A Case Study of the Development of an Agent-based Simulation in the Traffic Signal Control Domain using an MDD Approach

Fernando Santos¹,², Ingrid Nunes¹,³, Ana L.C. Bazzan¹

¹ Instituto de Informática, UFRGS, Brazil
² Depto. de Engenharia de Software, UDESC, Brazil
³ TU Dortmund, Germany

{fsantos, ingridnunes, bazzan}@inf.ufrgs.br

May 8, 2017
Agent Based Modeling and Simulation (ABMS)

- Paradigm that uses **agent-based simulations** to [re]produce, analyze, or predict a phenomenon under study
- Particularly suitable for **complex systems** and **emergent phenomena**

**Ant foraging**

**Traffic**
Agent-based Simulation Development

- Agent-based Simulation (ABS) Platforms

- Demand for technical expertise
- No **building blocks** for recurrent concepts and behaviors

**MASON**

**SWARM**

**GAMA PLATFORM**

**SeSAm**

*Shell for Simulated Multi-Agent Systems*

v1.3 Jan2017
Model-driven Development (MDD)

• Models as *first-class citizens*

• Trades *generality* for *expressiveness*
  – Domain concepts are available for modeling
  – Reduces the abstraction gap and increases productivity

• MDD in ABMS: **Limitations**
  – Limited to particular MDD aspects
    • Metamodels: AMASON, MAIA, metamodel of Ribino *et al.* (2014)
    • Methods for identifying certain domain concepts: easyAMBS, IODA
  – Much is left to be **developed manually**
    • Often, only code skeletons are produced automatically
  – **Lack of evidence of the real benefits** that an MDD approach can promote for developing agent-based simulations.
Research Question

What is the benefit an MDD approach can provide for developing agent-based simulations?

• We explore this question through a case study
  – Provision of an MDD approach for ABMS
  – Empirical assessment of its gains
    • Development effort
MDD for ABMS: preliminaries

• The more specific the application domain, the higher the chance of success

• Selected domain
  – *Agent-based simulations* of Adaptive Traffic Signal Control (ATSC)
Adaptive Traffic Signal Control

ATSC

• Traffic Signal Control (TSC) agents in charge of managing traffic light indicators so as to:
  – Maximize traffic flow
  – Minimize travel time
  – Other metrics

• Distributed and autonomous agents

• Availability of decision-making techniques
ATSC elementary concepts

- TSC agents
- Incoming/outgoing lanes
- Traffic light indicators
- Stages
- Phases
- Plans
- Cycle

(a) Basic Elements.

(b) Stages.

(c) Phase, Plan and Cycle.
MDD for ABMS: elements

1. Metamodel
   – Domain concepts → meta-entities and their relationship
   – Built through a domain analysis activity

2. Domain-specific language (DSL)
   – Building blocks for recurrent concepts → expressiveness
   – Allow modeling in an expressive way (abstraction gap)

3. Transformations & Code Generation
   – Rules for producing code automatically → productivity
Domain Analysis Method

• Bottom-up, based on existing simulations
  1. Build a preliminary list of agent-related concepts following the steps of existing methodologies for ABMS
  2. Refine the identified concepts using the ODD protocol
     • Adaptation, learning, collectives, ...
  3. Find the essence behind each identified concept
     • Recurrent characteristics and behaviors
  4. Build the domain model
Step 3

• How domain concepts were abstracted:

Reasoning of TSC Agents:
Learn which stage to activate

Uses Reinforcement Learning.
Technique: Q-Learning
States: active stage x queue length
Actions: existing stages
Reward: queue length of incoming lanes

Q-Learning algorithm overview:
At each timestep
  Update Q-table
  Select an action according to
  a selection policy
Metamodel

Step 4

- Also, acts as the DSL abstract syntax
Metamodel

Step 4

Entities & Agents

MMEntity
- name : String
- description : String
- pluralName : String

MMAgent

MMDecisionCapability
- timerSelectedOption : Double = 0.1
- decide() : MMDecisionOption

MMFlowControlCapability

MMStateMachine

MMAgentCapability

MMAAttribute
- name : String
- description : String
- value : AnySimpleType
- cardinality : Int

[0..*] attributes

[1..1] activation

MMDecisionOption

[0..*] options

MMActorState
- name : String
- isDefault : Boolean = false

[1..1] states

[1..*] activations

[1..*] regulators

MMActuator

[1..*] actuators

MMActuatorGroup

MMLearning

MMAdaptation

Flow Control
Metamodel

Step 4

Entities & Agents

MMEntity
- name : String
- description : String
- pluralName : String

MMAgent

MMAgentCapability

MMDecisionCapability
- timerSelectedOption : Double = 0.1
- decide() : MMDecisionOption

MMStateMachine

MMAgent

MMAAttribute
- name : String
- description : String
- value : AnySimpleType
- cardinality : Int

MMActuator
- id : Int

MMActuatorGroup
- name : String

MMActuatorState
- name : String
- isDefault : Boolean = false

MMCFlowControlCapability

[0..*] activations

[1..*] states

[0..*] selectedOption

[0..*] regulators

[1..*] activations

[1..1] actuators

[1..1] actuatable

[1..1] complexType

[0..*] attributes

[1..1] activation

[0..*] capabilities

[0..1] decisionCapability

Decision

Flow Control
DSL4ABMS: Modeling Language

Concrete Syntax

- UML-inspired building blocks for ABMS elements

---

Traffic Signal Controller

Flow Control

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Designer defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>9</td>
</tr>
<tr>
<td>Location</td>
<td>TrafficNode</td>
</tr>
</tbody>
</table>

Name | Init. | Update
--- | --- | ---

| streams [2] | (Expr) in links |
| activation | (Decision Capability) plan learning |

queue stage1: \( | v \text{ in Vehicles : location=streams [0]} | v \text{ in Vehicles : location=streams [1]} | \text{avg queue length} = \text{avg (queue stage1, queue stage2)} \)

---

Learning plan learning

State Def.

| (Expr) queue stage 1 |
| (Expr) queue stage 2 |
| (Expr) timerSelectedOption |

Reward

| (Expr) avg queue length - avg queue length \{-1\} |

Learning Parameters

| Technique | Q-Learning |
| Learning rate | 0.08 |
| Discount factor | 0.8 |
| Selection policy | Epsilon greedy |
| Epsilon | 0.9 |

---

State Machine plan north-south

<table>
<thead>
<tr>
<th>Option</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>phase 1</td>
<td>42 sec</td>
</tr>
<tr>
<td>phase 2</td>
<td>18 sec</td>
</tr>
</tbody>
</table>

---

State Machine plan west-east

<table>
<thead>
<tr>
<th>Option</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>phase 3</td>
<td>18 sec</td>
</tr>
<tr>
<td>phase 4</td>
<td>42 sec</td>
</tr>
</tbody>
</table>

---

Actuator States

- green
- yellow
- red

---

Actuator Groups

- stage 1 [0]
- stage 2 [1]

---

Actuator(s)

- [0 .. 3]
Model-to-code Transformations

- Production rules to generate NetLogo code. E.g.:

<table>
<thead>
<tr>
<th>Production Rule</th>
<th>Transformations to NetLogo Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent type</td>
<td>For each MMAgent $\rightarrow$ <strong>breed</strong></td>
</tr>
<tr>
<td>RL capability</td>
<td>For each MMReinforcementLearning $\rightarrow$</td>
</tr>
<tr>
<td></td>
<td>- Qlearning init.</td>
</tr>
<tr>
<td></td>
<td>- Qlearning reward def.</td>
</tr>
<tr>
<td></td>
<td>- Qlearning update Qtable</td>
</tr>
<tr>
<td></td>
<td>- Qlearning decision</td>
</tr>
<tr>
<td>Qlearning update QTable</td>
<td>executes Qlearning reward def. <strong>reporter</strong> to compute reward</td>
</tr>
<tr>
<td></td>
<td><strong>set</strong> statement for updating the Qtable</td>
</tr>
<tr>
<td></td>
<td>$Q(s, a) = Q(s, a)$</td>
</tr>
<tr>
<td></td>
<td>$+ \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$</td>
</tr>
</tbody>
</table>

- Specified and implemented using XPand
Evaluation: Questions

• Q1. MDD Effectiveness
  Is our MDD approach able to produce ready-to-run code from DSL4ABMS models?

• Q2. MDD Benefits
  Does our MDD approach decrease the effort to develop agent-based simulations?
Evaluation: Selected Simulations

• One simulation was selected for each decision capability

• Fixed plans → State machines
    http://ccl.northwestern.edu/netlogo/models/TrafﬁcGrid.

• Self-organizing Traffic Lights → Adaptation

• Signal Plan Learning → Reinforcement Learning
Q1. Generated Simulations

- TSC-related ready-to-run code was fully automatically generated.
Q2. Development Effort

• **Size metrics**, from cost estimation methods in SE

• **Lines of Code (LoCs)** is a key size measurement
  – Used in methods such as Function Points and COCOMO
  – Can be manually produced (MLoCs) or generated (GLoCs)

• **Atomic Model Element (AMEs)**
  – Used to measure graphical models
  – 1 AME is a visual element that is equivalent to 1 LoC
  – Counting rules are detailed in the paper
Q2. Results

<table>
<thead>
<tr>
<th>Simulation</th>
<th>AMEs</th>
<th>MLoCs</th>
<th>Effort*</th>
<th>GLoCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed Plan / State Machine</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetLogo</td>
<td>1</td>
<td>34</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Our MDE approach</td>
<td>14</td>
<td>0</td>
<td>14</td>
<td>116</td>
</tr>
<tr>
<td><strong>Total Effort Reduction</strong></td>
<td></td>
<td></td>
<td><strong>60.0%</strong></td>
<td></td>
</tr>
<tr>
<td>Self-organizing Traffic Lights / Adaptation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetLogo</td>
<td>1</td>
<td>82</td>
<td>83</td>
<td>1</td>
</tr>
<tr>
<td>Our MDE approach</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>103</td>
</tr>
<tr>
<td><strong>Total Effort Reduction</strong></td>
<td></td>
<td></td>
<td><strong>81.9%</strong></td>
<td></td>
</tr>
<tr>
<td>Signal Plan Learning / Reinforcement Learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITSUMO</td>
<td>0</td>
<td>257</td>
<td>257</td>
<td>0</td>
</tr>
<tr>
<td>Our MDE approach</td>
<td>37</td>
<td>0</td>
<td>37</td>
<td>297</td>
</tr>
<tr>
<td><strong>Total Effort Reduction</strong></td>
<td></td>
<td></td>
<td><strong>85.6%</strong></td>
<td></td>
</tr>
</tbody>
</table>

*Effort = AMEs + MLoCs

Our MDE approach reduces the effort
Q2. Results

<table>
<thead>
<tr>
<th>Simulation</th>
<th>AMEs</th>
<th>MLoCs</th>
<th>Effort*</th>
<th>GLoCs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Fixed Plan / State Machine</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetLogo</td>
<td>1</td>
<td>34</td>
<td>35</td>
<td>1</td>
</tr>
<tr>
<td>Our MDE approach</td>
<td>14</td>
<td>0</td>
<td>14</td>
<td>116</td>
</tr>
<tr>
<td><strong>Self-organizing Traffic Lights / Adaptation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NetLogo</td>
<td>1</td>
<td>82</td>
<td>83</td>
<td>1</td>
</tr>
<tr>
<td>Our MDE approach</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>103</td>
</tr>
<tr>
<td><strong>Signal Plan Learning / Reinforcement Learning</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITSUMO</td>
<td>0</td>
<td>257</td>
<td>257</td>
<td>0</td>
</tr>
<tr>
<td>Our MDE approach</td>
<td>37</td>
<td>0</td>
<td>37</td>
<td>297</td>
</tr>
</tbody>
</table>

*Effort = AMEs + MLoCs

- Why does our MDD approach produce simulations with more LoCs than NetLogo/ITSUMO?
  - It produces code for **reusable** domain-independent abstractions (e.g., state machines)
  - But no **human effort** is required to produce it
Conclusion

- Our MDD approach reduces the effort to develop agent-based simulations in the ATSC domain
- The domain analysis method is effective

“most efforts are at lower levels of solution maturity”

Future Work

• Experiment with humans to evaluate subjective aspects of our MDD approach
  – E.g., usability, comprehensibility

• Incorporate additional simulation concepts

• Long-term goal: to use MDD to ease the development of agent-based simulations
Invitation: Demo Sessions

• Thursday, May 11
  – 10:20h - 11:20h
  – 16:10h - 17:10h
A Case Study of the Development of an Agent-based Simulation in the Traffic Signal Control Domain using an MDD Approach

Fernando Santos\textsuperscript{1,2}, Ingrid Nunes\textsuperscript{1,3}, Ana L.C. Bazzan\textsuperscript{1}

\textsuperscript{1} Instituto de Informática, UFRGS, Brazil
\textsuperscript{2} Depto. de Engenharia de Sofware, UDESC, Brazil
\textsuperscript{3} TU Dortmund, Germany

{fsantos, ingridnunes, bazzan}@inf.ufrgs.br

May 8, 2017