A Java-Based Simulator for Agent-Object-Relationship Simulation

based on the results of my
PhD thesis at FU Berlin
Wolf-Ulrich Raffel
(uli@wuraffel.de)
Motivation

Simulation is easier than a reality experiment

Agent-based Simulation helps to structure simulation studies: encapsulation of system elements

For a domain expert, specifying visually is easier than programming.
Outline

• Introduction
  – Discrete-Event Simulation (DES)
  – Agent-Based Simulation (ABS)
  – Example application: Automatically Guided Vehicles (AGV)
• Agent-Based simulation model
  – Demands for a simulation specification language
  – Object-Based Simulation (OBS)
  – Agent-Based Simulation (ABS)
  – Reaction rules
• Visual specification of ABS (not focussed in this talk)
• Implementation of a simulator
  – Architecture of simulator
  – Demonstration
  – Simulation study AGVs
• Summary
Outline

• Introduction
  – Discrete-Event Simulation (DES)
  – Agent-Based Simulation (ABS)
  – Example application: Automatically Guided Vehicles (AGV)

• Agent-Based simulation model
• Visual specification of ABS
• Implementation of a simulator
• Summary
Discrete-Event Simulation (DES)

• System state changes uncontinuously when an event occurs
• Between two events, system state is constant
• Event set contains unprocessed events
• When processing an event, resulting events are generated

+ Popular, established
+ Clear, understandable structure
- Restricted expressiveness for complex systems
- No support for structuring systems
- No unique formalisation in the literature
Base model DES

- System state $S \in \text{State}$
- Simulation time $c$
- Event set $E \subseteq \text{Event}$
- Event function $f: (\text{State} \times \text{Event}) \rightarrow (\text{State} \times \emptyset(\text{Event}))$
- Discrete-Event Simulation in 2 lines:
  \[ \delta (S, c, E) = (S', t, (E \setminus \{e_{\text{min}}\}) \cup E'), \text{ where} \]
  \[ e_{\text{min}} \in E, t \text{ minimal and } (S', E') = f (S, e_{\text{min}}) \]
Agent-Based Simulation (ABS)

- Approach separated to DES
- Suitable for complex systems
- Active entities (agents)
  - interact with each other and with environment
  - do actions
  - percept and change their environment
  - have a mental state: knowledge, goals, memories, commitments, ...
Automatically Guided Vehicles (AGV)

- Tasks: awarding of transport orders, route scheduling, traffic regulation
- Decentrally controlled AGV systems are more robust than centrally controlled
- Perfect example for ABS
- Agent types:
  - TOM (Transport Order Manager)
  - Machine
  - AGV
  - TrafficController
Outline

• Introduction
• Agent-Based simulation model
  – Demands for a simulation specification language
  – Object-Based Simulation (OBS)
  – Agent-Based Simulation (ABS)
  – Reaction rules
• Visual specification of ABS
• Implementation of a simulator
• Summary
Demands for a simulation specification language

- Language and simulator must base on an agent-based meta model
  - Theoretical base for simulator

- Language must be declarative
  - Decoupling of specification and execution
  - Execution on different platforms
  - Natural and readable specification

- Language visual and UML-based
  - Easier to understand and to communicate
  - UML established as standard in science and industry
## State of the art

<table>
<thead>
<tr>
<th></th>
<th>Swarm</th>
<th>RePast</th>
<th>Spades</th>
<th>SeSAM</th>
<th>MadKit</th>
<th>COR-MAS</th>
<th>JADE</th>
<th>Robo-Cup</th>
<th>TAC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Meta Model</strong></td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Declarative</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>Visual</strong></td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><strong>UML-based</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

16.02.2006 W.-U. Raffel: A Java-Based Simulator for Agent-Object-Relationship Simulation
Object-Based Simulation (OBS)

- Extension of DES
- Splitting of system state into entities (object oriented system state)
- Distinction between exogenous events and resulting events
- State conditions and state effects
Object-Based Simulation (2)

- Event function $f$ realized by reaction rules (objects react on events)

<table>
<thead>
<tr>
<th>Trigger (=event)</th>
<th>Machine_pickup_TI (i) @ t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger Condition</td>
<td>-</td>
</tr>
<tr>
<td>Reactor</td>
<td>Machine [i]</td>
</tr>
<tr>
<td>State Condition</td>
<td>$in &gt; 0$</td>
</tr>
<tr>
<td>State Effect</td>
<td>$in = in@pre - 1$</td>
</tr>
<tr>
<td>Resulting Events</td>
<td>${ \text{ProductionFinished (i) @ t + Equal_RV (a, b)} }$</td>
</tr>
</tbody>
</table>
Agent-Based Simulation (ABS)

- Extension of OBS
- Distinction: agents/objects
  - objects: passive entities
  - agents: active entities
- Agents can send messages to each other
- Distinction between internal and external agent state
- Distinction between environmental simulator and agent simulators
Environmental simulator and agent simulators

- Environmental simulator ES
  - Environmental state S
    - A1, A2, A3, O1, O2
  - Event set E
    - Res, Exog

- Agent simulators
  - Agent simulator A1
    - Simulation time c
    - Internal state Int
      - ResTE, Perio
  - Agent simulator A2
  - Agent simulator A3
    - Time event set TE
      - A3
Progress of a cycle

Step 1: compute current events

Step 2: execute environmental function \( e \)

Step 3: send perceptions (incl. messages) to agent simulators

Step 4: compute current internal events and execute agent function \( a \)

Step 5: send actions (incl. messages) to environmental simulator
Step 1: compute current events

Current events:
- Resulting events in this cycle
- Exogenous events in this cycle
- Actions of agents from the end of last cycle
Step 2: environmental function $e$

$$e(S, E_{\text{cur}}) = (S', E_{\text{new}}, P1, P2, P3)$$

$$S \leftarrow S'$$

$$E \leftarrow E \setminus \text{Res} \cup E_{\text{new}}$$
Step 3: perceptions to agent simulators

The environmental simulator sends a message to each agent simulator, containing the perceptions of the simulated agent.
Step 4: agent step

Current internal events:

- Received perceptions
- Resulting time events in this cycle
- Periodical time events in this cycle

Agent function $a$:

$$a \left( \text{Int, TE}_{\text{cur}} \right) = \left( \text{Int}', \text{TE}_{\text{new}}, \text{Act} \right)$$
Step 5: actions from agent simulators

- The environmental simulator waits, until each agent simulator has sent the set of taken actions.
- It sets the time on cycle length forward.

Diagram:

- Environmental simulator ES
- Agent simulators A1, A2, A3
- Environmental state S: A1, A2, A3, O1, O2
- Event set E: Res, Exog
- Simulation time: \( c + \Delta c \)
### Reaction rules

- Distribution of functions $e$ and $a$ into reaction rules

#### An E-Rule:

<table>
<thead>
<tr>
<th>Event</th>
<th>AGV[i]:PutDownTI() @ t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Condition</td>
<td>-</td>
</tr>
<tr>
<td>State Condition</td>
<td>-</td>
</tr>
<tr>
<td>State Effect</td>
<td>context</td>
</tr>
<tr>
<td></td>
<td>a:AGV[i]</td>
</tr>
<tr>
<td></td>
<td>m:Machine[a.location.machine]</td>
</tr>
<tr>
<td></td>
<td>m.in=m.in@pre + 1 and</td>
</tr>
<tr>
<td></td>
<td>a.loadState=unloaded</td>
</tr>
<tr>
<td>Resulting Events</td>
<td>Ø</td>
</tr>
<tr>
<td>Perceptions</td>
<td>{Machine</td>
</tr>
<tr>
<td></td>
<td>[AGV[i].location.machine]:</td>
</tr>
<tr>
<td></td>
<td>{TIArrives()})</td>
</tr>
</tbody>
</table>

#### An A-Rule for Machine:

<table>
<thead>
<tr>
<th>Event</th>
<th>TIArrives()</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Condition</td>
<td>-</td>
</tr>
<tr>
<td>State Condition</td>
<td>prodState=idle</td>
</tr>
<tr>
<td>State Effect</td>
<td>orderedTI =</td>
</tr>
<tr>
<td></td>
<td>orderedTI@pre-1 and</td>
</tr>
<tr>
<td></td>
<td>prodState=producing</td>
</tr>
<tr>
<td>Resulting Time Events</td>
<td>{checkPO()@ c}</td>
</tr>
<tr>
<td>Messages</td>
<td>Ø</td>
</tr>
<tr>
<td>Actions</td>
<td>{ pickUpTI( ) }</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Agent-Based simulation model
- Visual specification of ABS (not focussed in this talk)
- Implementation of a simulator
- Summary
Visual specification of ABS

• Visual models are easier to understand and to communicate
• UML models used, because UML is established as standard in science and industry
• In theory: each UML tool is suitable
• In practice: only ArgoUML can be used without problems
Outline

• Introduction
• Agent-Based simulation model
• Visual specification of ABS
• Implementation of a simulator
  – Architecture of simulator
  – Demonstration
  – Simulation study AGVs
• Summary
Implementation of a simulator

- With the aid of Duc Minh Nguyen (diploma thesis)
- UML tool exports model as XMI
- XMI is transformed via XSLT to ABSimML (all relevant information for ABS)
- ABSimML can directly used as specification language as well
- A centralistic Java program reads ABSimML and generates code
- Automatical execution of generated code

UML

ABSimML

Java

16.02.2006

W.-U. Raffel: A Java-Based Simulator for Agent-Object-Relationship Simulation
Simulator kernel

- Centralistic Java program
- 1100 lines of code, Jar file 34 KB
- Set of abstract Java classes
- User (e.g. code generator) uses inheritance to specify concrete instances
- Main program connects environmental simulator and agent simulators

=> Extension for threading and distribution to several computers easy to realize
UML diagram environmental simulator

EnvEvent
- time
- rules
- simulate()

EnvRule
- triggeredByEventType(): boolean
- triggerCondition(): boolean
- stateCondition(): boolean
- stateEffect(): void
- resultingEvents(): EventSet
- perceptions(): PerceptionSet

ExogenousEvent
- stopCondition(): boolean
- timeToNextEvent(): double

ExogenousEvent
- exogenous_events

ResultingEvent
- resulting_events

Message
- future_messages

MessageEvent
- resulting_events

MessageRule

MessageEvent

ActionEvent
- stopCondition(): boolean
- timeToNextEvent(): double

Action

EnvSimulator
- DELTA_C: double
- clock: double
- envSimulatorStep() currentEvents()

SimAgent
- id: int

SimObject
- id: int

16.02.2006 W.-U. Raffel: A Java-Based Simulator for Agent-Object-Relationship Simulation
Transformations

• Transformation XMI -> ABSimML
  – Extraction of relevant information
  – XSLT Processor Xalan used

• Transformation ABSimML -> Java code
  – Adaptation to Java syntax and simulator classes
  – XML parser Xerces used
  – Parser for OCL subset generated using JavaCC

• Transformation Java code -> Java classes
  – Direct call of javac in a Java program (com.sun.tools.javac)

• Transformation Java classes -> running program
  – Direct call of java in a Java program (Runtime.exec())
Demonstration

• Transformation of a UML simulation model into a running simulation

• Demonstration example: Automatic Door
  – 1 motion sensor
  – 1 door
  – Sensor notifies door about movement, door opens and closes again after a period of time
Simulation study AGV

- Production hall 250x250m
- 18 Machines
- 5 working steps
- 36 nodes
- 120 Segments
- 1-20 AGVs
- 24 hours
Results simulation study

<table>
<thead>
<tr>
<th>Number of TOMs (transport order managers)</th>
<th>produced transport items without exchange (6 AGVs)</th>
<th>produced transport items with exchange (6 AGVs)</th>
<th>produced transport items without exchange (5 AGVs)</th>
<th>produced transport items with exchange (5 AGVs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>672,1</td>
<td>690,0</td>
<td>541,7</td>
<td>574,4</td>
</tr>
<tr>
<td>2</td>
<td>669,0</td>
<td>690,0</td>
<td>542,3</td>
<td>574,6</td>
</tr>
<tr>
<td>4</td>
<td>669,1</td>
<td>690,5</td>
<td>542,0</td>
<td>578,1</td>
</tr>
</tbody>
</table>

number of produced transport items with and without exchange of transport orders

16.02.2006  W.-U. Raffel: A Java-Based Simulator for Agent-Object-Relationship Simulation
Outline

• Introduction
• Agent-Based simulation model
• Visual specification of ABS (not focussed in this talk)
• Implementation of a simulator
• Summary
Summary

• Agent-Based simulation model
  – Formal execution semantics
  – Distinction between internal and external agent state
  – Environmental simulator and agent simulators
  – Based on Discrete-Event simulation

• Specification language
  – Based on simulation model
  – Visual language: UML-based
  – XML-based specification language: ABSimML

• Implementation
  – Automatical transformation and execution
    • of visually specified simulation models
    • of simulation models specified via ABSimML
  – Extensive simulation study of AGVs
Thank you!

Questions?