GeneSim

UML Modelling and Code Generation for Agent-based, Discrete Events Simulation

International Workshop
on Applied Modeling and Simulation
24-27 September 2012, Roma

Contents

- Quality goals for ADE Modelling and Simulation
- GeneSim approach
- Using UML, a brief and simple example
- Deriving C++ source code from the UML model
- Industrial case studies

Quality goals for ADE M&S

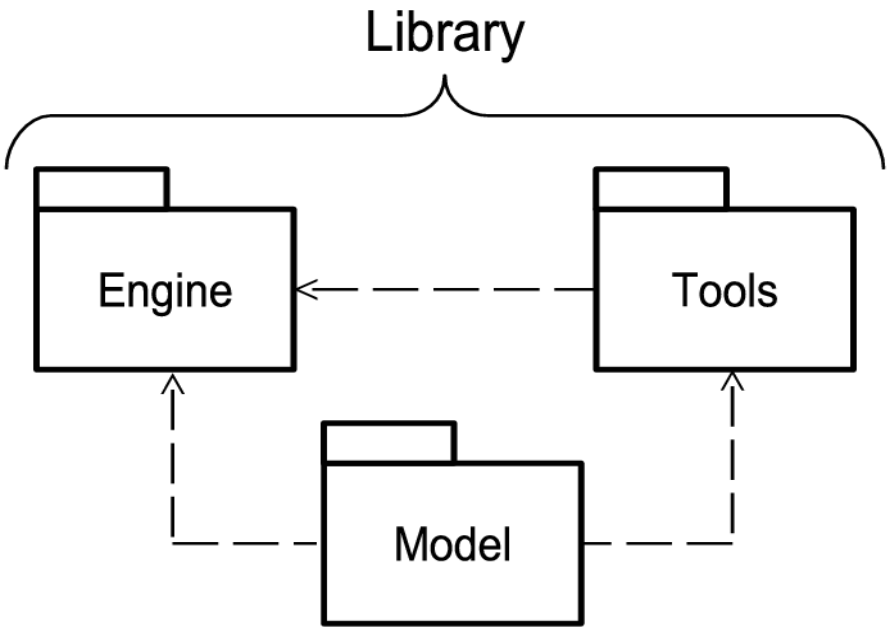
- Agent-based
  - Indentifiable, capable of making decisions
  - Characteristics, behaviour
- Discrete Events
  - Only relevant points in time
  - Classical modelling approaches (activity based, process based)
  - Each approach relies on its own simulator engine architecture
- Driving quality goals for model components
  - Re-usability of components in different system models
  - Traceability from specification to implementation
  - Independence of component behaviour

GeneSim approach

- Object orientation
  - Types and instances (classes and objects)
  - Inheritance and overloading
  - Exploited for re-usability and traceability
- Agent-based
  - Agent types (entities) and instances (agents)
  - Classes define the behaviour of all their instances
- Discrete Events
  - Event types and instances (events)
  - Defined behaviour: self dispatching events

GeneSim approach

- Engine
  - Time and agent states transitions
  - Provided by the framework
- Tools
  - Data structures (queues)
  - Random number generators
- Model
  - The custom part
  - Specifies the system, in UML
  - Generated and compiled with the library

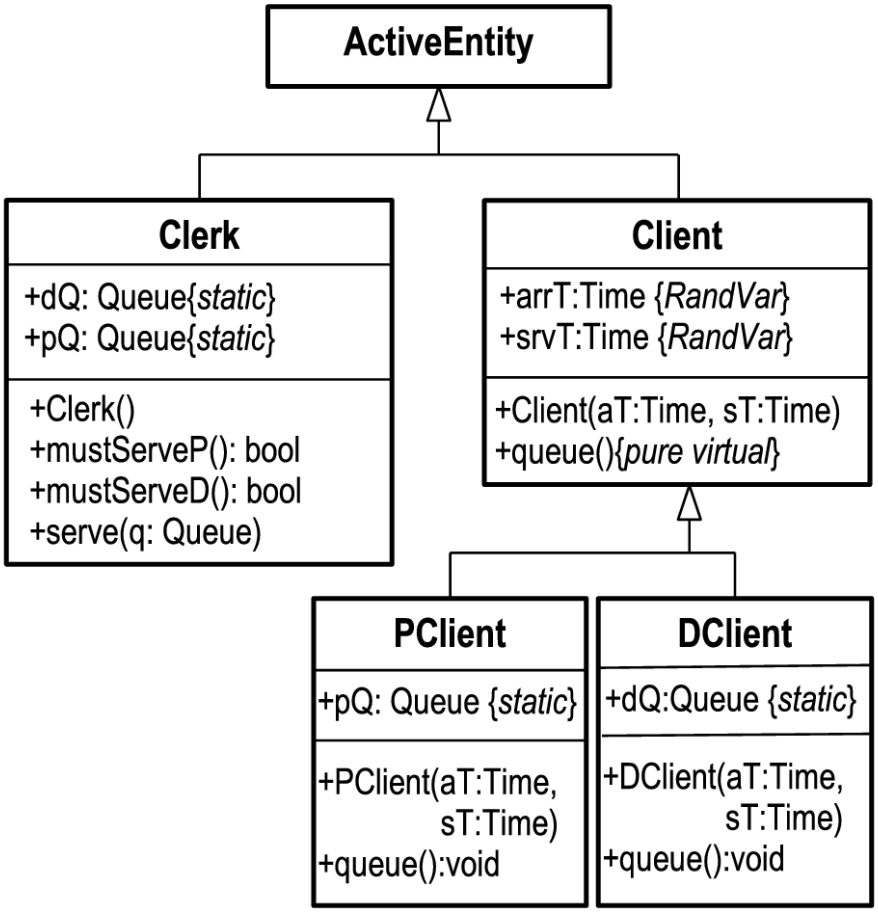


Using UML for simulation models

- Ways of using UML, in software engineering (and beyond)
  - Sketch
  - Blueprint
  - Program
- Blueprint level, forward engineering
  - To build a detailed design of the model before implementing it
  - Complete design but not enough for code generation
- Program level
  - Definitive model, still using UML syntax
  - Suitable for code generation
  - No manual coding needed (actually discouraged)

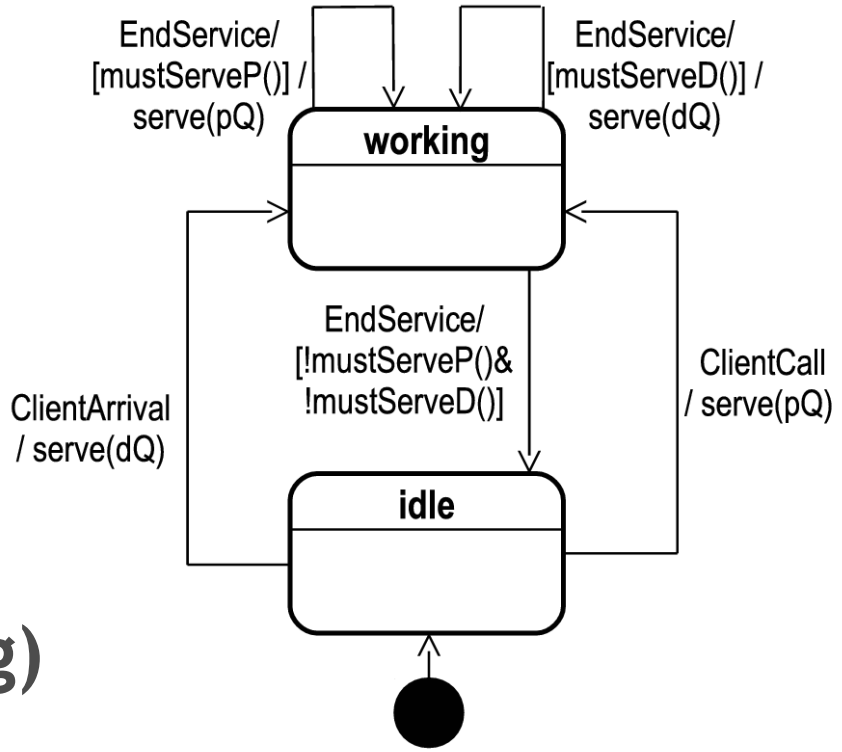
Structure (Agents)

- UML notation
  - Rules and conventions
  - Agents:
    class and state machine diagrams
  - Events:
    class and activity diagrams
  - Initial state: object diagram
  - Project: object diagram
- Agent Types Specifications
  - Attributes and methods
  - Characterizing the agents behaviour



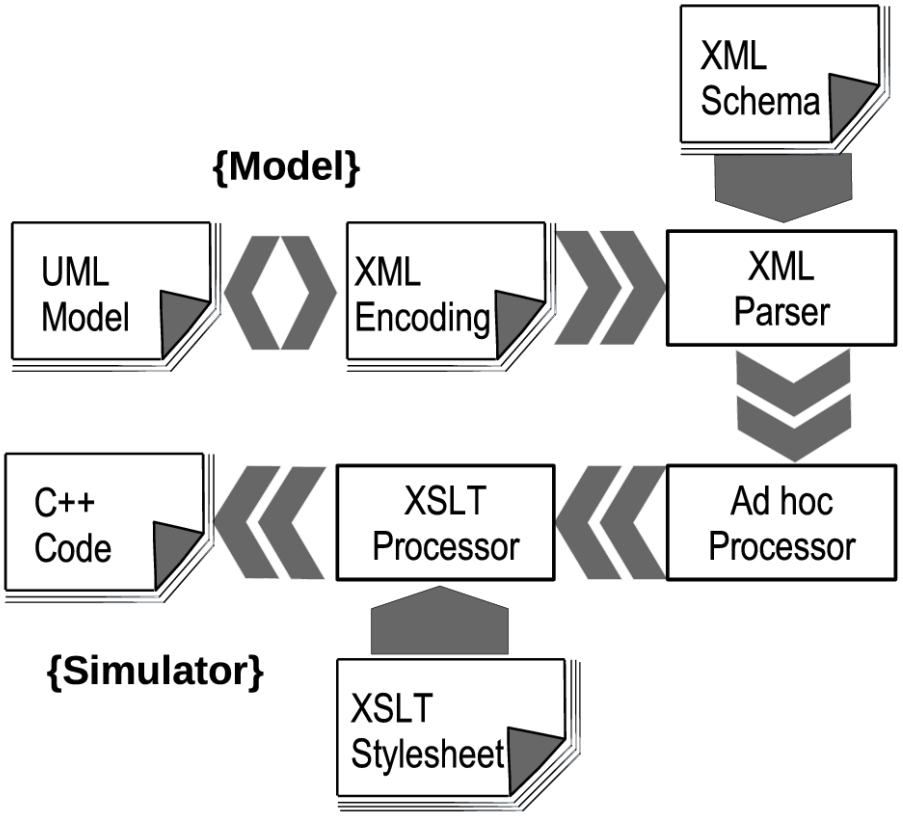
Behaviour (Agents wrt Events)

- Specifying how agents react to events
  - As a state machine
- Transitions
  - Event, mandatory
  - Guard, optional
    uses defined predicates
  - Action, optional
    uses defined methods
- Event behaviour (self dispatching)
  - Activity diagrams
  - Events wrt Agents



Code Generation

- UML to XML
  - XML encoding of UML model
  - XML suitable for code generation
- Actual code generation
  - Syntax validation
  - DOM processing
  - Semantic validation (well-formed)
  - XSLT transformations, based on predefined stylesheets
  - Result ready to be compiled
- Xerces and Xalan (Apache)



Industrial case studies

- The method and the tools were tested on the field
- A demand responsive public transport system
  - Validation of UML notation and code generation patterns
  - Province of Brescia and MAIOR srl
- Simulation of a public bus service
  - Targeted to validate the performance on a real size example
  - Network of 3071 vertices and 4300 links
  - Service schedule of 112 routes and 3029 daily courses
  - Simulation of a full day service required less than 1 sec on ordinary hardware
  - ATC Servizi spa (La Spezia) and MAIOR srl