ABM for Management Decision Support [Spring 2013]

Lab 1

Practising Agent-Based Model Design and Implementation

Peer-Olaf Siebers

The University of Nottingham
UNITED KINGDOM · CHINA · MALAYSIA

pos@cs.nott.ac.uk

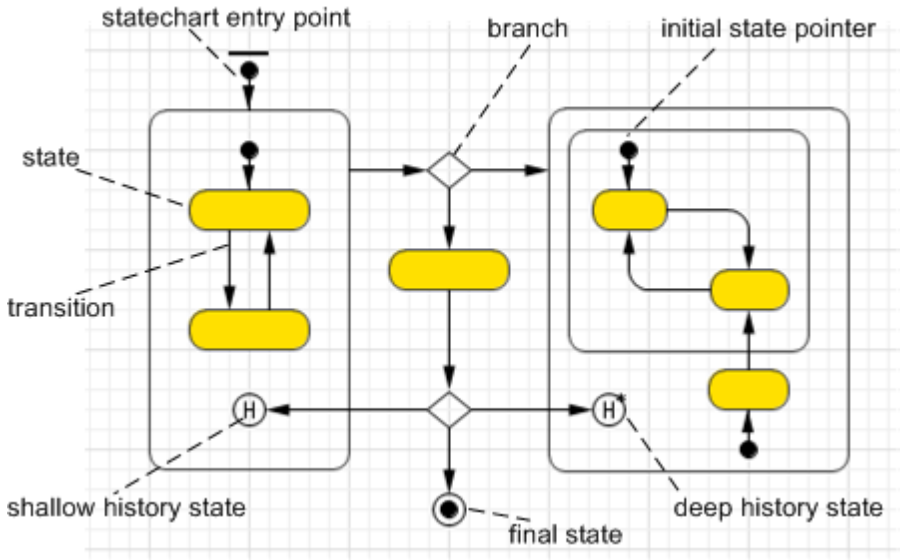
Content Tutorial

- Part 1: Modelling Practice
  - Representing Agent-Based Models
  - Conceptual Modelling Exercise

- Part 2: Model Implementation Practice
  - The AnyLogic IDE
  - Agent-Based Simulation Tutorial

1 Representing Agent-Based Models

- Defining behaviour by using state charts (alias state machine diagrams)



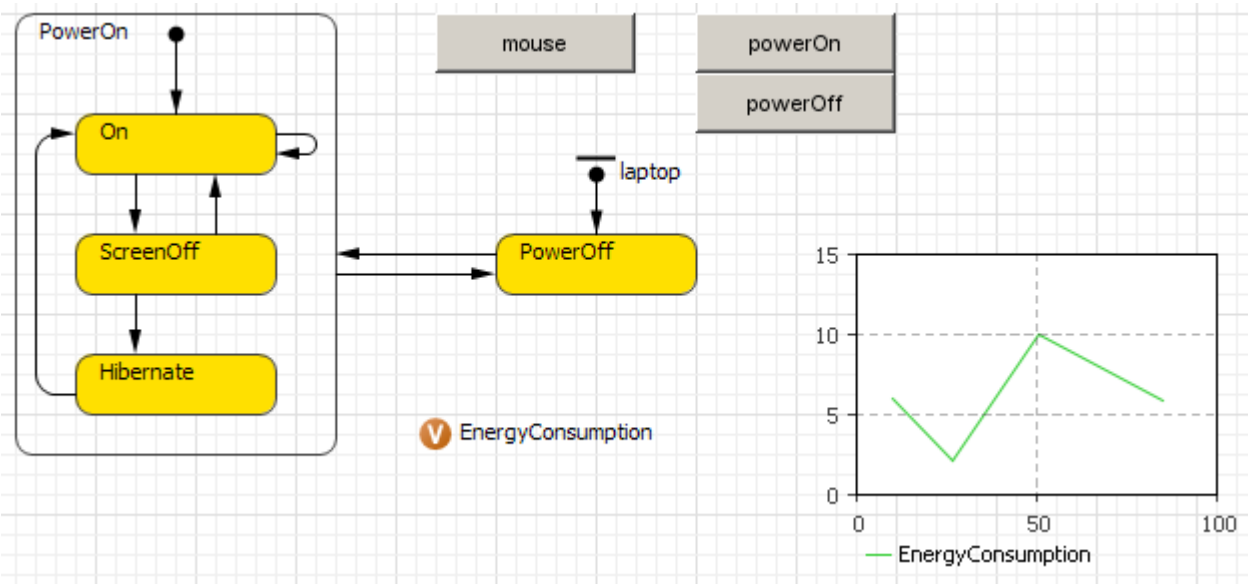
1 Representing Agent-Based Models

- State machine diagrams (copied from my G64OOS slides)
  - Helps us to understand what the **dependencies** are between **the state of an object** and **its reaction** to messages or other events
  - Show the states of a single object, the events or the messages that cause a transition from one state to another and the action that result from a state change

- State
  - A **condition during the life of an object** when it satisfies some condition, performs some action, or waits for an event

1 Representing Agent-Based Models

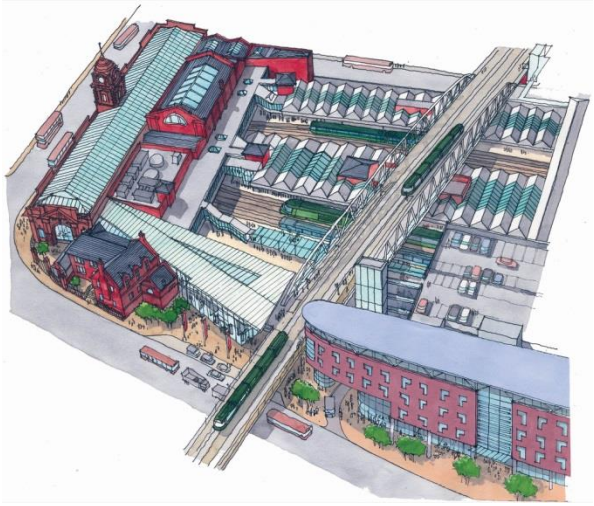
- Laptop example



2 Conceptual Modelling Exercise

- You have been hired by Nottingham County Council to help with the new design of the train station service centre (which is part of the main entrance hall). It is supposed to provide all the services required from "East Midlands Trains" prior to the train journey. These include providing information (direct and phone), selling tickets (direct and self-service) and collecting tickets (self-service).

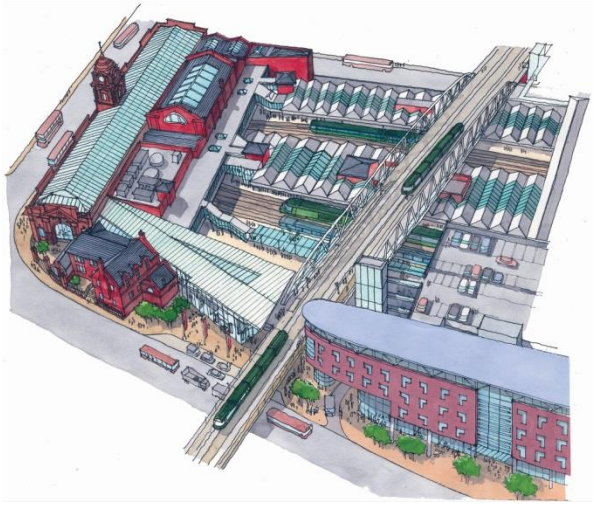
- Your task:
  - Develop a conceptual model that can help with the design using an agent-oriented approach



2 Conceptual Modelling Exercise

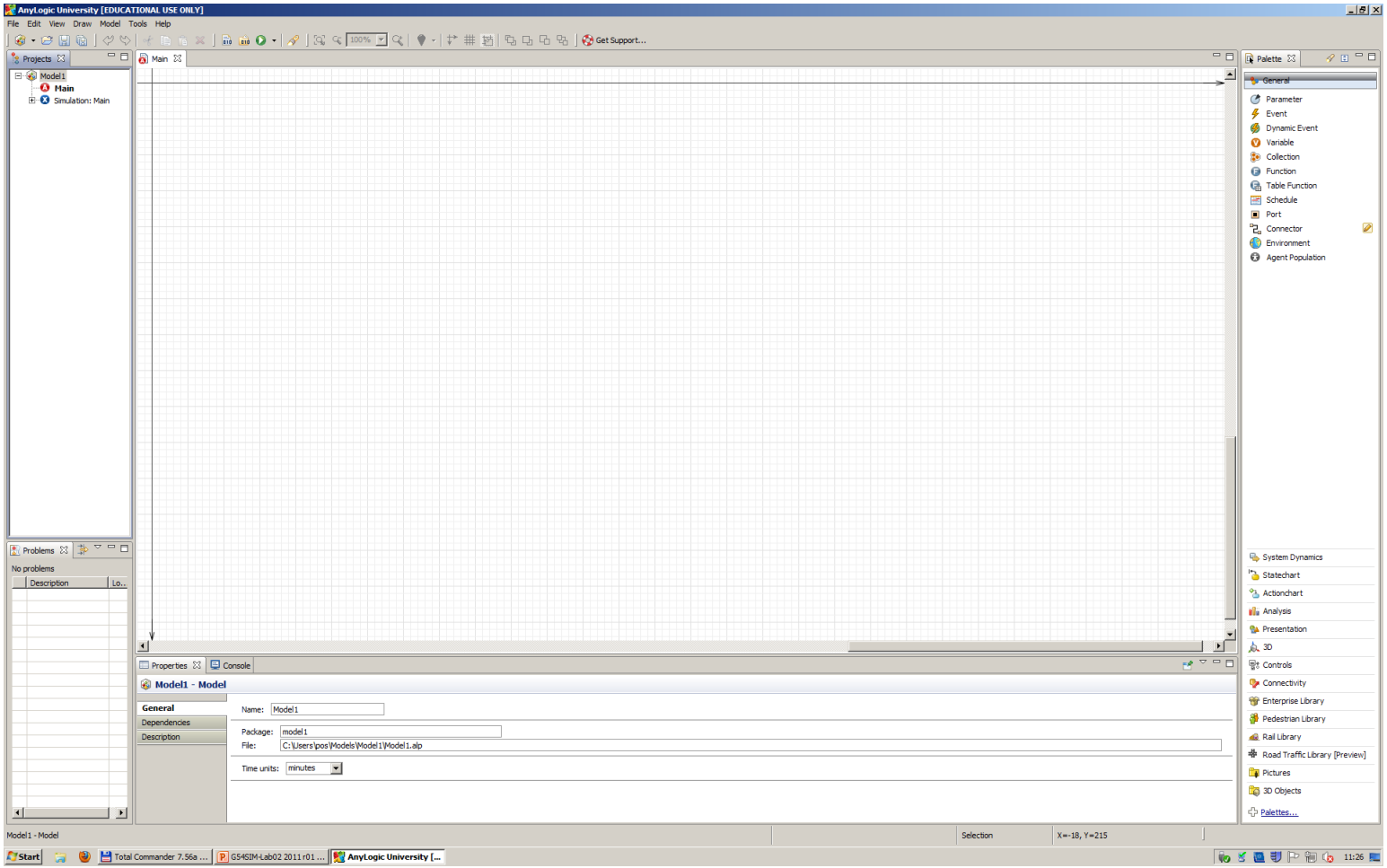
- I would propose to use a combined DES/ABS approach
  - First create a process flow chart for the system (you will later replace some of the passive entities by active ones
  - Then think about generic agents (active/passive entities)
  - Then think about what states these agents might be in
  - Then think about how they might transit from one state to another and how a transition would be triggered (timeout, rate, condition, ...)

- Make sure that you create state charts when you create your agents and not flow charts!



3

The AnyLogic IDE



3 The AnyLogic IDE

- Important Keys
  - F1: Help
  - Ctrl-Space: Code completion support
  - Ctrl-Enter: Perform refactoring (replace name occurrences)
  - Run: Select the correct model
  - Simulation: Set up simulation parameters

- AnyLogic Base Models
- AnyLogic Class Reference

The AnyLogic IDE

- Java Basics: AnyLogic requires some basic Java knowledge

Help - AnyLogic University

Search: Go Scope: All topics

Contents

- AnyLogic Help
  - Release Notes
  - Using Help
  - Basic Concepts
  - AnyLogic User Interface
  - Running the Model
  - Agent Based Modeling
  - System Dynamics Modeling
  - Parameters, Variables, Collections
  - Presentation and 3D Animation
  - Making Models Interactive Using Controls
  - Collecting and Visualizing Output Data
  - Scheduling Actions Using Events
  - Defining Behavior Using Statecharts
  - Defining Algorithms Using Functions and Action
  - Schedules and Table Functions
  - Java Basics for AnyLogic
    - Java in AnyLogic
    - Primitive data types
    - Classes
    - Variables (local variables and class fields)

AnyLogic Help > Java Basics for AnyLogic

Primitive data types

There are about ten *primitive data types* in Java, but in AnyLogic models we typically use these four:

Type name	Represents	Examples of constants
int	Integer numbers	12 10000 -15 0
double	Real numbers	877.13 12.0 12. 0.153 .153 -11.7 3.6e-5
boolean	Boolean values	true false
String	Text strings	"AnyLogic" "X = " "Line\nNew line" ""

The word "*double*" means real value with double precision. In AnyLogic engine all real values (such as time, coordinates, length, speed, random numbers) have double precision. The type `String` is actually a class (a non-primitive type, notice that its name starts with a capital letter), but it is a fundamental class, so some operations with strings are built into the core of Java language.

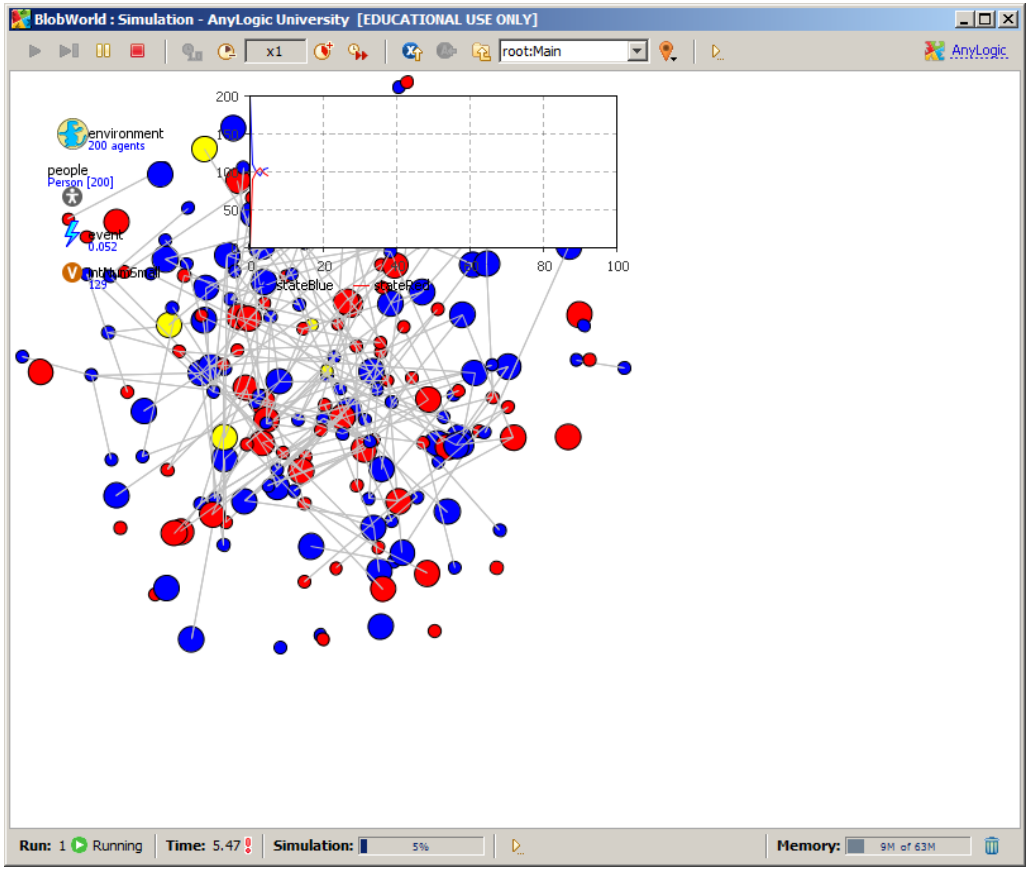
Consider the *numeric constants*. Depending on the way you write a number, Java will treat it either as real or as integer. Any number with the decimal delimiter "." is treated as a real number, even if its fractional part is missing or contains only zeros (this is important for integer division). If integer or fractional part is zero, it can be skipped, so ".153" is the same as "0.153", and "12." is the same as "12.0".

Boolean constants in Java are `true` and `false` and, unlike in languages like C or C++, they are not interchangeable with numbers, so you cannot treat false as 0, or non-zero number as `true`.

http://127.0.0.1:53741/help/topic/com.xj.anylogic.help/html/code/Java_Types.html

4 Agent-Based Simulation Tutorial

- Download Blob World tutorial [url]



Questions and Comments

- For more information about Systems Simulation have a look at the G54SIM module website: http://www.cs.nott.ac.uk/~pos/g54sim/