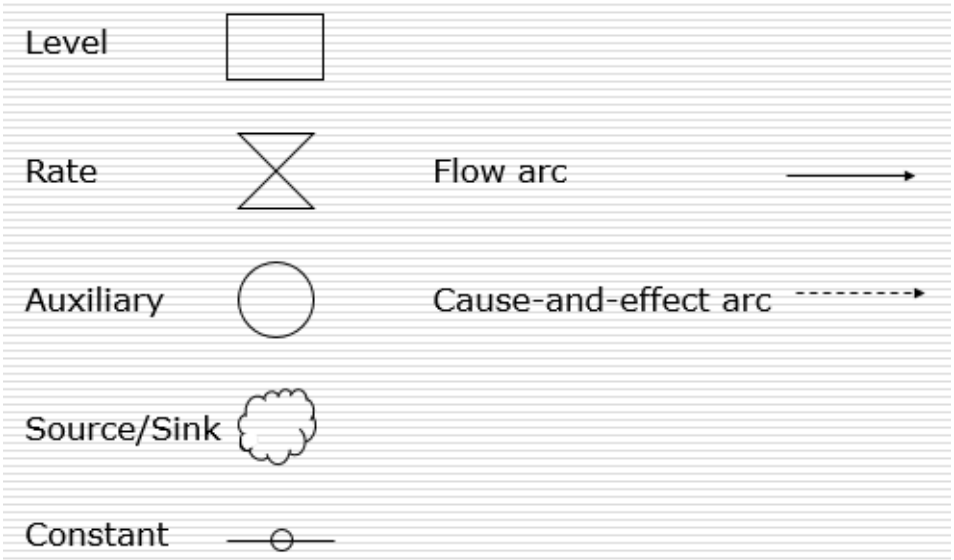
**Casual Loops**

- Provide insight into a system's structure;
- Often difficult to infer the behavior of a system from its casual-loop representation;
- Need to use computer simulation;
- Simulation model: flow diagrams, equations, simulation language;
- DYNAMO (DYNAmic MOdels):

> Not a general-purpose language, but special purpose language to aid in building computer models.

**Flow Graph Symbols**

Figure 3.2: Symbols of a Flow Graph




**Meaning of Symbols**

Level:

- AKA stock, accumulation, or state variable;
- A quantity that accumulates over time;
- Change its value by accumulating or integrating rates;
- Change continuously over time even when the rates are changing discontinuously.

Rate/Flow:

- AKA flow, activity, movement;
- Change the values of levels;
- The value of a rate is :
  - Not dependent on previous values of that rate
  - But dependent on the levels in a system along with exogenous influences

## Auxiliary:

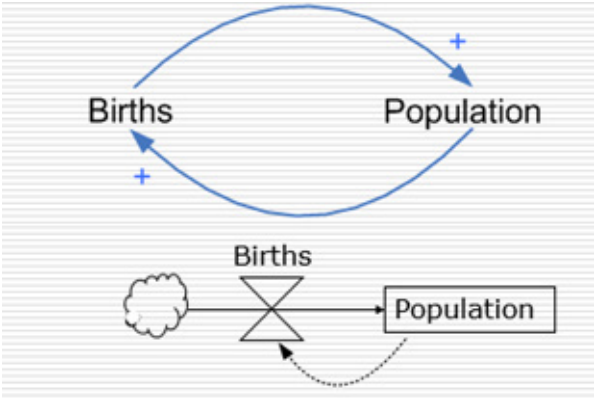
- Arise when the formulation of a level's influence on a rate involves one or more intermediate calculations;
- Often useful in formulating complex rate equations;
- Used for ease of communication and clarity;
- Value changes immediately in response to changes in levels or exogenous influences.

## Source and Sink:

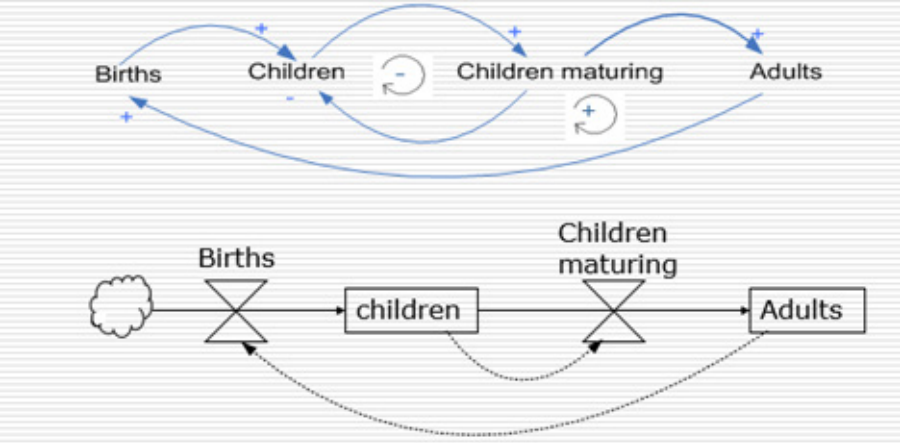
- Source represents systems of levels and rates outside the boundary of the model;
- Sink is where flows terminate outside the system.

**Examples of Flow Graphs**

*Example 1:*




*Example 2:*



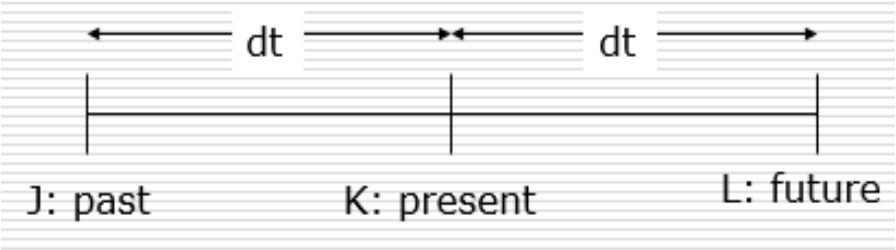
## DYNAMO

- Originally developed by Jack Pugh at MIT;
- First system dynamics simulation language;
- For a long time the language and the field were considered synonymous;
- Provides an equation based development environment for system dynamics models ;
- DYNAMO today runs on PC compatibles under Dos/Windows.

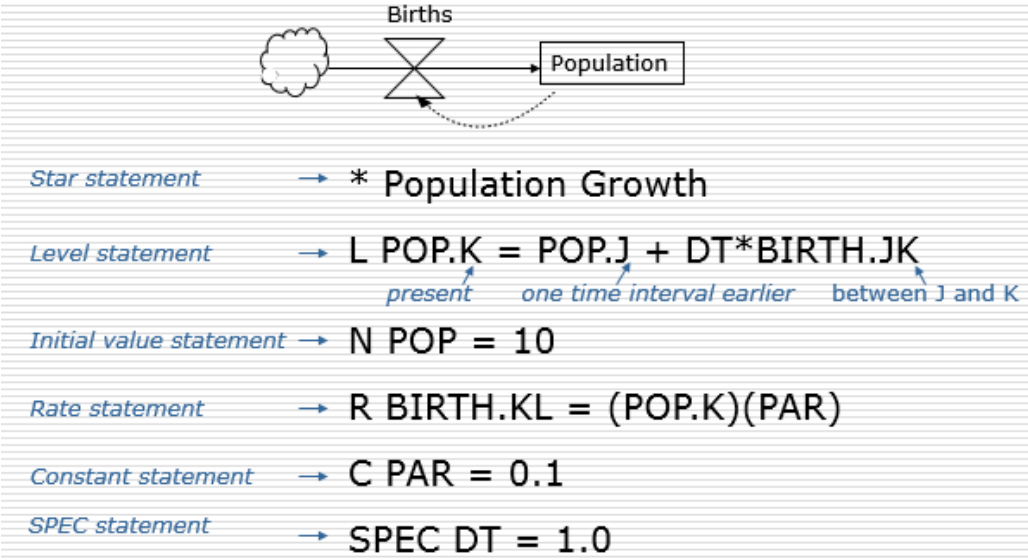
## Time in DYNAMO

- LEVEL.K: a level calculated at the present time;
- LEVEL.J: a level calculated one time interval earlier;
- DT: the length of the time interval between J and K.

*Figure 3.3: Time in DYNAMO*

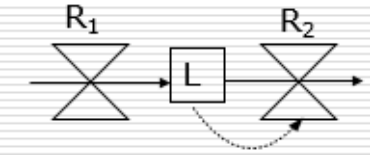



**DYNAMO Program: Population and Birth Model**



### Integral/Differential Equation

Diagram:




Integral Equation:

$$L(t) = \int [R1(s) - R2(s)] \, ds + L(t0)$$

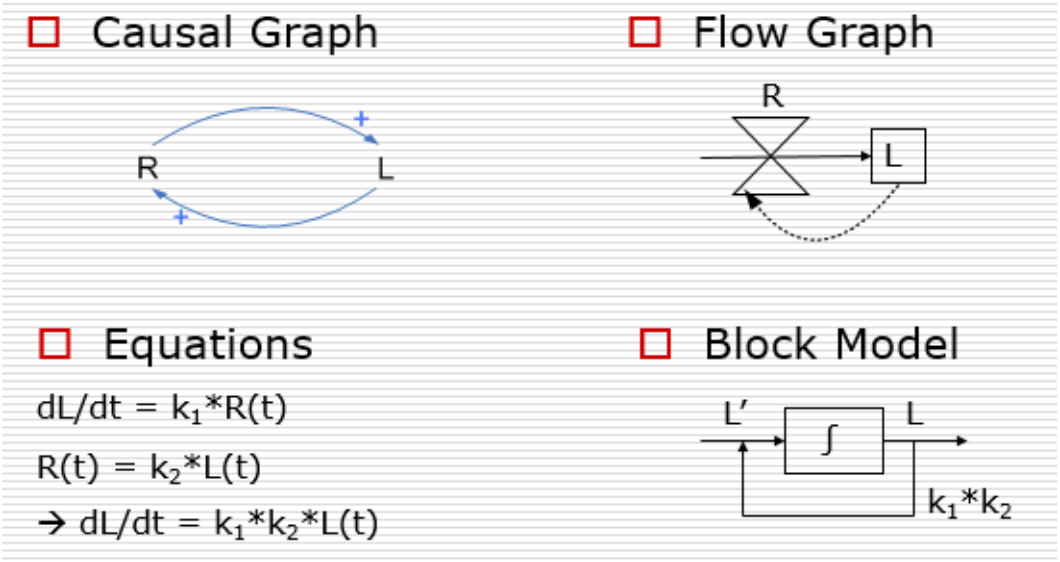
Differential Equation:

$$dL/dt = \text{Net Change in } L = R1(t) - R2(t)$$

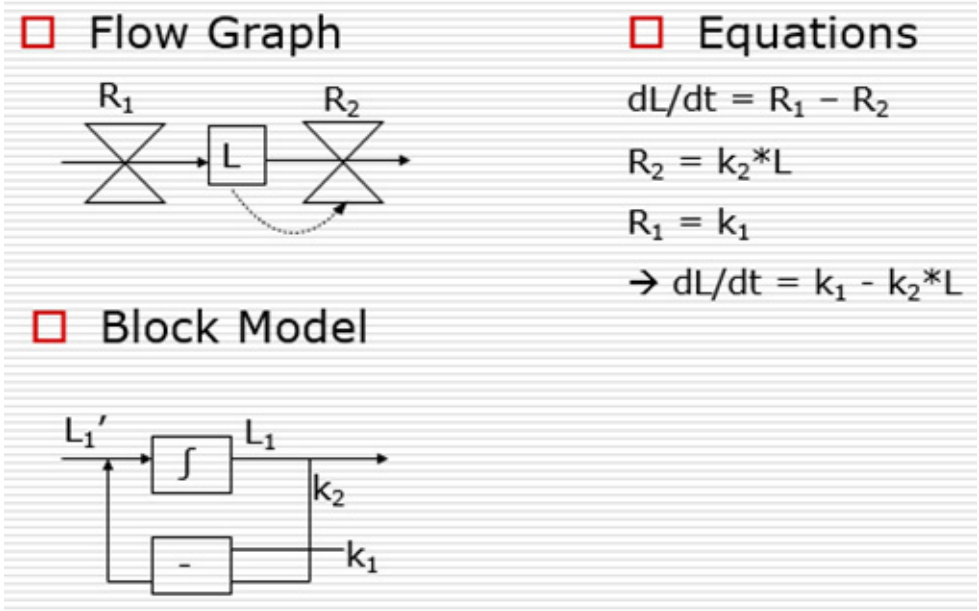
### System Dynamics Algorithm

```
Program Main
    We are given a concept graph with modes and arcs
        The arcs require sign (+,-) labeling
        The nodes require labeling: source, rate, level, constant,
            auxiliary
    For each level node (L) with an input rate node (R1) and
        and output rate node (R2) write:
        dL/dt = k1 * R1 – k2 * R2 ; k1 and k2 are rate constants
    End for
    For all other nodes (N) write:
        N(t) = a linear function of all inset members of this node
    End for
End Main
```

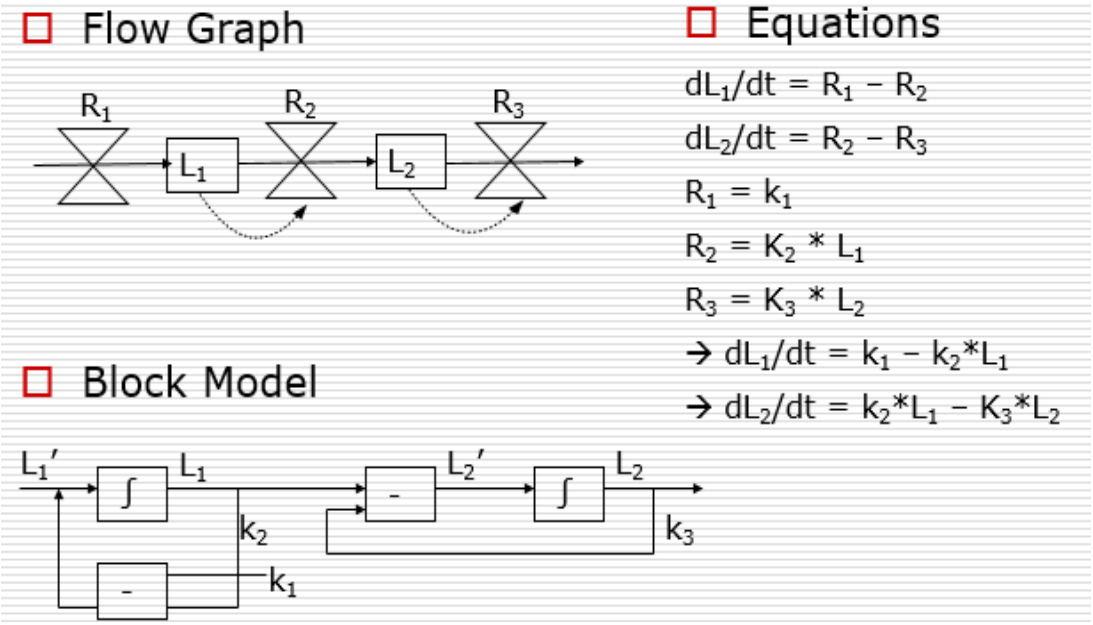
### From Causal Loop Diagram to Simulation Models 1




**From Causal Loop Diagram to Simulation Models 2**




**Causal Loop Diagram to Simulation Models 3**




**Building Construction**

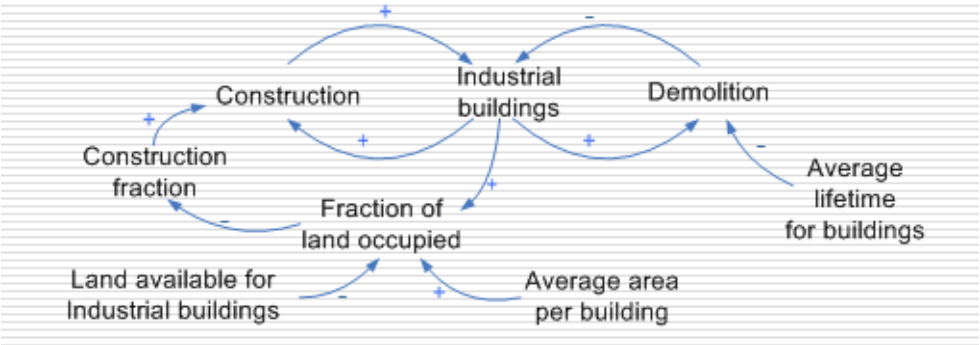
- Problem statement

  Fixed area of available land for construction

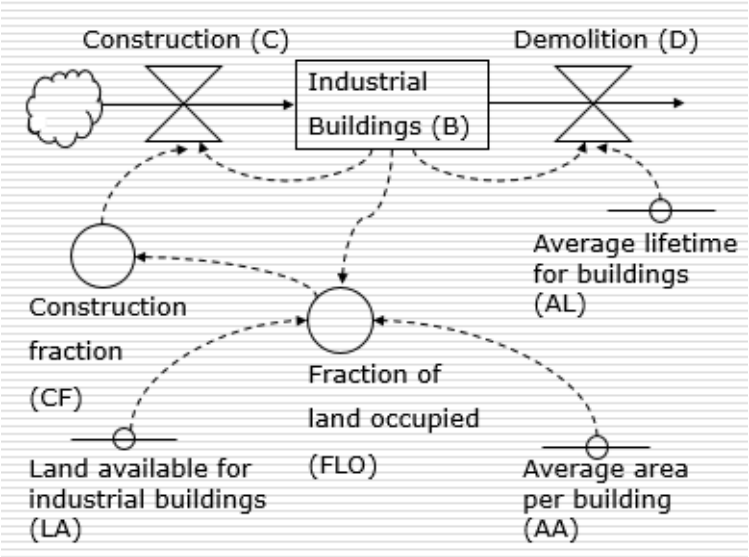
  New buildings are constructed while old buildings are demolished

  Primary state variable will be the total number of buildings over time

- Causal Graph



- Flow Graph




- Equations

$$dB_I/dt = C_r - D_r$$
$$C_r = f1(CF, B_I)$$
$$D_r = f2(AL, B_I)$$
$$CF = f3(FLO)$$
$$FLO = f4(LA, AA, B_I)$$

Activity 3: Laboratory 3: Susceptible, Exposed, Infectious, Recovered (SEIR) Model

**Laboratory Objective:**

We're about to build a model that displays the spread of a contagious disease among a large population. Our sample model will have a population of 10,000 people – a value we call TotalPopulation – of which one person is infectious.

During the infectious phase, a person comes into contact with an average of ContactRateInfectious = 1.25 people each day. If an infectious person comes into contact with a susceptible person, the susceptible person's probability of infection is Infectivity = 0.6.

After a susceptible person is infected, the infection latent phase lasts for

AverageIncubationTime = 10 days.

We'll use the word exposed to describe people who are in the latent phase.

After the latent phase, infectious phase starts. This phase lasts for AverageIllnessDuration = 15 days.

Persons who have recovered from the disease are immune to a second infection.

**Instructions:**

This laboratory consists of 4 Phases. The first phase is depicted in this activity. Attend it and continue the 3 remaining phases in our practical book (AnyLogic 7 in 3 Days)

**Grading:**

- The maximum mark for this lab is 50

**Time Required**: 3 Days

**Phase 1. Creating a Stock and Flow Diagram**

1. Create a new model by selecting File > New > Model from the menu, and then name it SEIR.

   Let's start with drawing stock and flow diagram. To model an epidemic's progress, we need to reduce our population diversity. In this example, we'll consider four important characteristics:

   *Susceptible* - people who are not infected by the virus

   *Exposed* - people who are infected but who can't infect others

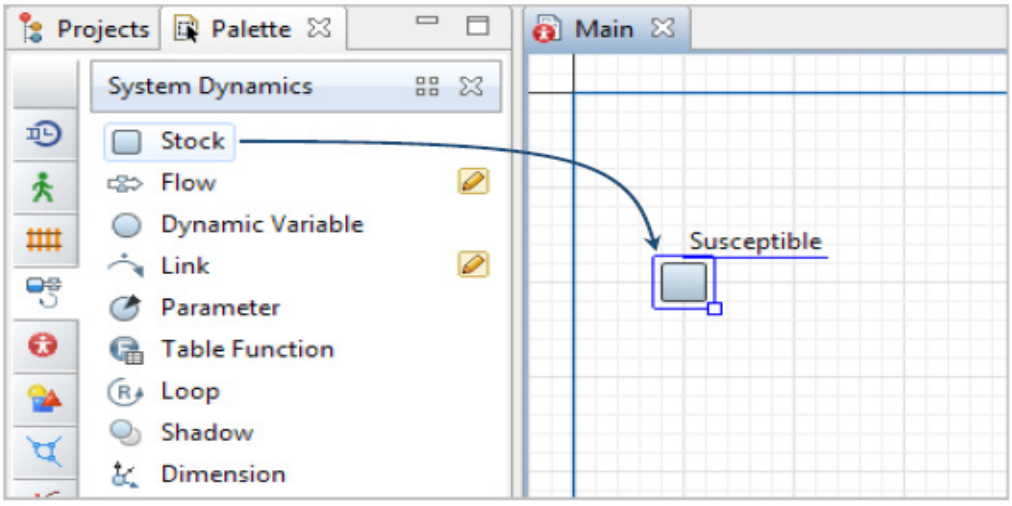
   *Infectious* - people who are infected and who can infect others

   Recovered – people who have recovered from the virus

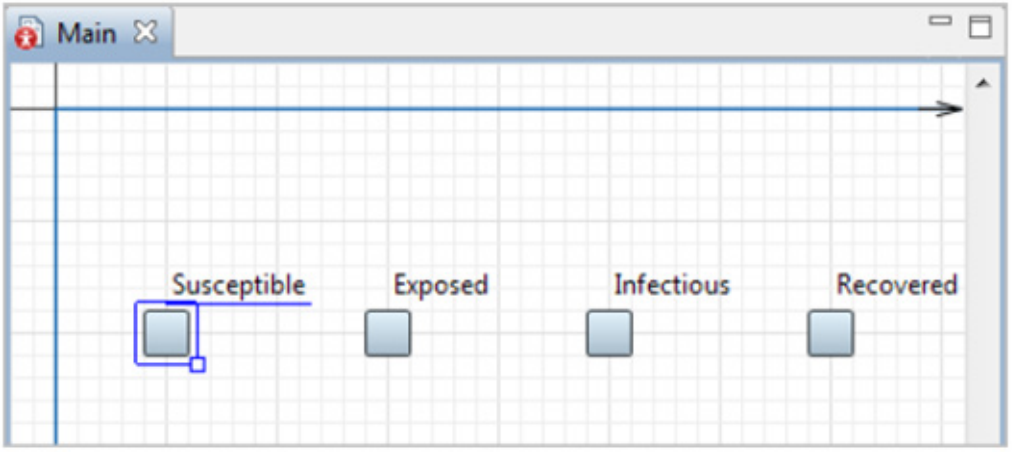
   SEIR is an acronym that represents the four stages: Susceptible-Exposed-Infectious-Recovered. The terminology and the overall structure of the problem is taken from the ("Compartmental models in epidemiology". n.d.) -- namely, from the SEIR (Susceptible Exposed Infectious Recovered) model.

   There are four stocks in our model - one for each stage.

2. Open the **System Dynamics** palette. Drag the Stock from the System **Dynamics** palette on to the diagram. Name it Susceptible.



3. Add three more stocks. Place them as shown in the figure and name them Exposed, Infectious, and Recovered.

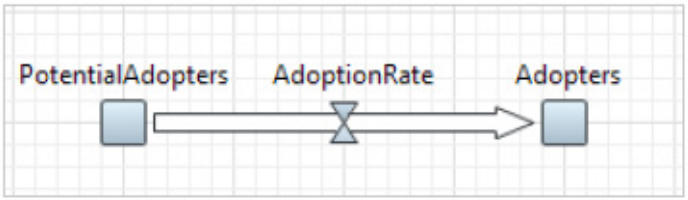



### Stocks and flows

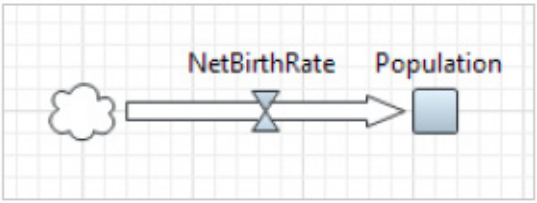
In System Dynamics, ***stocks*** (also known as levels, accumulations, or state variables) represent real-world stocks of material, knowledge, people, money, etc. ***Flows*** define their rate of change - how stock values change and define the system's dynamics. Here are some examples of stocks and flows:

| Stock | Inflows | Outflows |
|---|---|---|
| **Population** | Births<br>Immigration | Deaths<br>Emigration |
| **Fuel tank** | Refueling | Fuel consumption |

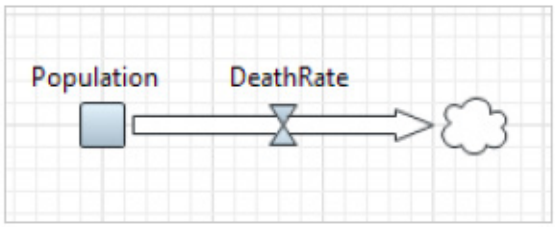
Flow may flow out of one stock and flow in another. Such a flow is outflow for the first stock and inflow for the second one at the same time:




Flow may flow into a stock from nowhere. In this case the cloud (denoting "source") is drawn at the flow's starting point.



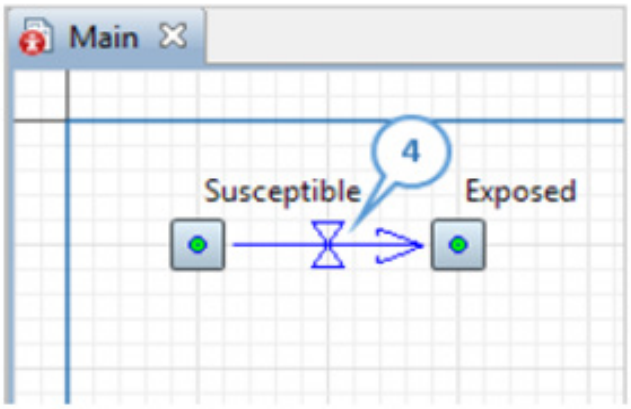

And symmetrically, flow may flow out from a stock to "nowhere". In this case the cloud (denoting "sink") is drawn at the flow's end point.



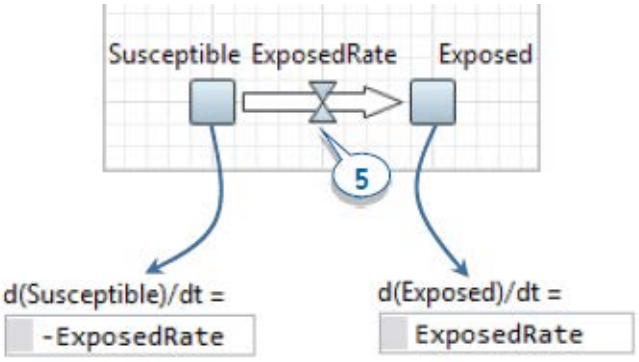

The flow's arrow shows its direction.

In our model, susceptible people are exposed to the virus, become infectious, and then recover. It's a progression that requires our model to use three flows to drive people from one stock to the next.

4. Add the first flow that flows from the stock Susceptible to Exposed. Double-click the stock where the flow flows out (Susceptible), and then click the stock where it flows in (Exposed).




5. Name the flow ExposedRate.




6. You can look at the formulas of Susceptible and Exposed stocks. Since our ExposedRate flow reduces the value of Susceptible stock and increases Exposed, the formulas should be the same as in the figures below. AnyLogic automatically created these formulas when you added the flow.

### Formulas of stocks

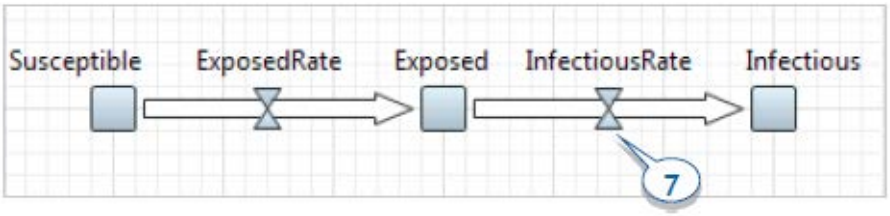
*AnyLogic* automatically generates a stock's formula according to the user's stock-and-flow diagram.

Stock value is calculated according to flows flowing in and out from the stock. The value of inflows – the flows that increase stock value – are added and the value of outflows – flows that reduce the stock – are subtracted from the stock's current value:

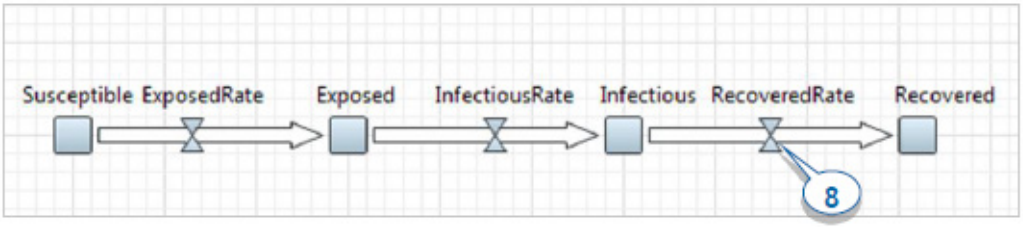
inflow1 + inflow2 … - outflow1 - outflow2 …

In the classic system, dynamics notation only flows can appear in the formula. The formula is non-editable and no other elements other than flows flowing in and out the stock can appear in the formula.

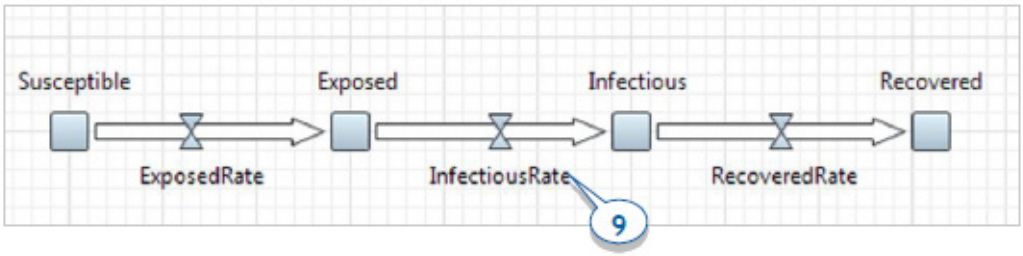
7. Add a flow from Exposed to Infectious, and then name it InfectiousRate.




8. Add a flow from Infectious to Recovered, and then name it RecoveredRate.

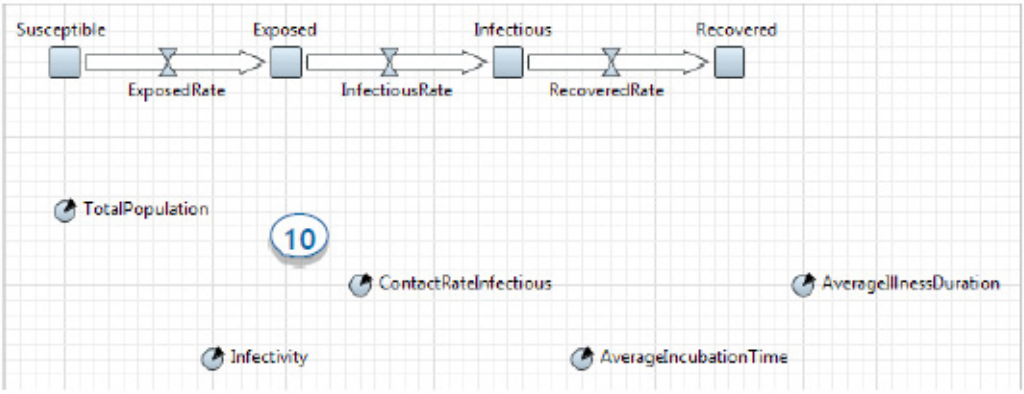



9. Rearrange the flow names as shown in the figure below. To do this, select a flow and then drag its name.



10. Now, let's define the parameters and dependencies. Add five Parameters, name them, and then define their default values according to the information below:

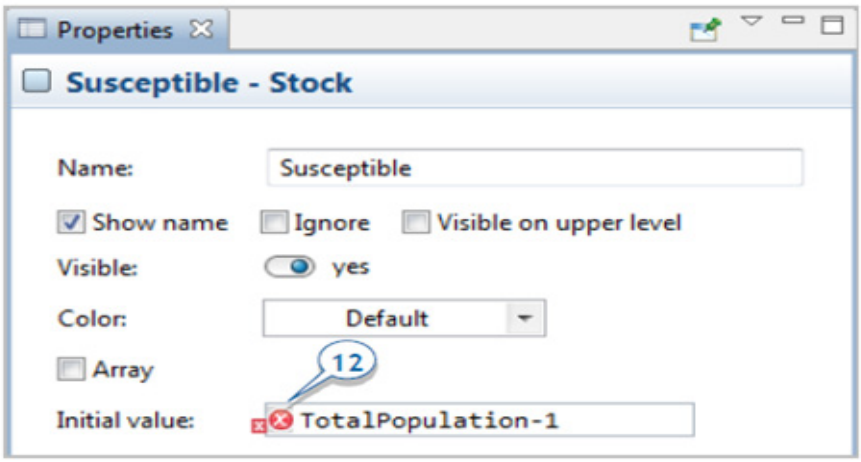
- TotalPopulation = 10 000
- Infectivity = 0.6
- ContactRateInfectious = 1.25
- AverageIncubationTime = 10
- AverageIllnessDuration = 15




11. Define the number of infected people by specifying 1 as the Initial Value of the stock Infectious.

12. Define the Initial Value for the stock Susceptible: TotalPopulation-1.

You may press Ctrl+space (Mac OS: Alt+space) and then select the parameter's name from the Code Completion assistant.)

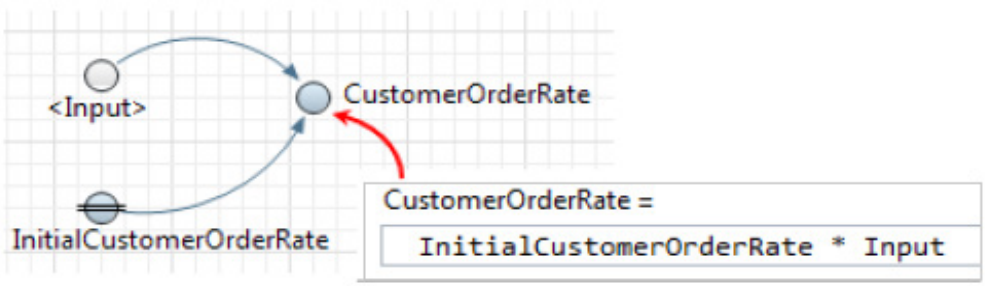



You'll see the red sign to the expression's left. The reason for the problem is you've defined a dependency between two elements in the stock and flow diagram (the stock Susceptible's initial value depends on the parameter TotalPopulation), but this dependency is not defined graphically as it should be.

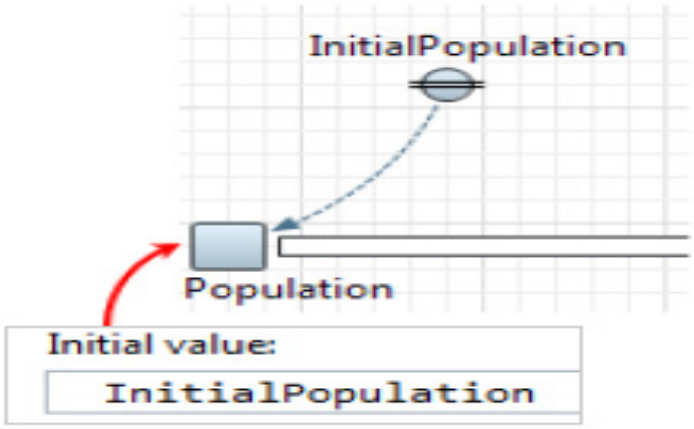
### Dependency links

Stock and flow diagrams have two types of dependencies:

- An element (stock, flow, auxiliary, or parameter) is mentioned in a flow or auxiliary's formula. This link type is drawn with a solid line:




An element is mentioned in the stock's initial value. This link type is drawn with a dotted line:

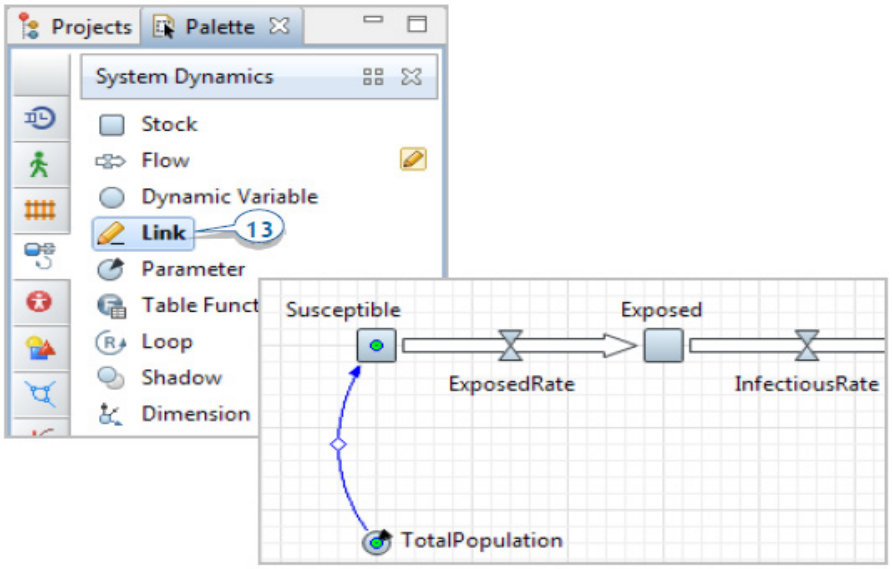



You should use links to graphically define dependencies among a stock and flow diagram's elements.

If an element A is mentioned in the equation or element B's initial value, you should first connect these elements with a link from A to B and then type the expression in B's properties.

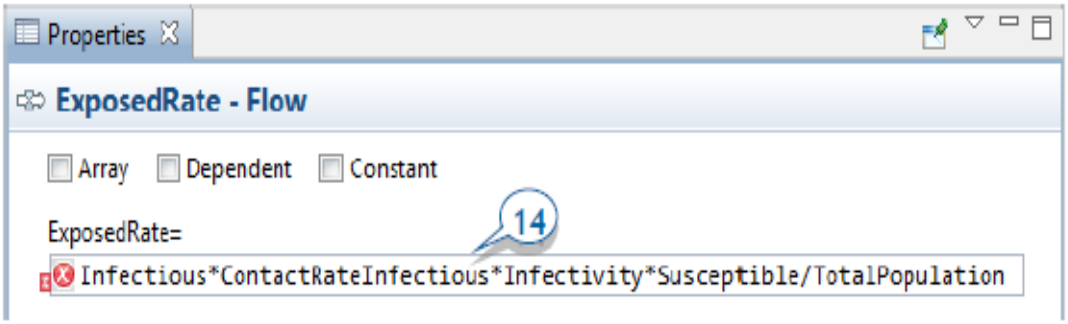
13. Draw a dependency link from TotalPopulation to Susceptible:

    In the System Dynamics palette, double-click the Link element, click TotalPopulation, and then click the stock Susceptible. You should see the link with small circles drawn on its end points:



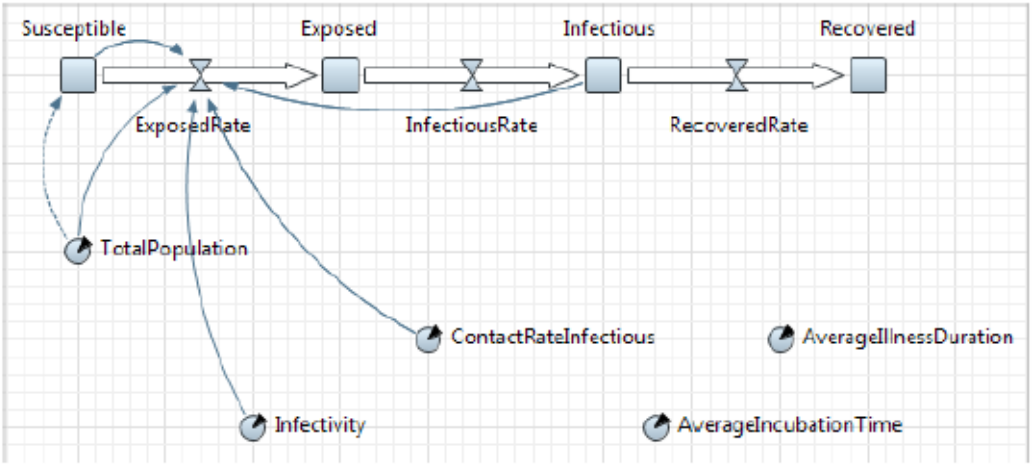

14. Let's define the formula for the flow ExposedRate.

    Click the flow and define the following formula using the Code Completion assistant: Infectious*ContactRateInfectious*Infectivity*Susceptible/TotalPopulation




We need to draw dependency links from the mentioned variables and parameters to this flow. You may find it tedious to manually draw the links, so we'll show you how to add links using AnyLogic's link auto-creation mechanism.

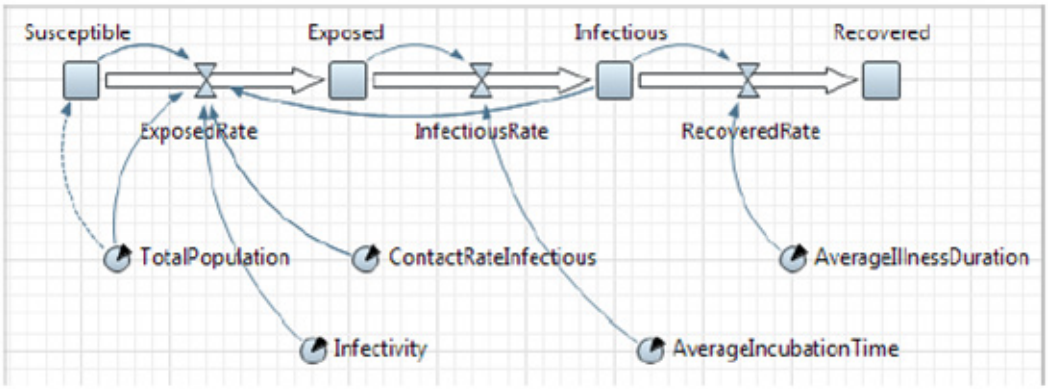
15. 15. Right-click ExposedRate flow in the graphical diagram, and choose **Fix Variable Links > Create Missing Links** from the context menu. Afterward, you should see the links in the stock and flow diagram:



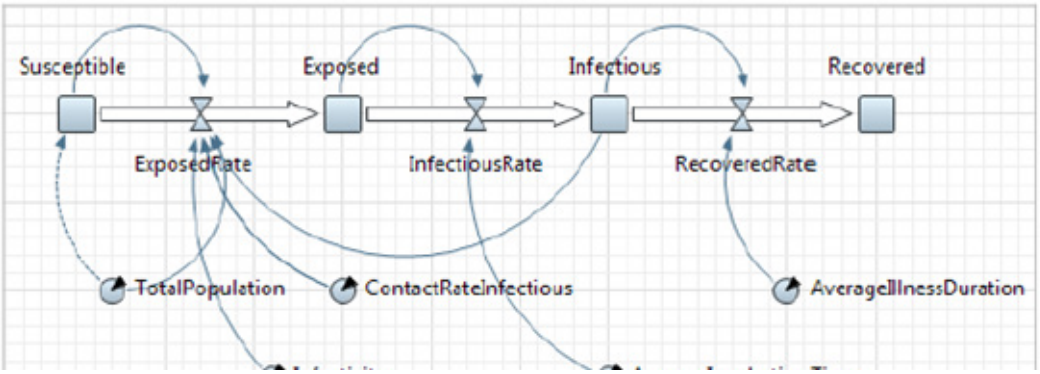

16. Define the following formula for InfectiousRate: Exposed/ AverageIncubationTime

17. Define the following formula for RecoveredRate: Infectious/ AverageIllnessDuration

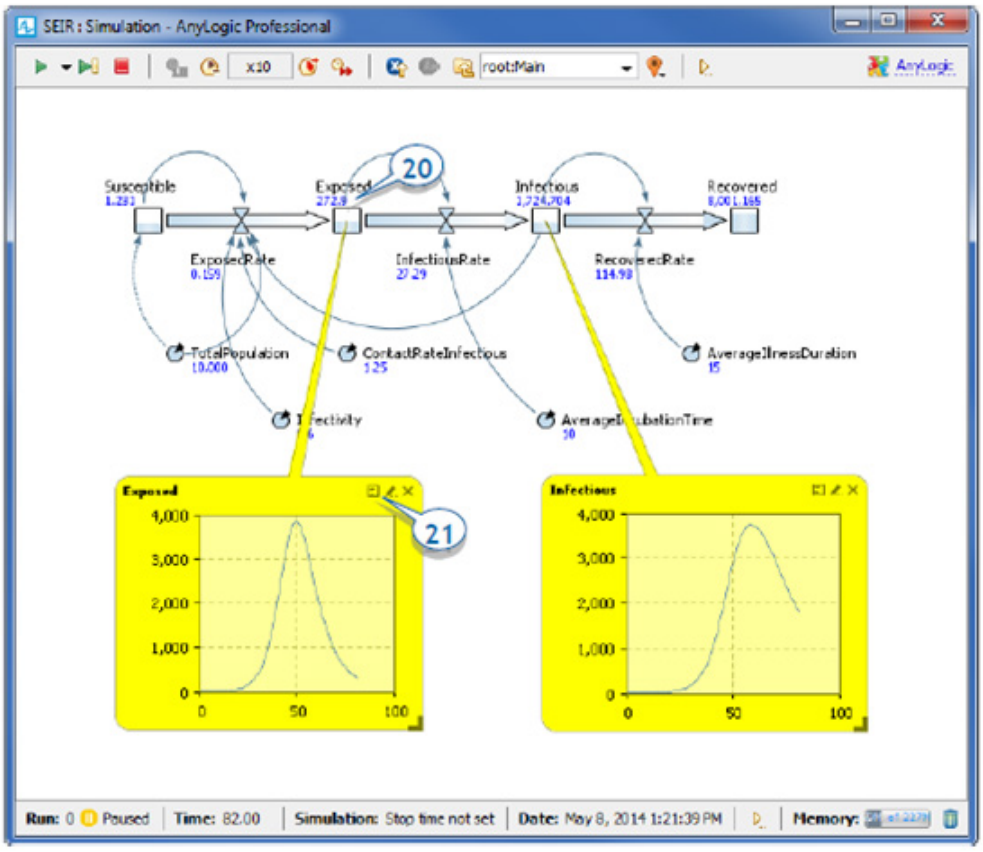
18. Draw the missing dependency links, and your stock and flow diagram should resemble the following image:




19. Adjust the appearance of dependency links. Modify the links' bend angles to make the diagram match the figure below. To adjust the link's bend angle, select it, and then drag the handle in the middle of the link.




20. Run the model and inspect the dynamics using the variables' inspect windows. To open a variable's inspect window, click the variable to select it. To resize the window, drag its lower right corner.




21. To switch the inspect window to the plot mode, click the leftmost icon in its toolbar.

22. Increase the model execution speed to make the simulation go faster.

## Unit Summary

In this Unit we have examined a perspective and set of conceptual tools that enabled us to understand the non-linear behaviour of complex systems over time using stocks, flows, internal feedback loops, and time delays and use them to design more effective policies and organizations. To build effective SD models, you should identify a problem that is worth the study, develop hypotheses about it causes and effects, build causal and flow graphs using your assumptions and then translate these graphs into DYNAMO (DYNAmics Model) programs.

## Unit Assessment

**Instructions:**

Attend all the questions below

## Grading

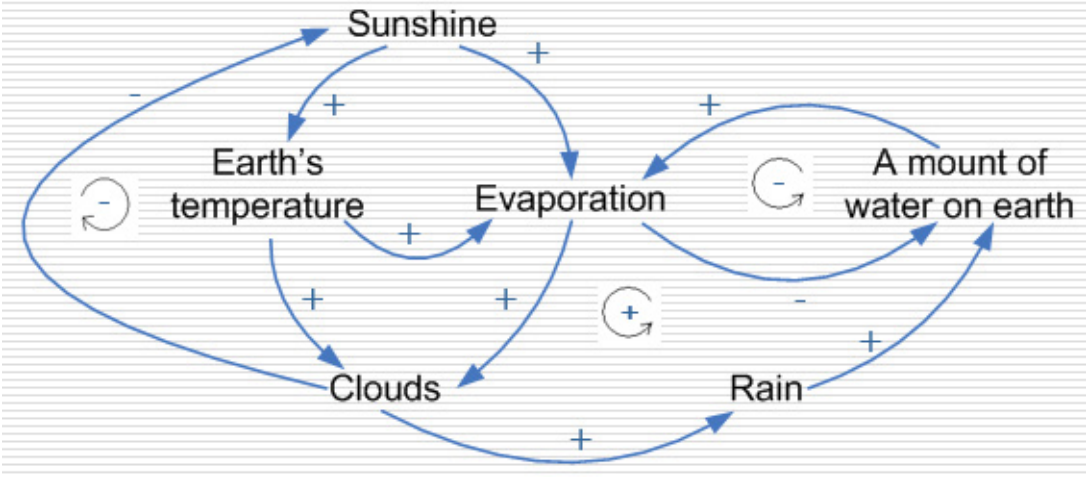
Maximum marks for each questions are in brackets

The maximum overall mark is 50

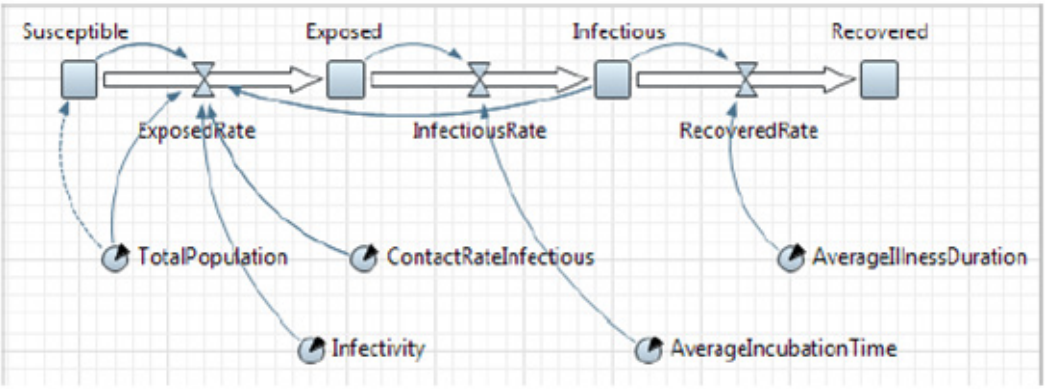
1. What is the difference between system dynamics and agent-based modelling? [5]
2. Explain why system dynamics is appropriate to simulate the propagation of an infectious disease like Ebola [5]
3. Give the difference between a positive and negative feedback in SD [3]
4. What is a stock in system dynamics? [2]
5. Draw and augment the following feedback loops [5]

   Study Hard → Pass Exam, Pass Exam → Study Hard

   Contract Virus → Fall Sick, Fall Sick → Contract Virus
6. What is a delay CLD? Give an example of CLD with a delay (different from the one in notes) [3]
7. Build the flow graph of the following CLD [10]




8. What is the meaning of the following flow graph built in AnyLogic? [7]




9. What is a DYNAMO Program? [2]

10. Give the DYNAMO Algorithm and Differential Equation of the following flow graph [8]

## Unit Readings and Other Resources

- John D. Sterman. (2000). Business Dynamics: Systems thinking and modeling for a Complex world, , McGraw-Hill
- Fishwick. P, A. (1995). Simulation Model Design and Execution: Building Digital Worlds, Prentice-Hall

- Roberts et al. (1983). Introduction to Computer Simulation: A System Dynamics Modeling Approach, Addison-Wesley
- Banks, J. & Carson, J.C. & Nelson, B. L. (19996). Discrete Event System Simulation," 2nd Edition, Prentice Hall

# Unit 4. Discrete Event Modelling and Simulation

## Unit Introduction

Discrete event modelling  is nearly the same age as system dynamics. In 1961, IBM engineer Geoffrey Gordon introduced GPSS, considered to be the first software implementation of the discrete event modeling method. Today, a number of programs including modern versions of GPSS - offer discrete event modeling.

Discrete event modeling requires a modeller to think about the system that he or she wants to model as a process - a sequence of operations that agents perform.

A model's operations can include delays, service by various resources, process branch selections, splits and many others. As long as agents compete for limited resources and can be delayed, queues will be part of nearly all discrete event models.

The model is specified graphically as a process flowchart where blocks represent operations. The flowchart usually starts with "source" blocks that generate agents and inject them into the process and ends with "sink" blocks that remove them.

Agents – originally named transactions in GPSS or entities in other simulation software can represent clients, patients, phone calls, physical and electronic documents, parts, products, pallets, computer transactions, vehicles, tasks, projects, ideas, and so forth. Resources represent staff, doctors, operators, workers, servers, CPUs, computer memory, equipment, and transport.

Service times and agent arrival times are usually stochastic, and since they're drawn from a probability distribution, discrete event models are themselves stochastic. In simple terms, this means a model must run for a specific amount of time or complete a specific number of replications before it produces meaningful output.

Typical output expected from a discrete event model includes:

- Utilization of resources;
- Time spent in the system or its part by an agent;
- Waiting times;
- Queue lengths;

System throughput.

Bottlenecks.

## Unit Objectives

Upon completion of this unit you should be able to:

- Understand and apply the main principles of discrete event simulation;
- Use simulation methods to model and analyze real discrete dynamic; systems;
- Apply the tools and techniques of simulation in a real life project that involves multiple scenarios and process improvement.

## Key Terms

**Discrete Event** :The great majority of processes we observe in the world consist of continuous changes. However, when we try to analyze these processes it often makes sense to divide a continuous process into discrete parts to simplify the analysis. Discrete Event Modeling techniques approximate continuous real-world processes with non-continuous events that you define.

Here are some examples of events:

A customer arrives at a shop,

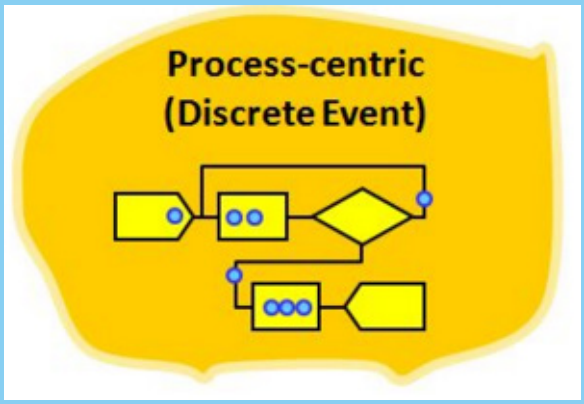
A truck finishes unloading,

A new product is launched,

Inventory levels reaches a certain threshold, etc.

In discrete event modeling the movement of a train from point A to point B would be modelled with two events, namely a departure event and an arrival event. The actual movement of the train would be modelled as a time delay (interval) between the departure and arrival events. This doesn't mean however that you can't model the train as moving. In fact, with AnyLogic you can produce visually continuous animations for logically discrete events.

*Figure 4.1: Discrete Event*



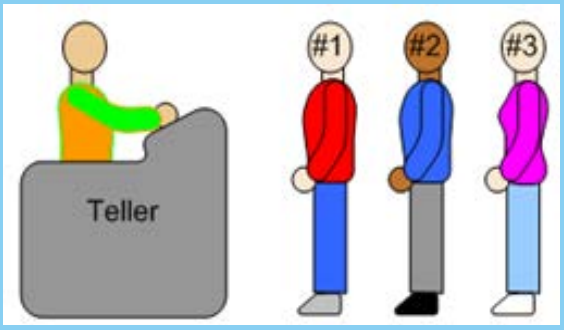

The term Discrete Event is however mainly used in the narrower sense to denote "Process-Centric" modeling that suggests representing the system being analyzed as a sequence of operations being performed on entities (transactions) of certain types such as customers, documents, parts, data packets, vehicles, or phone calls. The entities are passive, but can have attributes that affect the way they are handled or may change as the entity flows through the process. Process-centric modeling is a medium-low abstraction level modeling approach. Although each object is modelled individually as an entity, typically the modeller ignores many "physical level" details, such as exact geometry, accelerations, and decelerations. Process-centric modeling is used widely in the manufacturing, logistics, and healthcare fields.

**Agent:** When modelling and simulating, the modeller identifies the active entities, the agents (which can be people, companies, projects, assets, vehicles, cities, animals, ships, products, etc.), defines their behaviour (main drivers, reactions, memory, states, ...), puts them in a certain environment, establishes connections, and runs the simulation. The global behaviour then emerges as a result of interactions of many individual behaviours.

**Queue:** *A Queue is "an abstract data type in which elements are added to the rear and removed from the front; a 'first in, first out' (FIFO) structure."*

The main difference between a queue and a stack is that elements in a queue are put on the bottom and taken off the top (remember, in a stack, elements put on the top and taken off the top). For an everyday queue example, consider a line of customers at a bank waiting to be served. Each new customer gets in line at the rear. When the teller is ready to help a new customer, the customer at the front of the line is served. This is a queue--the first customer in line is the first one served.

*Figure 4.2: A Queue at a Bank*




**Clock**: Simulation must keep track of the current simulation time, in whatever measurement units are suitable for the system being modelled. In discrete-event simulations, as opposed to real-time simulations, time 'hops' because events are instantaneous – the clock skips to the next event start time as the simulation proceeds.

**Events List**: The simulation maintains at least one list of simulation events. This is sometimes called the pending event set because it lists events that are pending as a result of previously simulated event but have yet to be simulated themselves. An event is described by the time at which it occurs and a type, indicating the code that will be used to simulate that event. It is common for the event code to be parameterized, in which case, the event description also contains parameters to the event code.

**Ending Condition**: Because events are bootstrapped, theoretically a discrete-event simulation could run forever. So the simulation designer must decide when the simulation will end.

## Learning Activities

Activity 1: Representing Structure and Behavior

Imagine the Simulation!

**The Situation**

There are three Trucks that bring product from the Factory.

- On average, they take 3 days to arrive.
- Each truck brings somewhere between 10 and 20 products—all equally likely.
- We've got five Distributors who pick up product from the Factory with orders.
- Usually they want from 5 to 25 products, all equally likely.
- It takes the Distributors an average of 2 days to get back to the market, and an average of 5 days to deliver the products.

**Question: How much product gets sold like this?**

Don't Use Continuous Simulation

**i.e.:**

- We don't want to wait that number of days in real time.
- We don't even care about every day:
- There will certainly be timesteps (days) when nothing happens of interest.
- We're dealing with different probability distributions.
- Some uniform, some normally distributed.
- Things can get out of synchronization:
- A Truck may go back to the factory and get more product before a Distributor gets back.
- A Distributor may have to wait for multiple trucks to fulfill orders (and other Distributors might end up waiting in line)

### We Use Discrete Simulation

- We don't simulate every moment continuously.
- We simulate discrete events.

### No time loop:

In a discrete event simulation: There is no time loop:

- There are events that are scheduled;
- At each run step, the next scheduled event with the lowest time gets processed.
- The current time is then that time, the time that event is supposed to occur.

### Agent don't act:

In a discrete event simulations, agents don't act:

- Instead, they wait for events to occur.
- They schedule new events to correspond to the next thing that they're going to do.

### Agent get blocked:

Agents can't do everything that they want to do:

- If they want product (for example) and there isn't any, they get blocked.
  *They can't schedule any new events until they get unblocked.*

- Many agents may get blocked awaiting the same resource.
  *More than one Distributor may be awaiting arrival of Trucks*

**Key**: We have to keep track of the Distributors waiting *in line (in the* **queue***)*

### Key Ideas of Simulation

To have an (*not the*) answer to our question, we can take advantage the following theories:

1. Queuing Theory
2. Discrete Probability (Random *Discrete* Variables)

### Using Queues in Simulation

In general, queues are often used as "waiting lines". Here are a few examples of where queues would be used:

In operating systems, for controlling access to shared system resources such as printers, files, communication lines, disks and tapes.

A specific example of print queues follows:

In the situation where there are multiple users or a networked computer system, you probably share a printer with other users. When you request to print a file, your request is added to the print queue. When your request reaches the front of the print queue, your file is printed. This ensures that only one person at a time has access to the printer and that this access is given on a first-come, first-served basis.

For simulation of real-world situations. For instance, a new bank may want to know how many tellers to install. The goal is to service each customer within a "reasonable" wait time, but not have too many tellers for the number of customers. To find out a good number of tellers, they can run a computer simulation of typical customer transactions using queues to represent the waiting customers.

### Example of Algorithm of a Queue:

We can use a queue to simulate the flow of customers through a check-out line in a store. In this simulation we will have the following details:

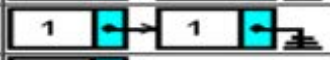
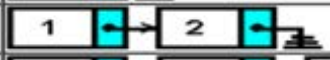
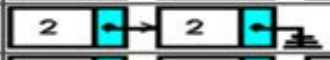
- One check-out line;
- The expected service time for each customer is one minute (However, they may have to wait in line before being serviced);
- Between zero and two customers join the line every minute.

We can simulate the flow of customers through the line during a time period n minutes long using the following algorithm:

```
Initialize the queue to empty.
for ( minute = 0 ; minute < n ; ++minute )
{
        if the queue is not empty, then remove the customer at the front of the queue.
        Compute a random number k between 0 and 3.
        If k is 1, then add one customer to the line.
        If k is 2, then add two customers to the line.
        Otherwise (if k is 0 or 3), do not add any customers to the line.
}
```

Given a time of 5 minutes, the following demonstrates the operations on the queue:

*Figure 4.3: Demonstration of Operations on a Queue*

| minute | queue isempty? | k | queue operations | corresponding diagram |
|---|---|---|---|---|
| 0 | yes | | no dequeue (empty queue) |  |
| | | 1 | enqueue(0) |  |
| 1 | no | | |  |
| | | | dequeue |  |
| | | 2 | enqueue(1) |  |
| | | | enqueue(1) |  |
| 2 | no | | |  |
| | | | dequeue |  |
| | | 2 | enqueue(2) |  |
| | | | enqueue(2) |  |
| 3 | no | | |  |
| | | | dequeue |  |
| | | 1 | enqueue(3) |  |
| 4 | no | | |  |
| | | | dequeue |  |
| | | 0 | no enqueue (k=0) |  |

What can we do with a queue?

- push(anObject): Tack a new object onto the tail of the queue
- pop(): Pull the end (head) object off the queue.
- peek(): Get the head of the queue, but don't remove it from the queue.
- size(): Return the size of the queue

**Builduing a Queue (Java):**

> Queue line = new Queue();

> line.push("Fred");

> line.push("Mary");

> line.push("Jose");

> line.size()

**Accessing a Queue (Java):**

> line.peek() “Fred”

> line.pop() “Fred”

> line.peek() “Mary”

> line.pop() “Mary”

> line.peek() “Jose”

> line.pop() “Jose”

> line.pop() java.util.NoSuchElementException:

**Queue Methods**

```
// Methods
/** Push an object onto the Queue */
 public void push(Object element)
{
elements.addFirst(element);
}
/** Peek at, but don’t remove, top of queue */
 public Object peek()
{
return elements.getLast();
}
/** Pop an object from the Queue */
 public Object pop()
{
Object toReturn = this.peek();
elements.removeLast();
return toReturn;
}
/** Return the size of a queue */
public int size() { return elements.size();}
```

### Discrete Random Variable

Here we give definition of random variable. In most practical cases the random variables are either discrete or continuous. In the present section discrete random variables will be discussed. A discrete random variable is defined as:

A random variable X and its corresponding distribution are said to be discrete, if X has the following properties.

(1) The number of values for which X has a probability different from 0, is finite or utmost countable infinite.

(2) Each finite interval on the real line contains at most finitely many of those values. If an interval a ≤X ≤ b does not contain such a value, then P (a < X ≤ b) = 0. Here a and b are the upper and lower limits of the stochastic variable X.

If x_1, x_2, x_3 be values for which X has positive corresponding probabilities p_1, p_2 p_3, then function

$$f(x) = \Pr(X = x_j) = p_j, \quad j = 1, 2, \ldots, n$$
$$= 0, \text{ otherwise}$$

with the condition,

$$\left.\begin{array}{l} (i)\ f(x) \geq 0 \\ (ii)\ \sum_x f(x) = 1 \end{array}\right\}$$

### Expected Value and Variance of Random Discrete Variables

A quantity called Expected Value (EV), which is associated with every random variable will be discussed in this section. Expected value, also called mean of the discrete random data denoted by μ, is the sum of the product of all the values, a random variable takes, with its probabilities assigned at those values. Thus expected value is defined as

$$E(X) = \Sigma_{R(x)} \Pr(X = x) \cdot x$$
$$= \Sigma_{R(x)} f(x) \cdot x$$

**Example**: In the Table below, the probability of arrival of the sum of the numbers on two faces of two dice thrown simultaneously, are given. Calculate the value of E(x)?

Table : Probability of occurrence of number (X) in a throw of two dice

| Number expected (X) | Probability f(x) | Number expected (X) | Probability f(x) |
|---|---|---|---|
| 2 | $\frac{1}{36}$ | 7 | $\frac{6}{36}$ |
| 3 | $\frac{2}{36}$ | 8 | $\frac{5}{36}$ |
| 4 | $\frac{3}{36}$ | 9 | $\frac{4}{36}$ |
| 5 | $\frac{4}{36}$ | 10 | $\frac{3}{36}$ |
| 6 | $\frac{5}{36}$ | 11 | $\frac{2}{36}$ |
| | | 12 | $\frac{1}{36}$ |

**Solution**: It can be seen from the table that:

$$E(x) = \sum f(x) \cdot x$$
$$= (1/36) \cdot [2 + 2 \times 3 + 3 \times 4 + 4 \times 5 + 5 \times 6 + 6 \times 7 + 5 \times 8 + 4 \times 9 + 3 \times 10 + 2 \times 11 + 12]$$
$$= 7$$

Which is nothing but the mean of the values of X

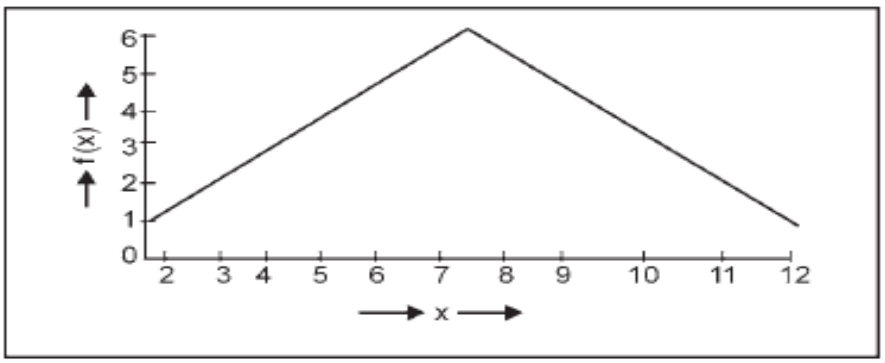
Another important quantity closely associated with every random variable is its variance. Basically, the variance is a measure of the relative spread in the values, the random variable takes on. In particular, the variance of a discrete random variable is defined as

$$\begin{aligned} \text{Var}(X) &= E\ \{X - E(X)\}^2 \\ &= \Sigma_{R(X)} \Pr(X = x)[x - E(x)]^2 \\ &= \Sigma_{R(X)} f(x)\left[x - E(x)\right]^2 \end{aligned}$$

**Example:** Determine the variance (Var) of the random variables X given in the Table above

*Solution:*

$$\begin{aligned} \text{Var}\,(x) &= \Sigma f(x) \cdot [x - E(x)]^2 \\ &= 1/36[\ 1(-5)^2 + 2(-4)^2 + 3(-3)^2 + 4(-2)^2 + 5(-1)^2 + 6(0)^2 \\ &\qquad + 5(1)^2 + 4(2)^2 + 3(3)^2 + 2(4)^2 + 1(5)^2] \\ &= 1/36[25 + 32 + 27 + 16 + 5 + 0 + 5 + 16 + 27 + 32 + 25] \\ &= 210/36 \\ &= 5.83333 \end{aligned}$$




*Probability of occurrence of number in a throw of two dice.*

**Some Important Distribution Functions**

- **Cumulative Distribution Function**

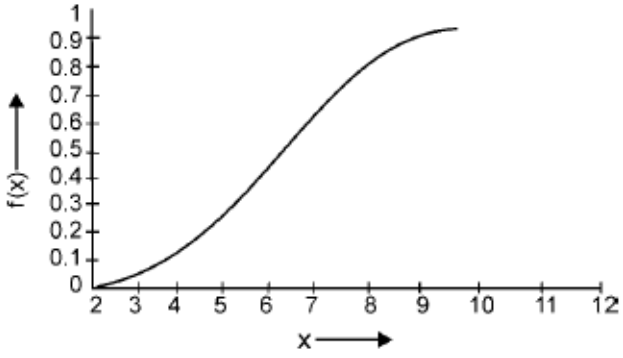
Closely related to Probability Distribution Function (PDF) is a function called **Cumulative Distribution Function** (PDF). This function is denoted by F(x). Cumulative distribution function of a random variable X is defined as

$$F(x) = \Pr(X \leq x)$$

which can be expressed as,

$$F(x) = \sum_{X \leq x} f(z)$$

F(x) represents a probability that the random variable X takes on a value less than or equal to x. We can see that in the example of rolling pair of dice only two outcomes less than or equal to three are f (2) and f (3), thus




*Variation of F(x) vs. x in case of throwing of two dice.*

$$F(3) = \Pr(X \leq 3) = f(2) + f(3)$$

$$= \frac{1}{36} + \frac{2}{36} = \frac{3}{36}$$

- **Uniform Distribution Function**

The uniform distribution has wide application in various problems in modeling. Flow of traffic on road, distribution of personnel in battle field, and distribution of stars in sky are examples of uniform distribution. It is the basic distribution required for calculating the other distributions. For the evaluation of different types of warheads, it is generally assumed that the distribution of ground targets (say distribution of personnel and vehicles in a battle field) is randomly uniform. Even most of the warheads have sub-munitions with uniform ground patterns.

Cumulative distribution function of uniform distribution is:

$$F(x) = \int_a^x \frac{1}{b-a}\, dx = \frac{x-a}{b-a}$$

**Binomial Distribution Function**

Suppose an experiment can yield only two possible outcomes, say 0 or 1, where 0 represents a failure and 1 represents a success. For example on tossing a coin we get either a head or a tail. If head is the outcome of tossing, we say experiment is a success i.e., p = 1, otherwise it is a failure (p = 0), where p is the probability of a success in each trial of an experiment. Now repeat the trial n times under identical conditions. If x represent the number of successes in the n experiments, then x is said to have a binomial distribution whose PDF is given by:

$$f(x) = \Pr(X = x) = C_x^n p^x (1-p)^{n-x}$$

where, $x = 0,1,\ldots, n; \quad 0 < p < 1$

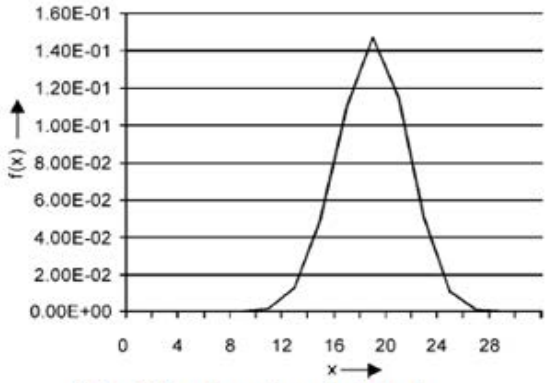
**Example**: In an examination, total thirty students appeared. If probability of passing the examination of one student is 0.6 (i.e., p = 0.6), then what is the probability that none of the student will pass and twenty students will pass.

**Solution:**

This problem can easily be solved by Binomial distribution. Probability of failing all the students will be f (0) and passing of 20 students will be f (20). Thus using the equation:

$$f(0) = C_0^{30}(0.6)^0(0.4)^{30} = 1153 \times 10^{-12}$$

$$f(20) = C_{20}^{30}(0.6)^{20}(0.4)^{10} = 0.115185$$




*Probability of passing of students.*

- **Poisson Distribution**

Another distribution which will be used in combat survivability analysis is Poisson's distribution after SD Poisson. This distribution function is required for the cases, where probability of success of an event is very low in large number of trials. Interestingly, this probability density function was first devised to study the number of cavalry soldiers who died annually due to direct hit by horses on their head. The PDF for a random variable X has the variates.

$$f(x) = \Pr(X = x)$$

$$= \frac{e^{-\lambda}\lambda^{x}}{x!}, \text{ where } x = 0, 1, 2,\ldots$$

$$\text{and } \lambda > 0.$$

Here **λ** is the expected value of the Poisson's distribution. Poisson's distribution is used in the case where out of large number of trials, probability of success of an event is quite low.

Activity 2 : Laboratory

**Job Shop Model**

**Laboratory Objective:**

Our goal is to create a discrete-event model that will simulate a small job shop's manufacturing and shipping processes. The raw materials that are delivered to the receiving dock are placed into storage until processing takes place at the CNC machine.

**Instructions:**

This laboratory consists of 4 Phases. The first phase is depicted in this activity. Attend it and continue the 3 remaining phases in our practical book (AnyLogic 7 in 3 Days)

**Grading:**

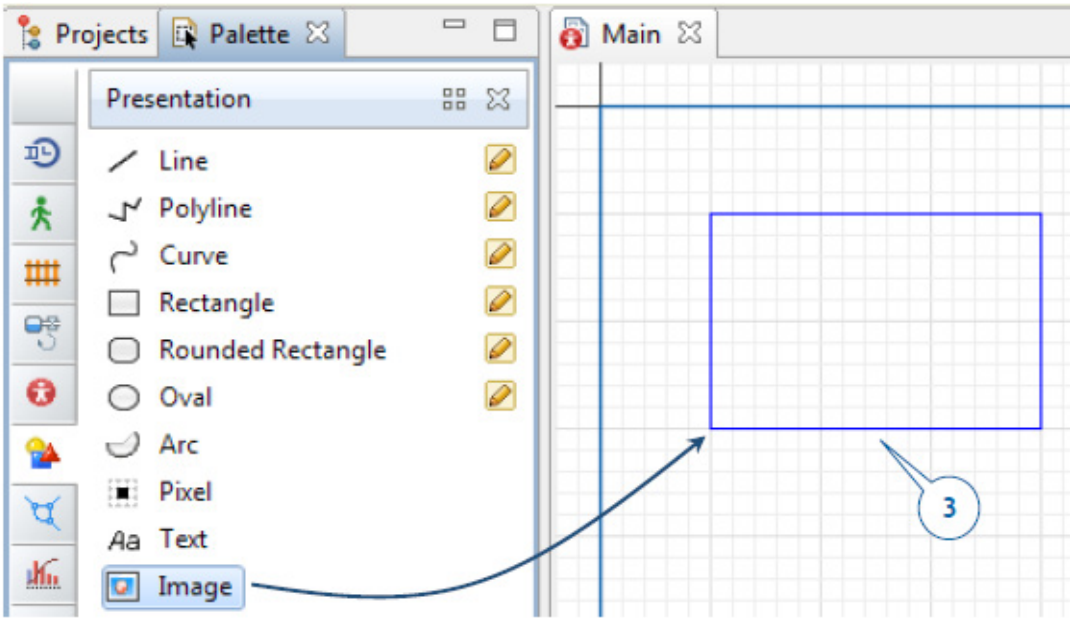
- The maximum mark of this laboratory is 60

**Time Required**: 3 Days

### Part 1: Creating a Simple Model

We'll start by creating a simple model that will simulate the pallets' arrival at the job shop, their storage at the shipping dock, and their arrival at the forklift area.

1. Create a new model. In the **New Model** wizard, set the **Model name**: Job Shop, and **Model time units: minutes**.
2. Open the **Presentation** palette. The palette has several shapes that you can use to draw model animation, including a rectangle, a line, an oval, a polyline and a curve.
3. On the **Presentation** palette, select the **Image** shape and then drag it on to the Main diagram. You can use the **Image** shape to add images in several graphic formats -- including PNG, JPEG, GIF, and BMP – to your presentation.

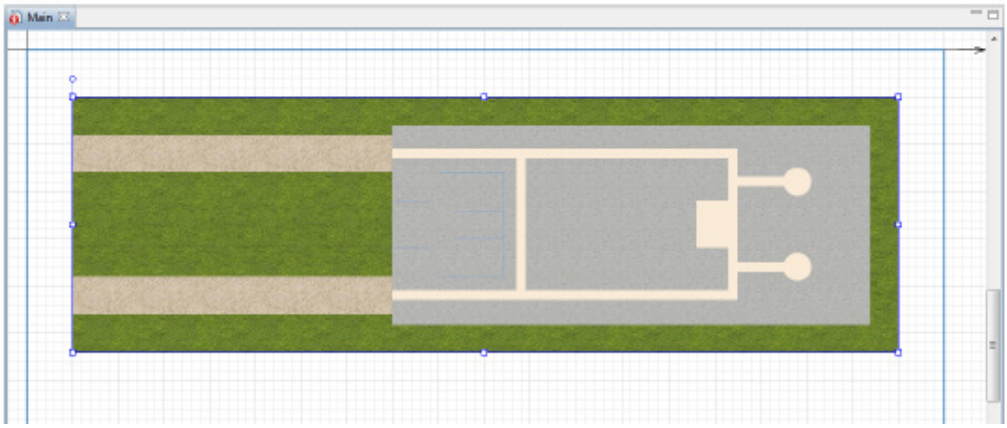



4. You'll see the dialog box that prompts you to choose the image file the shape will display.
5. Browse to the following location and then select the layout.png image:

   AnyLogic folder/resources/AnyLogic in 3 days/Job Shop

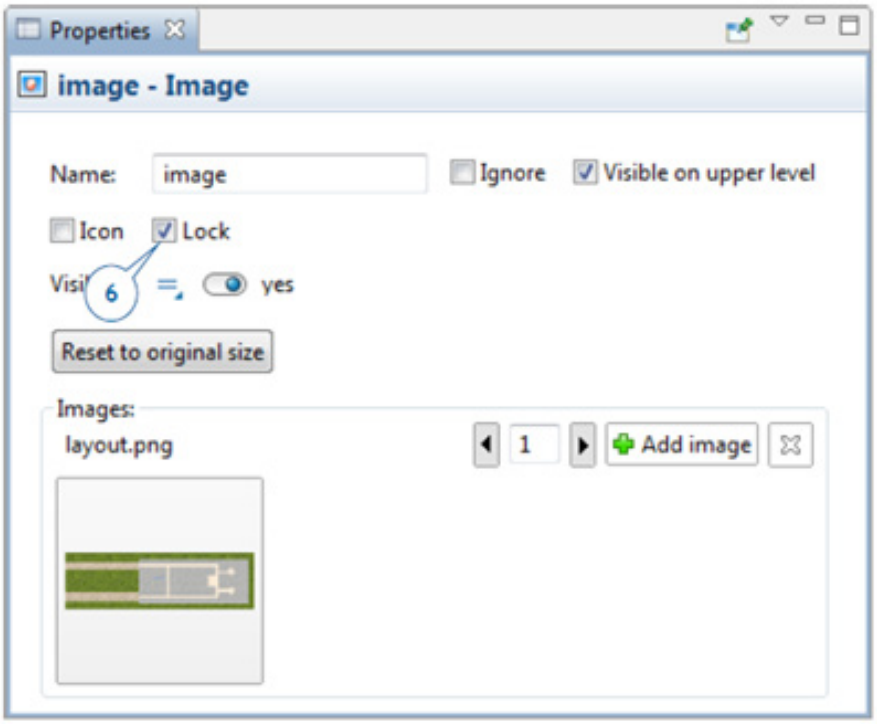
   After you select the layout.png image, our diagram of the Main agent type

   should look like the following image:



AnyLogic adds the image in its original size on to the Main diagram, but you can also change the image's width or length. If you distort the image's proportions as in the figure below, you can revert to the image's original size by opening the **Properties** view and clicking **Reset to original size.**

6. Select the image in the graphical editor. In the **Properties** view, select the **Lock** checkbox to lock the image.



### Locking shapes

You can lock a shape to ensure it doesn't respond to your mouse click and you can't select it in the graphical editor. You'll find this very helpful as you draw shapes on top of layouts that represent facilities such as factories or hospitals.

If you need to unlock a shape, right-click in the graphical editor and select **Unlock All Shapes** from the menu.

### Space markup elements

Our next step is to use the **Space Markup** palette to place space markup shapes on top of the job shop's layout. The palette includes a **Path** element, three **Node** elements, an **Attractor** element, and **Pallet Rack** shapes.

### Creating a network

*Paths* and *nodes* are space markup elements that define the locations of agents:

A **Node** is a place where agents may reside or perform an operation.

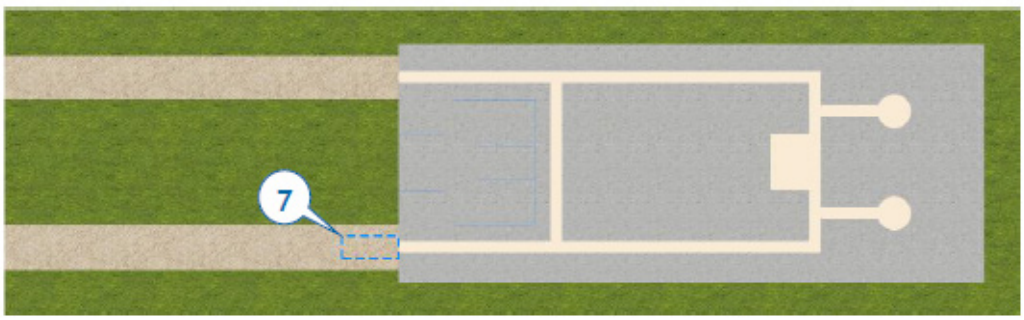
A **Path** is a route that agents can use to move between nodes.

Together, nodes and paths make up a network that a model's agents can use to move along the shortest paths between their origin and destination nodes. You'll usually create a network when your model's processes take place in a defined physical space and it has moving agents and resources. It is assumed that network segments have unlimited capacity, and the agents do not interfere with one another.

Now that you know a little bit about networks and their component parts, we're ready to create a network that will define the movement paths for our model's pallets. The first step is to use rectangular nodes to define specific areas on the job shop's layout.

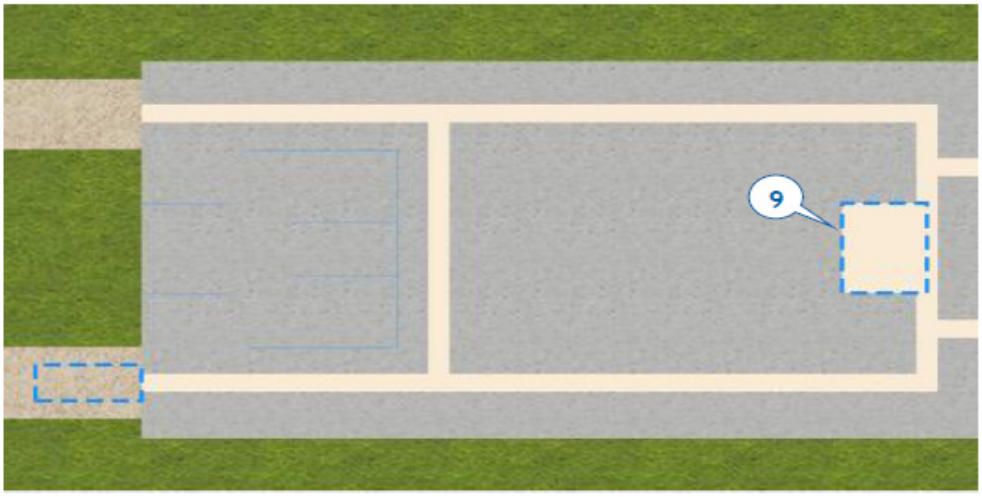
Draw the rectangular node over the job shop's entrance, as shown in the figure below, to represent our model's pallet receiving dock.

7. Open the **Space Markup** palette, and drag the **Rectangular Node** element on to the Main diagram. Resize the node. The node should look as in the figure below.




8. Name the created node receivingDock.

9. Draw a node to define the location where the model's agents will park forklift trucks once the trucks are idle or the agents no longer need them to complete a task. Use another **Rectangular node** to draw the parking area as shown in the figure below and then name this node forkliftParking.



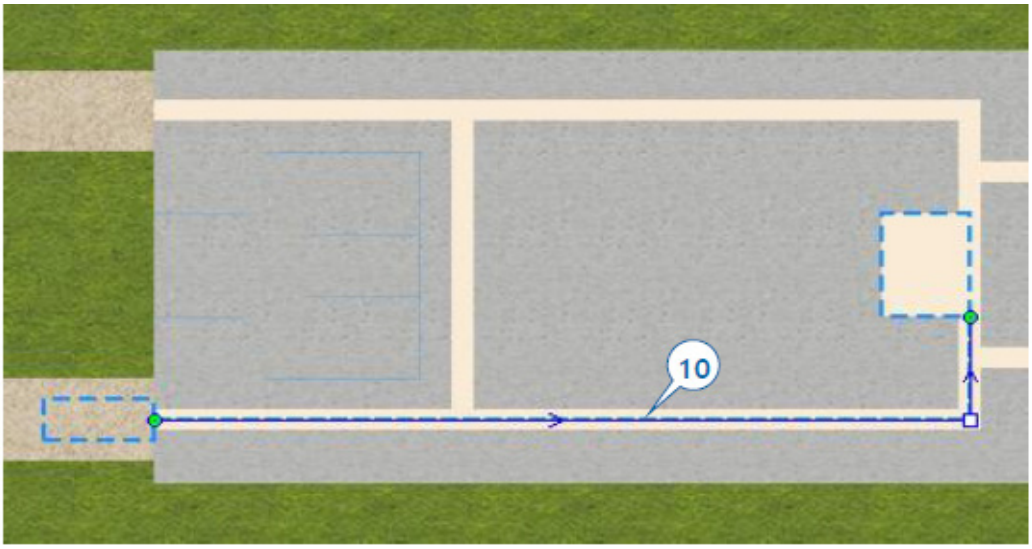

Let's draw a movement path to guide our model's forklift trucks.

10. 10. Do the following to draw a movement path that will guide our model's forklift trucks:

    a. In the Space Markup palette, double-click the Path element to activate its drawing mode.

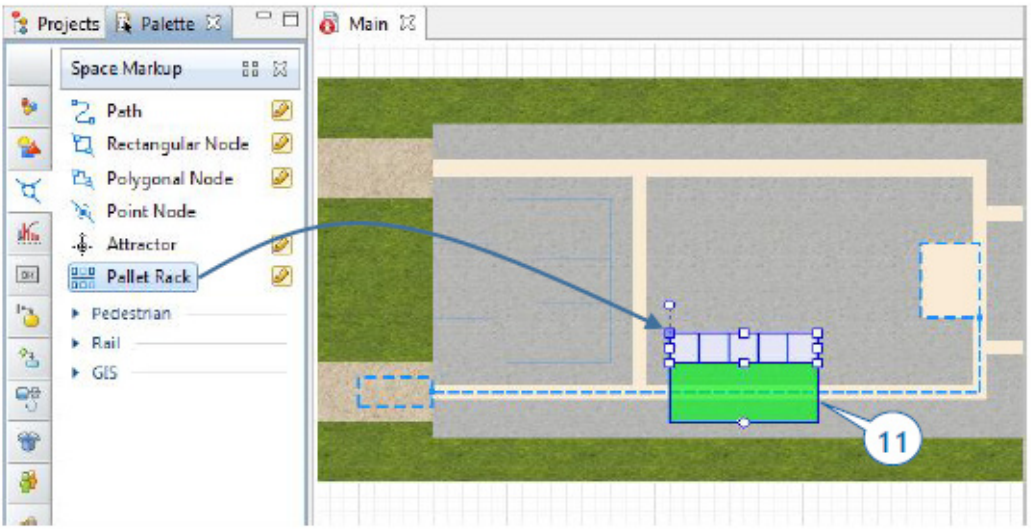
    b. Draw the path as shown in the figure below by clicking the receivingDock border, clicking in the diagram to add the path's turning point, and then clicking the forkliftParking node's border.

    If you've successfully connected the nodes, the path's connection points will display cyan highlights each time you select the path.



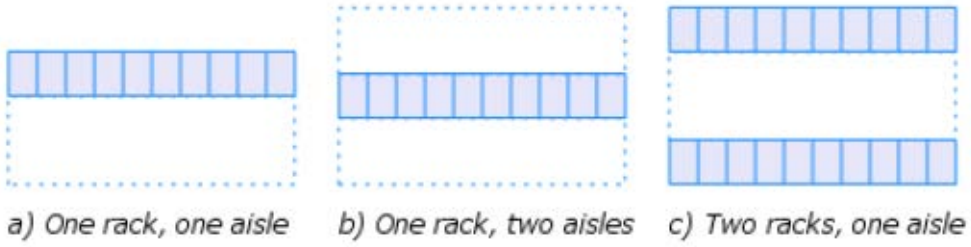
By default, paths in AnyLogic 7 are bidirectional. However, you can limit movement along a selected path to one direction by clearing the Bidirectional property and then defining the movement direction. You can view a given path's direction by selecting the path and then viewing the direction arrow that displays in the graphical editor.

11. Define your model's warehouse storage by dragging the **Pallet Rack** element from the **Space Markup** palette on to the layout and placing its aisle on the path. A correctly-placed pallet rack will display a green highlight that shows it is connected to the network.




### Pallet rack

The **Pallet Rack** space markup element graphically represents the pallet racks you often see in warehouses and storage zones. As you can see below, the element has three alternative configurations




During runtime, the **Pallet Rack** element manages the agents that the model stores in the single-level or multiple level cells that are available on side(s) of the aisle.

12. In the pallet rack's **Properties** area, do the following:

    a. Set Type to: two racks, one aisle

    b. Number of cells: 10

    c. Level height: 10

    In the Position and size section:

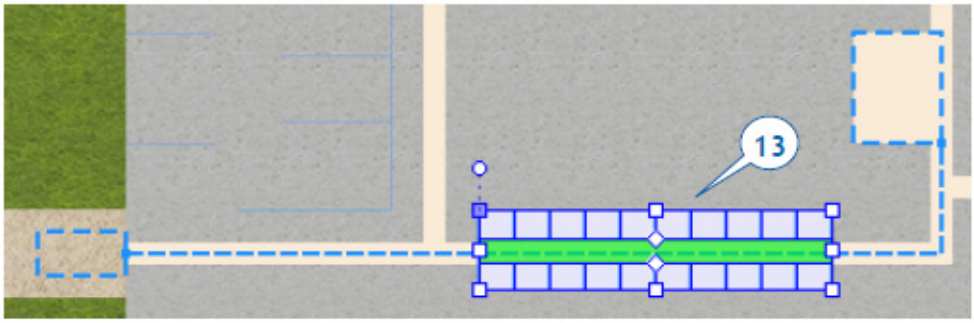
    d. Length: 160

    e. Left pallet rack depth: 14

    f. Right pallet rack depth: 14

    g. Aisle width: 11

13. After you've completed these changes, the pallet rack should resemble the pallet rack shown in the figure below. If necessary, move the pallet rack so that its center aisle lies on the path. Make sure the pallet rack is connected to the network by clicking it twice to select it. Your first click will select the entire network, and the second will select the pallet rack. The pallet rack should display a green highlight that shows it is connected to the network.

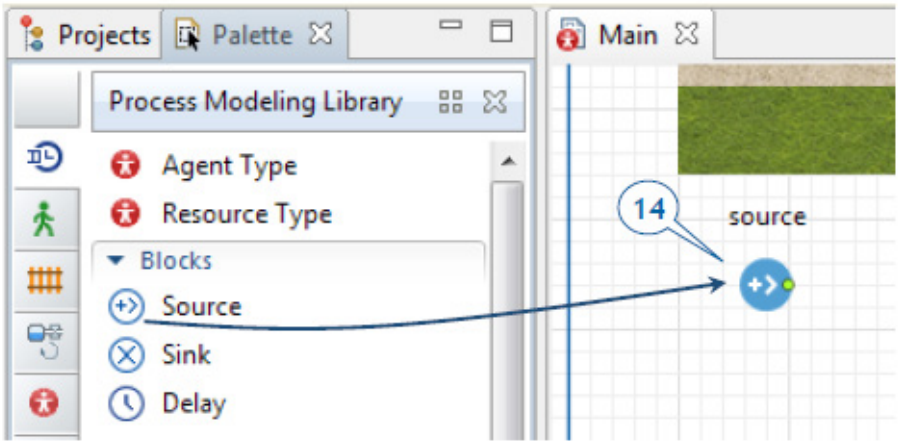



We've marked up our model's space by drawing the important locations and paths on top of our layout, and we'll now use the AnyLogic ***Process Modeling Library*** to model the processes.

### Process Modeling Library

The blocks in AnyLogic's ***Process Modeling Library*** allow you to use combinations of ***agents, resources***, and ***processes*** to create process-centric models of real-world systems. You learned about agents and resources earlier in this section, and we'll build upon that foundation by defining processes as operations sequences that include queues, delays, and resource utilization.

Your model's processes are defined by ***flowcharts***, the graphical process representations you construct from the Process Modeling Library's blocks. In the following steps, you'll create the process flowchart.




14. Drag the **Source** element from the **Process Modeling Library** palette on to the graphical diagram and name the block sourcePallets.

    While the **Source** block usually acts as a process starting point, our model will use it to generate pallets.

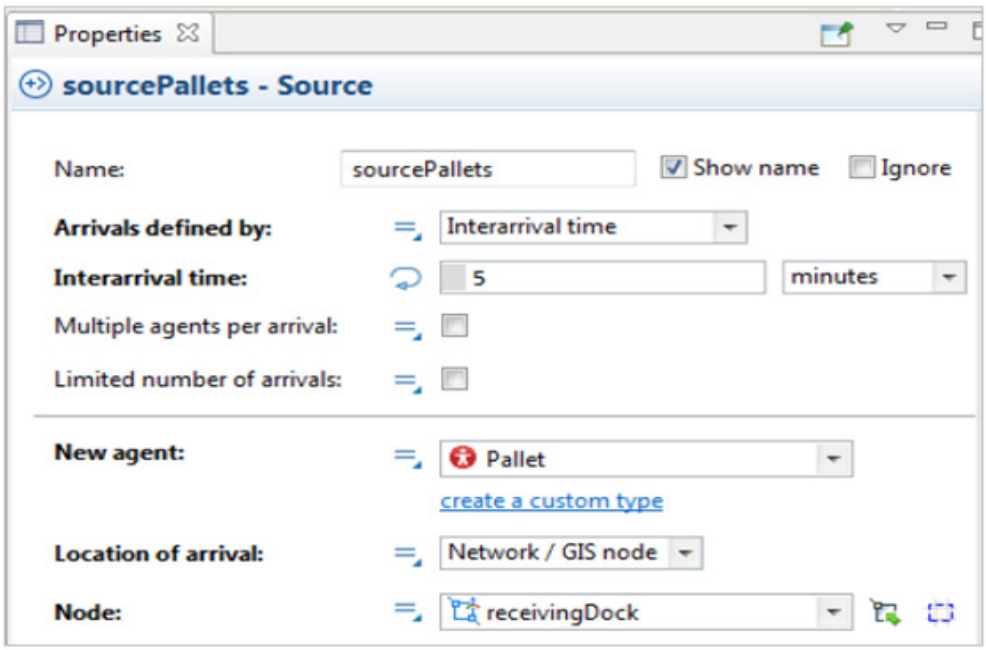
15. In the sourcePallets block's **Properties** area, do the following to ensure the model's pallets arrive every five minutes and appear in the receivingDock node.

    a. In the **Arrivals defined** by area, click **Interarrival time**.

    b. In the **Interarrival time** box, type 5, and select **minutes** from the list on the right to have pallets arrive every five minutes.

    c. In the **Location of arrival** area, click **Network / GIS node** in the list.

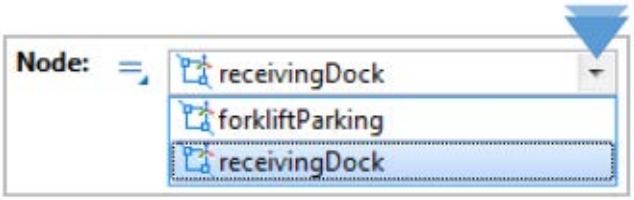
    d. In the **Node** area, click receivingDock in the list.



### How to refer to model elements from block's parameters

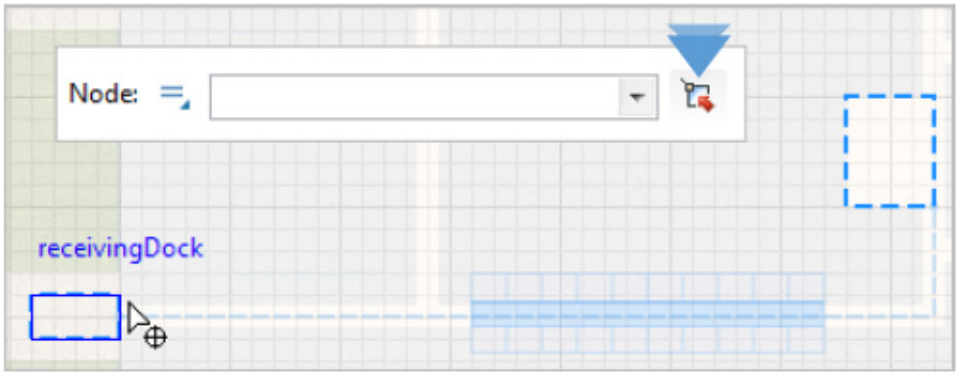
The block's parameters offer two ways to select a graphical element:

You can select a graphical element from the list of available and valid elements that displays beside the parameter.




You can select a graphical element by clicking the selection button that displays beside the list. If you click the selection button, it will limit your choices to the available and valid elements that you can select by clicking in the graphical editor:

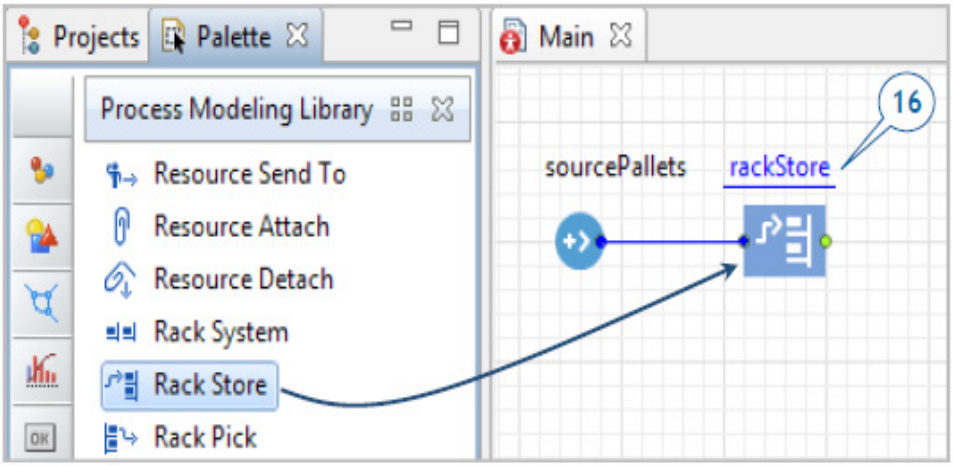
You can select a graphical element by clicking the selection button that displays beside the list. If you click the selection button, it will limit your choices to the available and valid elements that you can select by clicking in the graphical editor:




Continue constructing the flowchart by adding other **Process Modeling Library** blocks:

16. Drag the **RackStore** block from the **Process Modeling Library** palette on to the diagram and place it near the sourcePallets block so they are automatically connected as shown in the diagram below.

The **RackStore** block places pallets into a given pallet rack's cells.

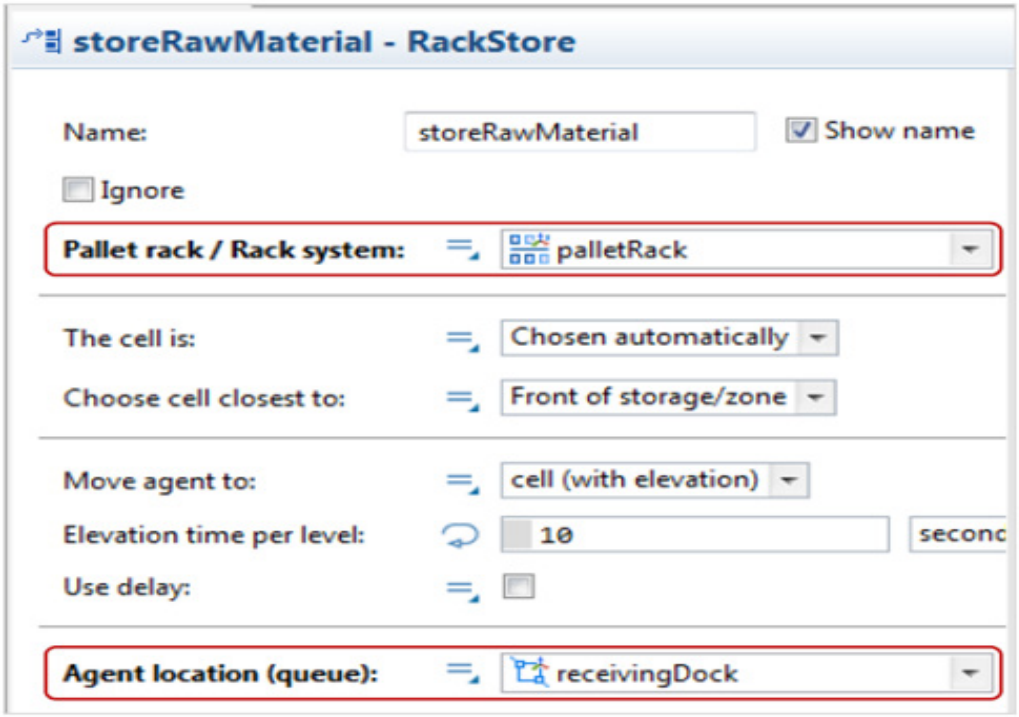



17. In the rackStore block's **Properties** area, do the following:

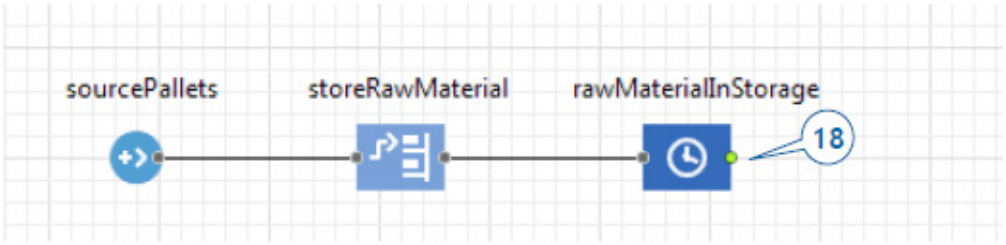
    a. In the **Name** box, type storeRawMaterial.

    b. In the **Pallet rack / Rack system** list, click palletRack.

    c. In the **Agent location (queue)** list, click receivingDock to specify the location where agents wait to be stored.




18. Add a **Delay** block to simulate how pallets wait in the rack and then name the block rawMaterialInStorage.

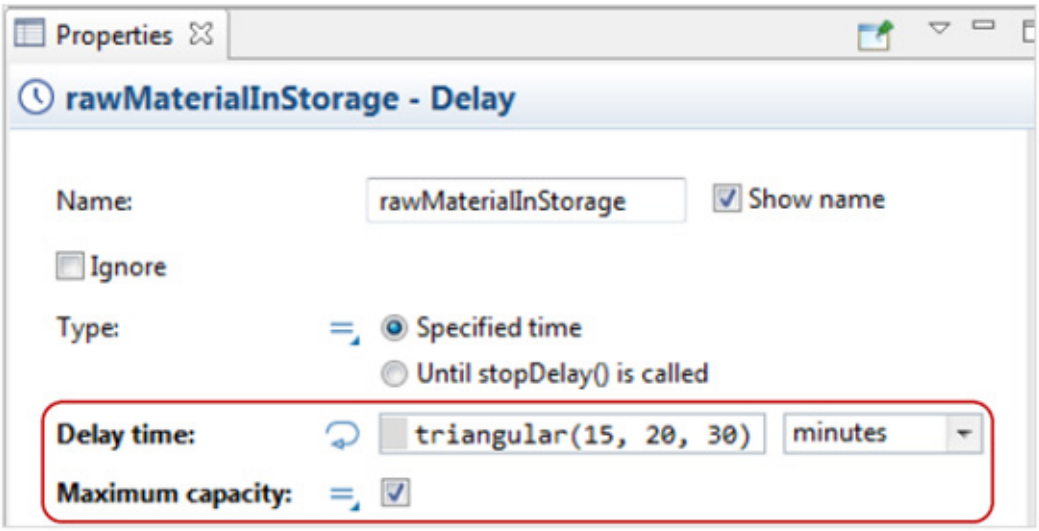



You've probably noticed that AnyLogic automatically connects the block's right port to the following block's left port. Each Process Modeling Library block has a left input port and a right output port, but you should only connect input ports to output ports.

19. In the rawMaterialInStorage block's **Properties area**, do the following:

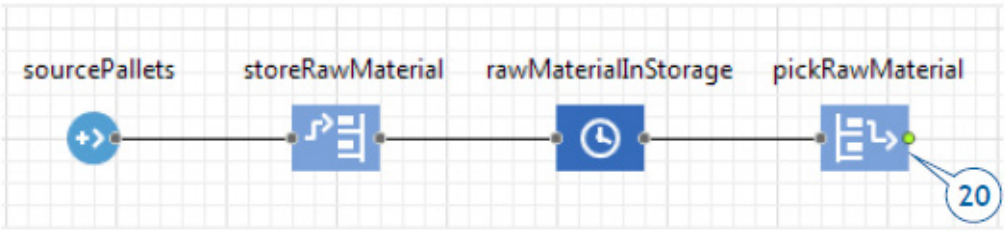
    a. In the **Delay time** box, type triangular(15, 20, 30) and select **minutes** from the list.

    b. Select the **Maximum capacity** checkbox to ensure agents will not get stuck as they wait to be picked up from storage.




20. Add a **RackPick** block, connect it to the flowchart, and then name it pickRawMaterial.

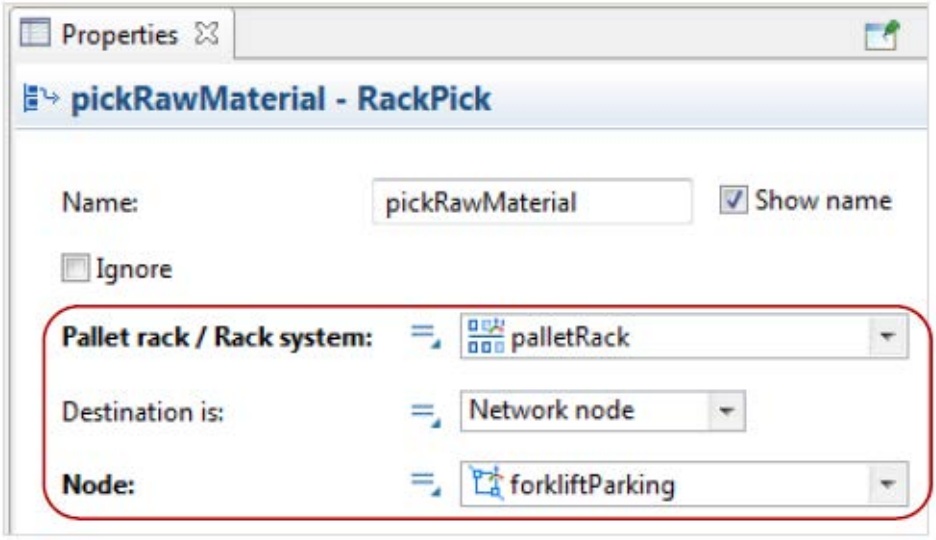
    In our model, the **RackPick** block removes a pallet from a cell in the pallet rack and then moves it to the specified destination.



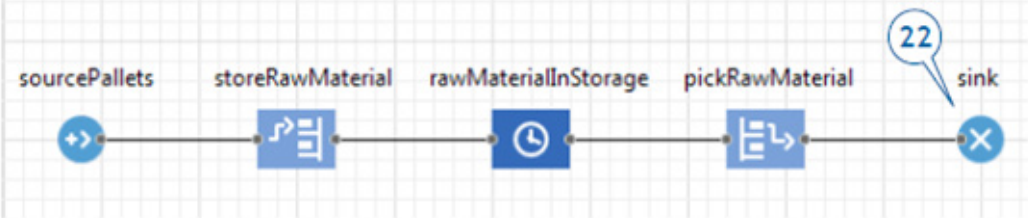
21. In the pickRawMaterial block's **Properties** area, do the following:

    a. In the P**allet rack / Rack system** list, click palletRack to select the pallet rack that will provide pallets to agents.

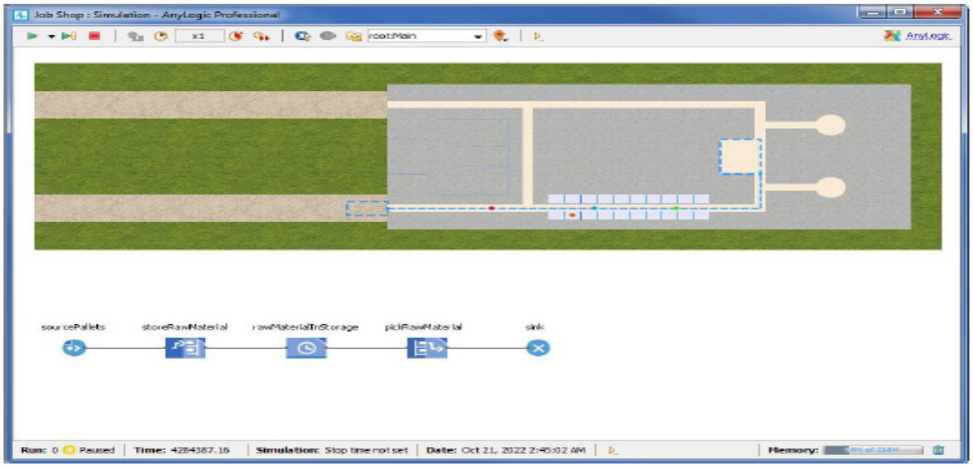
    b. In the Node list, click forkliftParking to specify where the agents should park forklift trucks.



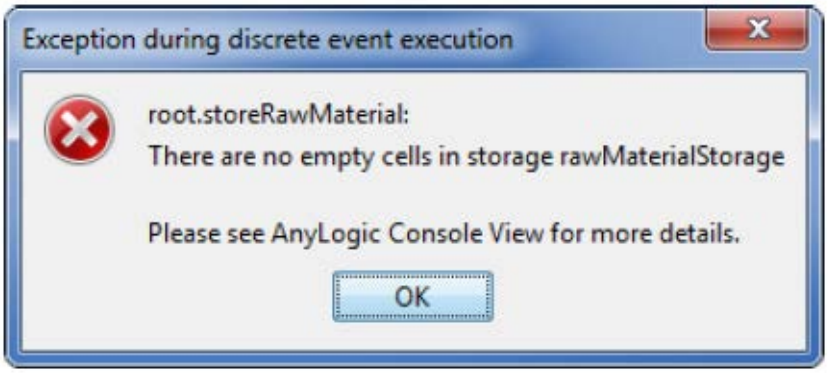

22. Add a **Sink** block. The **Sink** block disposes agents and is usually a flowchart's end point.




23. We've finished building this simple model, and you can now run it and observe its behavior. Run the model (Job Shop / Simulation experiment).



If the **Exception during discrete event execution** error message displays, you must connect your pallet rack to the network. You should select the pallet rack shape in the graphical editor, move it until the pallet rack's aisle displays a green highlight that shows it has connected to the network, and then rerun the model.




## Unit Summary

In this Unit, we have gone through the process of codifying the behaviour of complex systems with highly variable scholastic parameters and constrained resources in order to improve the organization of delivered services.

Discrete Event Modelling is mainly used in systems seen as a sequence of operations being performed on entities (transactions) of certain types such as customers, documents, parts, data packets, vehicles, or phone calls. The entities are passive, but can have attributes that affect the way they are handled or may change as the entity flows through the process. Discrete event modeling supports medium and medium-low abstraction. Although each object is modelled individually as an entity, typically the modeller ignores many "physical level" details, such as exact geometry, accelerations, and decelerations. This modeling approach is used widely in the manufacturing, logistics, and healthcare fields.

## Unit Assessment

**Instructions:**

Attend all the questions below

## Grading

Maximum marks for each questions are in brackets

The maximum overall mark is 55

1. What is the difference between a discrete and continuous event [5]
2. Briefly explain why agents act in DEV as compared to ABM [5]
3. What is Monte Carlo Simulation? Briefly explain how it is different from DEV [5]
4. Explain why scholastics and queues are important in DEV [5]
5. Determine whether each of the following characteristics apply to a stack, a queue, both, or none. [5]

    a. An element is inserted at a special place called the top.

    b. An element is inserted at a special place called the rear.

    c. The structure can hold only one type of data element.

    d. An element is deleted at the front.

    e. The ith position may be deleted.

    f. An element is deleted at the top.

    g. The structure is a LIFO structure.

    h. The structure is a FIFO structure.

    i. The structure is a restricted access structure.

Suppose q is an instance of the Queue class and assume that the previous array implementation is used. Also, assume that the size of the array is 5. Show q after all of the following operations have been completed assuming the queue is empty to start with. Show how the front, rear and elements change. [5]

```
q.enqueue (39);
q.enqueue (22);
item1 = q.dequeue();
q.enqueue (59);
item2 = q.dequeue ();
item3 = q.dequeue ();
```

6. Give the algorithm for dequeue [5]

7. Random queue. Create an abstract data type RandomizedQueue.java that supports the following operations: isEmpty(), insert(),random(), and removeRandom(), where the deletion operation deletes and returns a random object. Hint: maintain an array of objects. To delete an object, swap a random object (indexed 0 through N-1) with the last object (index N-1). Then, delete and return the last object. [5]

public class RandomQueue<Item> (generic random queue)

| | | |
|---|---|---|
| | RandomQueue() | *create a random queue* |
| boolean | isEmpty() | *is the queue empty?* |
| void | enqueue(Item item) | *add an item* |
| Item | dequeue() | *remove a random item (sample without replacement)* |
| Item | sample() | *return a random item (do not remove) (sample with replacement)* |

A supervisor in a manufacturing plant has three men and three women working for him. He wants to choose two workers for a special job. Not wishing to show any biases in his selection, he decides to select the two workers at random. Let Y denote the number of women in his selection. Find the probability distribution for Y. [5]

When the health department tested private wells in a county for two impurities commonly found in drinking water, it found that 20% of the wells had neither impurity, 40% had impurity A, and 50% had impurity B. (Obviously, some had both impurities.) If a well is randomly chosen from those in the county, find the probability distribution for Y, the number of impurities found in the well. [5]

The probability distribution for a random variable Y is given in table. Find the mean, variance, and standard deviation of Y. [5]

| $y$ | $p(y)$ |
|---|---|
| 0 | 1/8 |
| 1 | 1/4 |
| 2 | 3/8 |
| 3 | 1/4 |

## Unit Readings and Other Resources

### Core Text


- Grigoryev, L. (2015). AnyLogic 7 in Three Days, A Quick Course in Simulation Modelling, ISBN-13: 978-1508933748


### Background Text


- Singh. V, P. (2009). Simulation and Modelling. New Age International (P) Ltd., Publishers
- John D. Sterman. (2000). Business Dynamics: Systems thinking and modeling for a Complex world, , McGraw-Hill
- Banks, J. & Carson, J.C. & Nelson, B. L. (19996). Discrete Event System Simulation," 2nd Edition, Prentice Hall

**The African Virtual University Headquarters**

Cape Office Park

Ring Road Kilimani

PO Box 25405-00603

Nairobi, Kenya

Tel: +254 20 25283333

contact@avu.org

oer@avu.org

**The African Virtual University Regional Office in Dakar**

Université Virtuelle Africaine

Bureau Régional de l'Afrique de l'Ouest

Sicap Liberté VI Extension

Villa No.8 VDN

B.P. 50609 Dakar, Sénégal

Tel: +221 338670324

bureauregional@avu.org