




Agents and Model-Driven Architecture: First Steps

Kuldar Taveter

The University of Melbourne



Talk outline

- 1 Model-Driven Architecture overview.
- 1 Agent paradigm and Model-Driven Architecture.
- 1 What kinds of agents does the industry currently need?
- 1 Generating agents from business models.
- 1 Summary.

The OMG Model Driven Architecture (MDA)

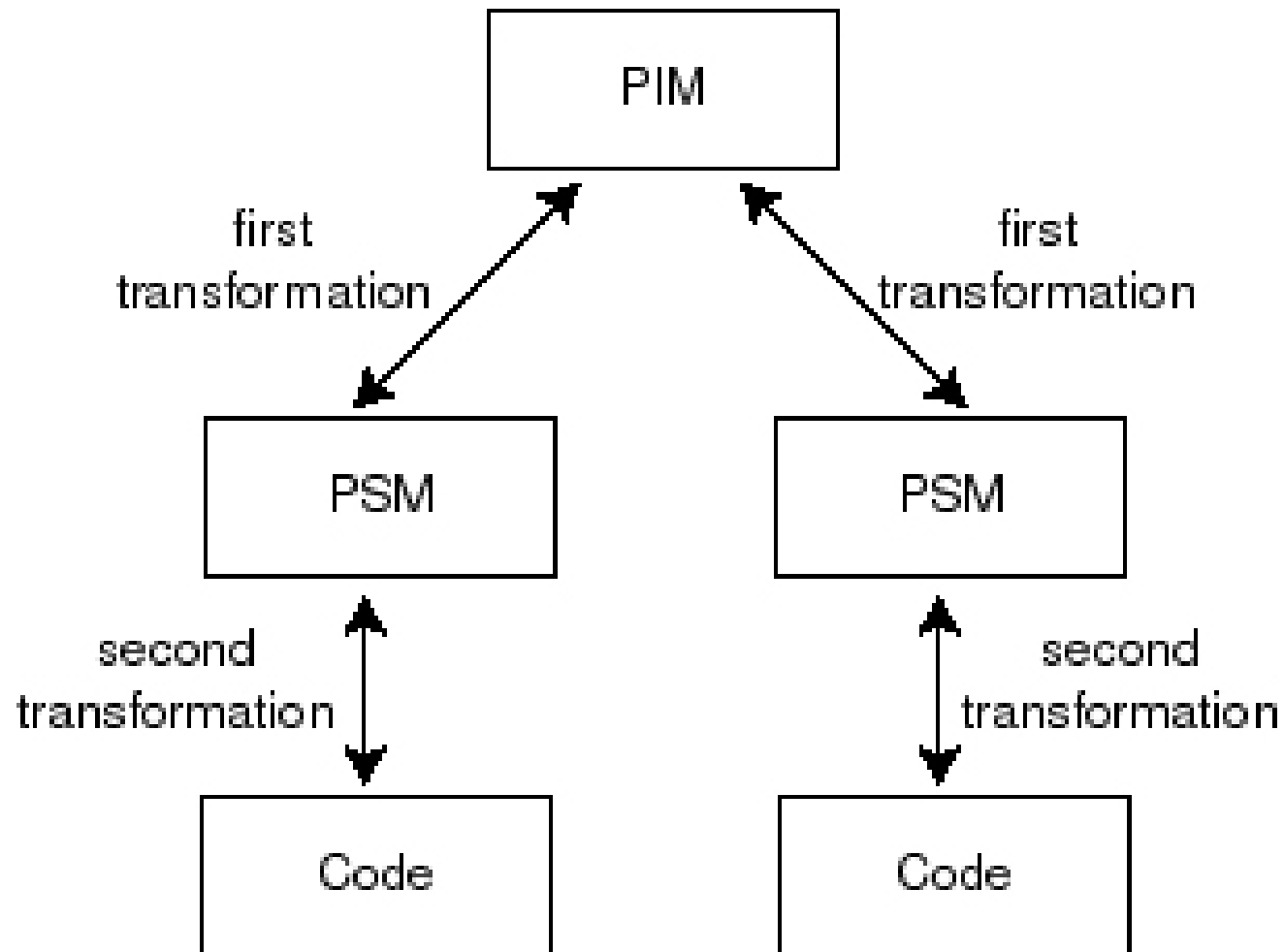
Step	Result
Conceptual Domain Modelling	Computation Independent Models (CIM)
Platform-Independent Computational Design	Platform Independent Models (PIM)
Platform-Specific Implementation	Platform Specific Models (PSM) and Code



Logical levels according to Zachman

- 1 Business (domain).
- 1 Information system.
- 1 Technology and implementation.

The MDA process



Agents and MDA



1 Agent as a:

- modelling abstraction (actor);
- computational abstraction;
- implementation unit.

1 Agent paradigm spans all three levels of MDA.

The RAP Viewpoint Modelling Framework

<i>Viewpoint models</i>	<i>Viewpoint aspect</i>		
<i>Abstraction level</i>	<i>Interaction</i> (org. + interactional)	<i>Information</i> (inform.)	<i>Behaviour</i> (motiv. + func. + behav.)
<i>Conceptual Domain Modelling</i>			
<i>Platform-Independent Computational Design</i>			
<i>Platform-Specific Design and Implementation</i>			

The RAP Viewpoint Modelling Framework populated by models of ROADMAP/Prometheus

<i>Viewpoint models</i>	<i>Viewpoint aspect</i>		
<i>Abstraction level</i>	<i>Interaction</i> (org. + interactional)	<i>Information</i> (inform.)	<i>Behaviour</i> (motiv. + func. + behav.)
<i>Conceptual Domain Modelling</i>	Role Model, Social Model, Models of Actions and Percepts	Environment Model, Knowledge Model	Goal Model
<i>Platform-Independent Computational Design</i>	Agent Descriptors, Agent Acquaintance Diagrams, System Overview Diagram , interaction Protocols	Data/Knowledge Diagrams, Belief Models	Service Models, Agent Lifecycle Models, Agent Overview Diagrams, Process Specifications
<i>Platform-Specific Design and Implementation</i>	Event Descriptions	Knowledge and Data Descriptions	Capability Descriptors, Plan Event Diagrams

The RAP Viewpoint Modelling Framework populated by models of AOR and UML

<i>Viewpoint models</i>	<i>Viewpoint aspect</i>		
Abstraction level	<i>Interaction</i> (org. + interactional)	<i>Information</i> (inform.)	<i>Behaviour</i> (motiv. + func. + behav.)
<i>Conceptual Domain Modelling</i>	AOR Actor Diagrams, UML Use Case Diagrams, AOR Interaction Frame Diagrams, AOR Interaction Sequence Diagrams	AOR Actor Diagrams	Goal Models , AOR Interaction Pattern Diagrams, Goal-Based Use Case Models, AOR Activity Diagrams, Object Constraint Language
<i>Platform-Independent Computational Design</i>	UML Use Case Diagrams, AOR Reaction Frame Diagrams, User Interface Design Models, Security Models	AOR Actor Diagrams	AOR Reaction Pattern Diagrams, AOR Internal Activity Diagrams, Object Constraint Language
<i>Platform-Specific Design and Implementation</i>	UML Deployment Diagrams, UML Class Diagrams, UML Interaction Diagrams	UML Class Diagrams	UML Class Diagrams, UML State Machine Diagrams

External AOR Diagram

Internal AOR Diagram

Comparison with other frameworks

<i>Abstraction level</i>	<i>Audience/Stakeholders</i>	<i>Viewpoint Names</i>		
		<i>MDA</i>	<i>RM-ODP</i>	<i>Zachman</i>
<i>Conceptual Domain Modelling</i>	owners/customers, users, domain experts	CIM	Enterprise	Rows 1+2
<i>Platform-Independent Computational Design</i>	systems analysts, software architects	PIM	Information + Computational	Row 3
<i>Platform-Specific Computational Design and Implementation</i>	programmers, database implementers, system integrators	PSM	Engineering + Technology	Rows 4+5

What kinds of agents does the industry currently need?

- 1 Agent as an abstraction.
- 1 More and simple rather than less and complicated.
- 1 Reactive rather than proactive agents.
- 1 Situated and attached to human actors.
- 1 Automating routine activities.
- 1 Working in “soft” real time (there are almost no enterprises working in real time yet!).
- 1 Reconciling ontological differences.

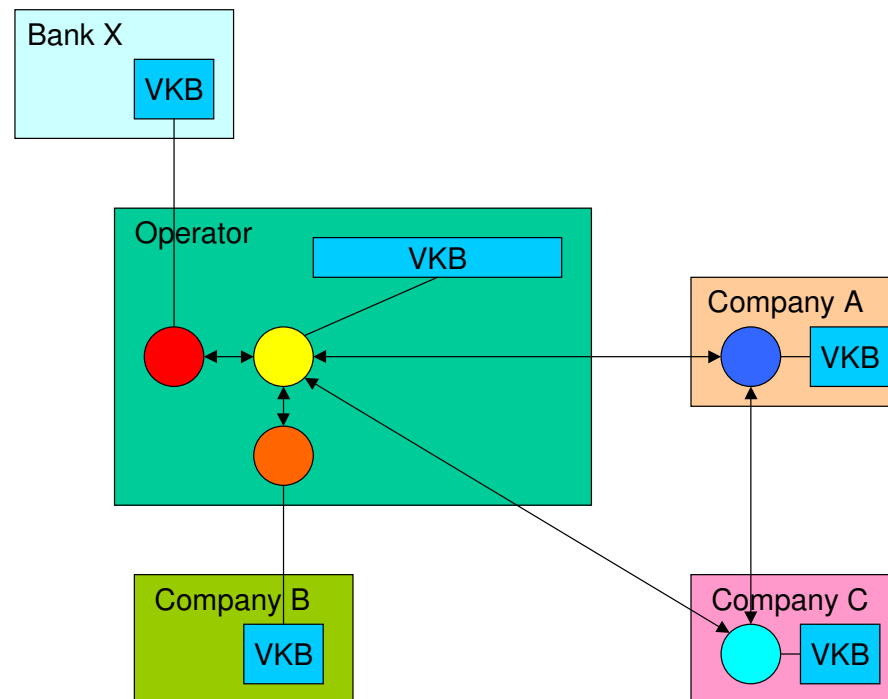


Ambient Intelligence

- 1 Ambient Intelligence (IST Advisory group, 2005): “... an environment of potentially thousands of embedded and mobile devices (or software components) interacting to support user centred goals and activity ...”.

Example: agent-based business process automation

- 1 Plug-and-Trade B2B: a Finnish domestic project performed at VTT Information Technology in 2003 - 2004 (<http://www.vtt.fi/tte/agents>).
- 1 Companies participating in a business process (e.g., quoting) are represented by agents. Agents of some companies are “outsourced” to the operator. Circle: agent; arrow: communication; VKB: agent’s knowledge base which includes interfaces to the internal information systems of the company:



Buyer Agent

The screenshot displays the 'Buyer GUI' application window. The main window has a title bar 'Buyer GUI' and a menu bar 'File Help'. The main content area is titled 'Buyer GUI Plug And Trade' and is divided into several sections:

- active RFQs:** A table listing active Request for Quote (RFQ) items. The table has columns for product name, product code, valid until, and number of responses.
- product description:** A text area showing the description for the selected RFQ: 'KYTKENTÄJOHTO'.
- valid from -> until:** A text field showing the validity period: '31.03.2004 00:00'.
- ordered amount:** A text field showing the amount: '34.0'.
- ordered delivery date:** A text field showing the delivery date: '31.03.2004 00:00'.
- issued by (contact info):** A text field showing the contact name: 'Maija Meikäläinen'.
- 2 responses received and ranked:** A table showing the ranked responses for the selected RFQ.

Buttons for 'refresh' and 'details' are located below the active RFQs table. A 'request a new quote' section contains a 'new RFQ' button. The Windows taskbar at the bottom shows the Start button, several Command Prompt windows, and the Buyer GUI window.

product name	product code	valid until	number of responses
MKEM 35 PUNANEN JOHDI...	4826	30.03.2004 00:00	1
MKEM 1,5 HARMAA JOHDIN...	4701	30.03.2004 00:00	1
MKEM 35 PUNANEN JOHDI...	4826	30.03.2004 00:00	0
MKEM 6 PUN 90AST JOHDI...	4766		
MKEM 35 PUNANEN JOHDI...	4826		
MKEM 2,5 HARMAA JOHDIN...	4721		
MKEM 35 PUNANEN JOHDI...	4826		
MKEM 2,5 HARMAA JOHDIN...	4721		
MKEM 2,5 HARMAA JOHDIN...	4721		
SARJA UUSIA ERIK.TYÄKA...	5000		
MKEM 35 PUNANEN JOHDI...	4826		
SARJA UUSIA ERIK.TYÄKA...	5000		

bidder name	bidder credibi...	product name	amount	price	delivery date	substitution r...	status
IEKY	0.39	MKEM 2,5 HA...	34.0	0.5	31.03.2004 0...		ISBID
HEDTEC	0.7	MKEM 2,5 HA...	34.0	0.6	31.03.2004 0...		ISBID

Example: Simulation and automation of a ceramics factory

The screenshot displays a simulation and automation interface for a ceramics factory. The interface is divided into several main sections:

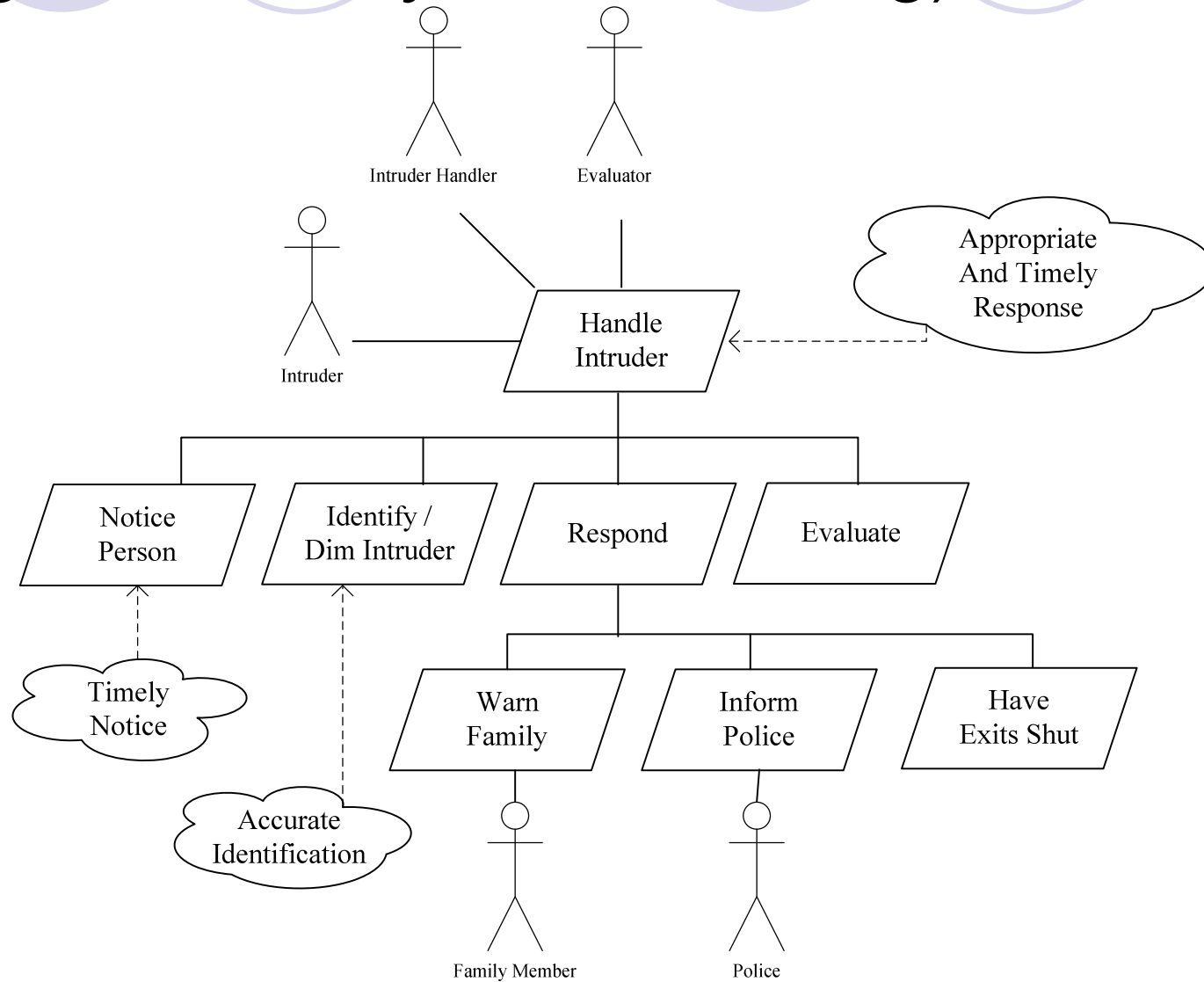
- Customer Window:** A form for entering production orders. It includes a dropdown for 'Product type' (currently 'treitud keraamikatoode'), a text field for 'Quantity' (set to '200'), and a 'Submit production order' button.
- Production Department Window:** A window showing the production schedule for the order. It contains a table with the following data:

Production Activity Name	Start Time	End Time
vormimine	Mon Sep 05 08:00:00 EST...	Mon Sep 05 12:32:00 EST...
viimistlemine	Tue Sep 06 12:33:00 EST...	Tue Sep 06 13:05:00 EST...
angoobimine	Mon Sep 12 08:00:00 EST...	Mon Sep 12 13:20:00 EST...
eelpoletus	Wed Sep 14 08:00:00 ES...	Wed Sep 14 16:00:00 ES...
jarelvimistlus	Fri Sep 16 08:00:00 EST 2...	Fri Sep 16 08:32:00 EST 2...
maalimine	Fri Sep 16 08:33:00 EST 2...	Fri Sep 16 11:53:00 EST 2...
glasuurimine	Mon Sep 19 08:00:00 EST...	Mon Sep 19 13:20:00 EST...
glasuurpoletus	Wed Sep 21 08:00:00 ES...	Wed Sep 21 16:00:00 ES...
dekoolimine	Fri Sep 23 08:00:00 EST 2...	Fri Sep 23 10:00:00 EST 2...
dekoolipoletus	Mon Sep 26 08:00:00 EST...	Mon Sep 26 09:30:00 EST...
pakkimine	Tue Sep 27 09:31:00 EST...	Tue Sep 27 10:19:00 EST...
- Production Activity Configuration Windows:** A vertical stack of windows for each activity, allowing users to set the 'Actual start time' (month, day, year, hour, minute) and 'Actual end time' (month, day, year, hour, minute). The activities and their start times are:
 - vormimisüksus: Sep 5, 2005, 08:00
 - angoobimisüksus: Sep 12, 2005, 08:00
 - viimistlusüksus: Sep 16, 2005, 08:00
 - glasuurimisüksus: Sep 19, 2005, 08:00
 - maalimisüksus: Sep 23, 2005, 08:00
 - poletusüksus: Sep 26, 2005, 08:00
 - ladu: Sep 27, 2005, 09:31

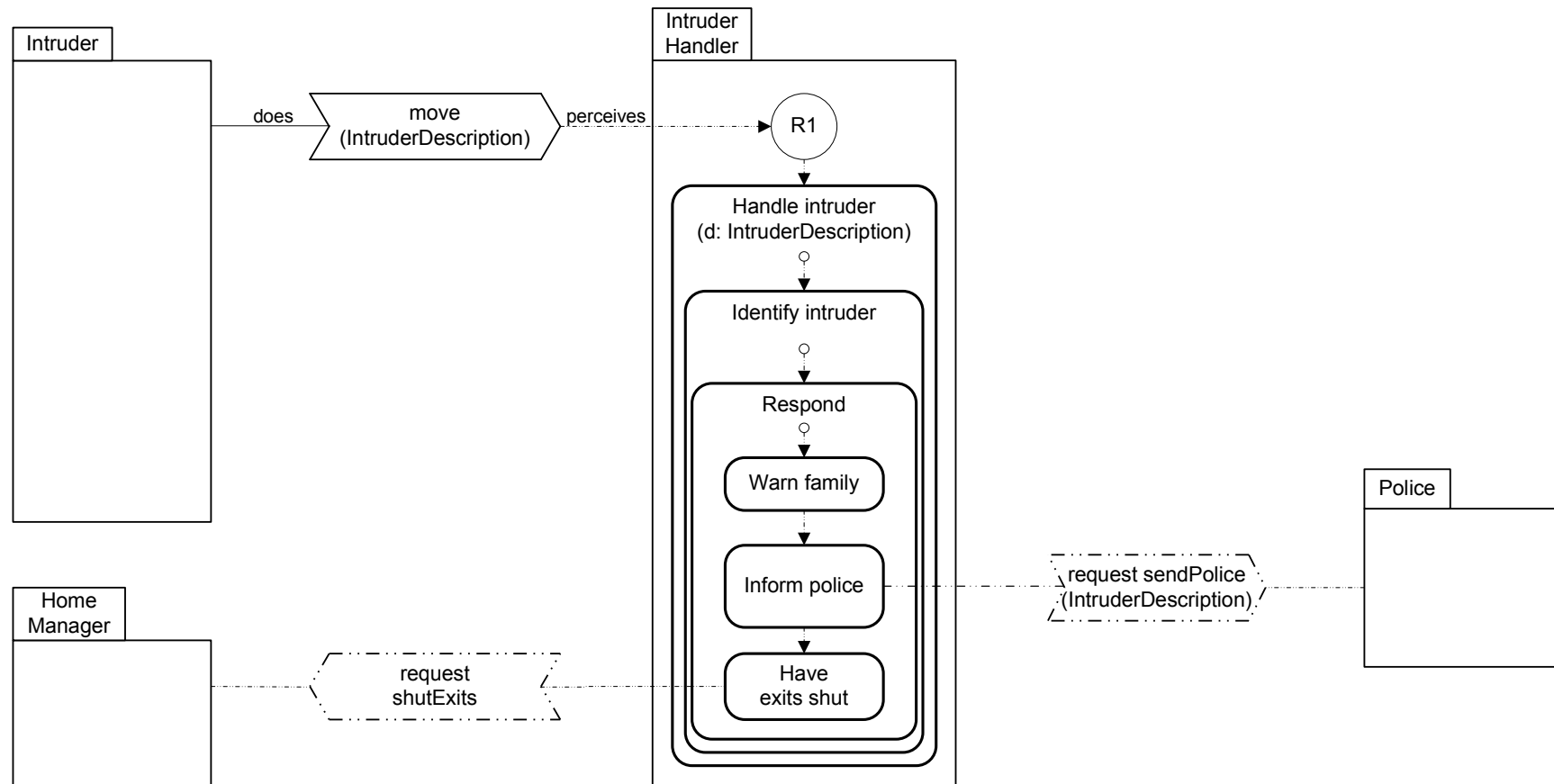
Generating agents from business models: Intruder handling (slide by Leon Sterling)

- 1 Computer vision in a networked home finds a stranger in the house.
- 1 Mining of family images shows he is not a friend or relative.
- 1 The house owner and his family and scheduled guests are warned to stay away via their smartphone, PDA or PC.
- 1 Police is automatically notified, and mining of the police image database brings up the records of the suspect.

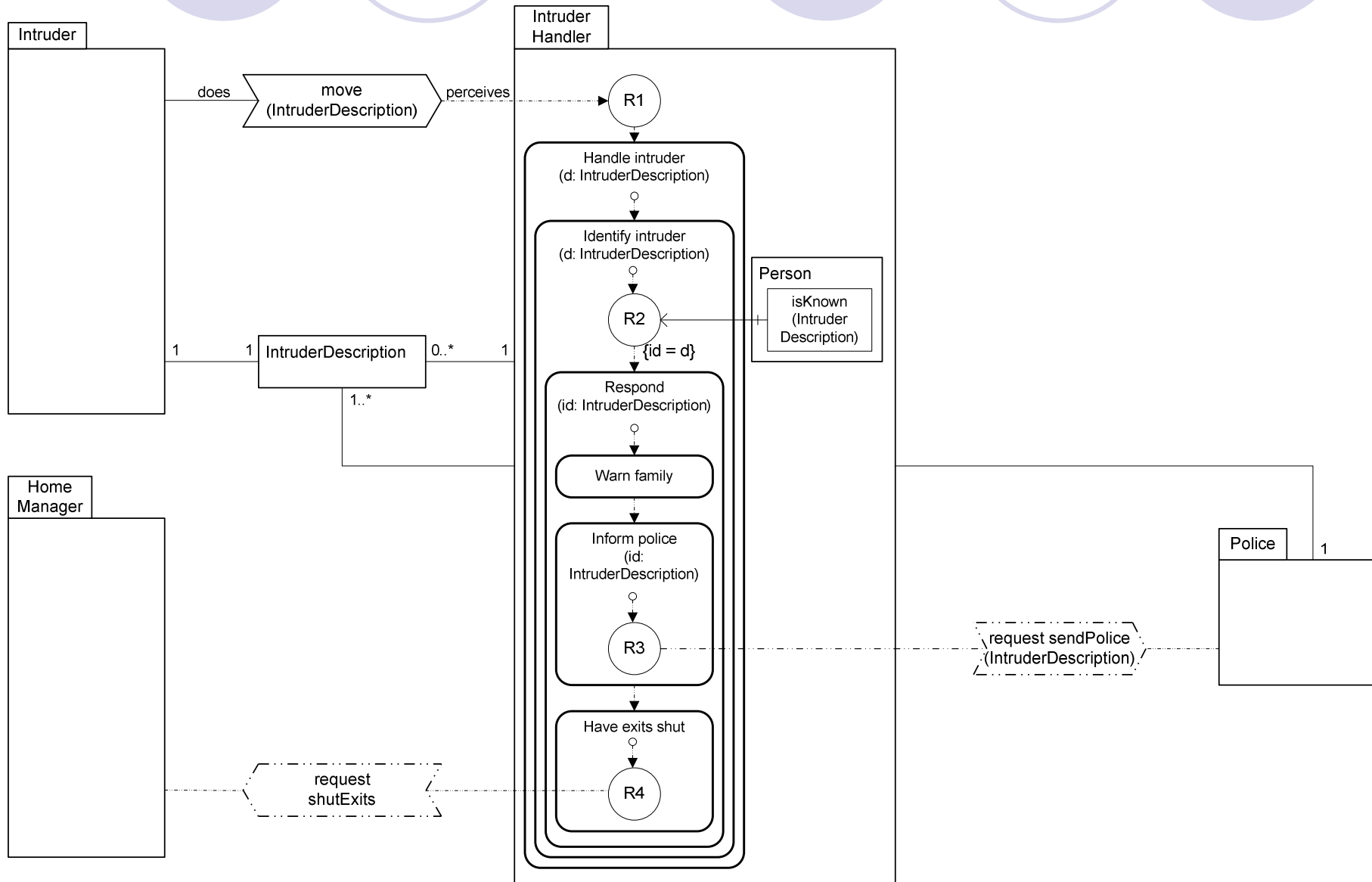
Function modelling (original slide by Leon Sterling)



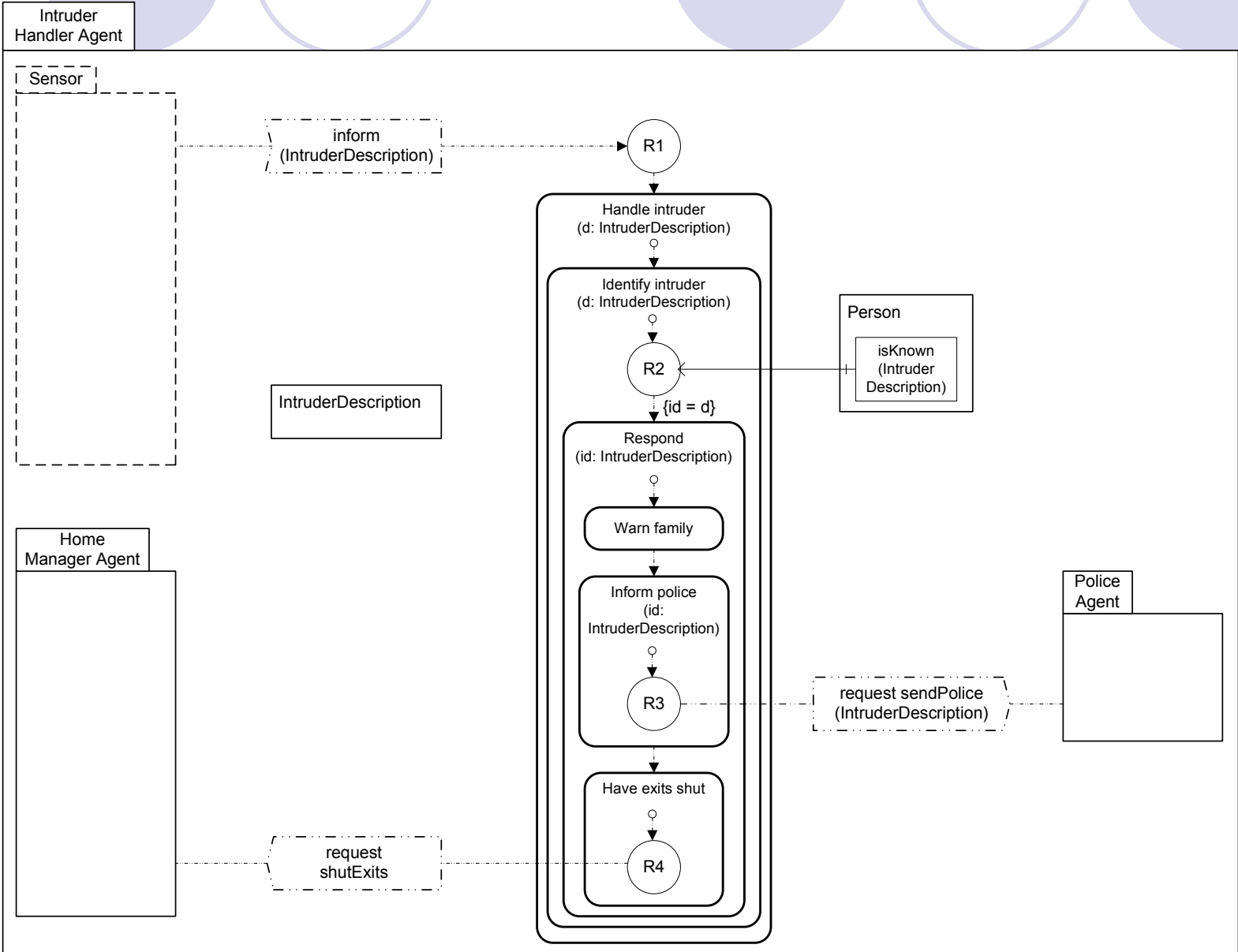
Conceptual domain modelling I



Conceptual domain modelling II



Platform-independent design



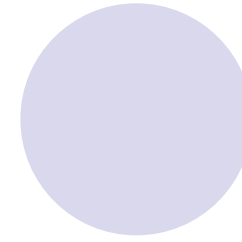
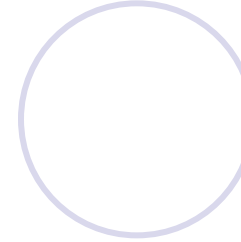
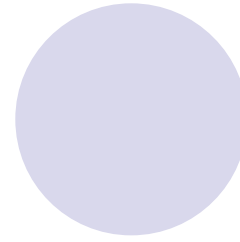
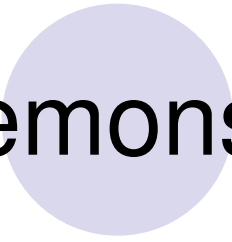
Simulation and implementation I

- 1 Activity diagrams of AORML can be straightforwardly transformed into the corresponding JADE constructs and executed in JADE:
 - ; actor types are mapped to JADE agent types;
 - ; object types are mapped to JADE object types;
 - ; activity types are mapped to *behaviour* types of JADE;
 - ; reaction rules are mapped to the constructs of JADE that invoke and sequence behaviours.

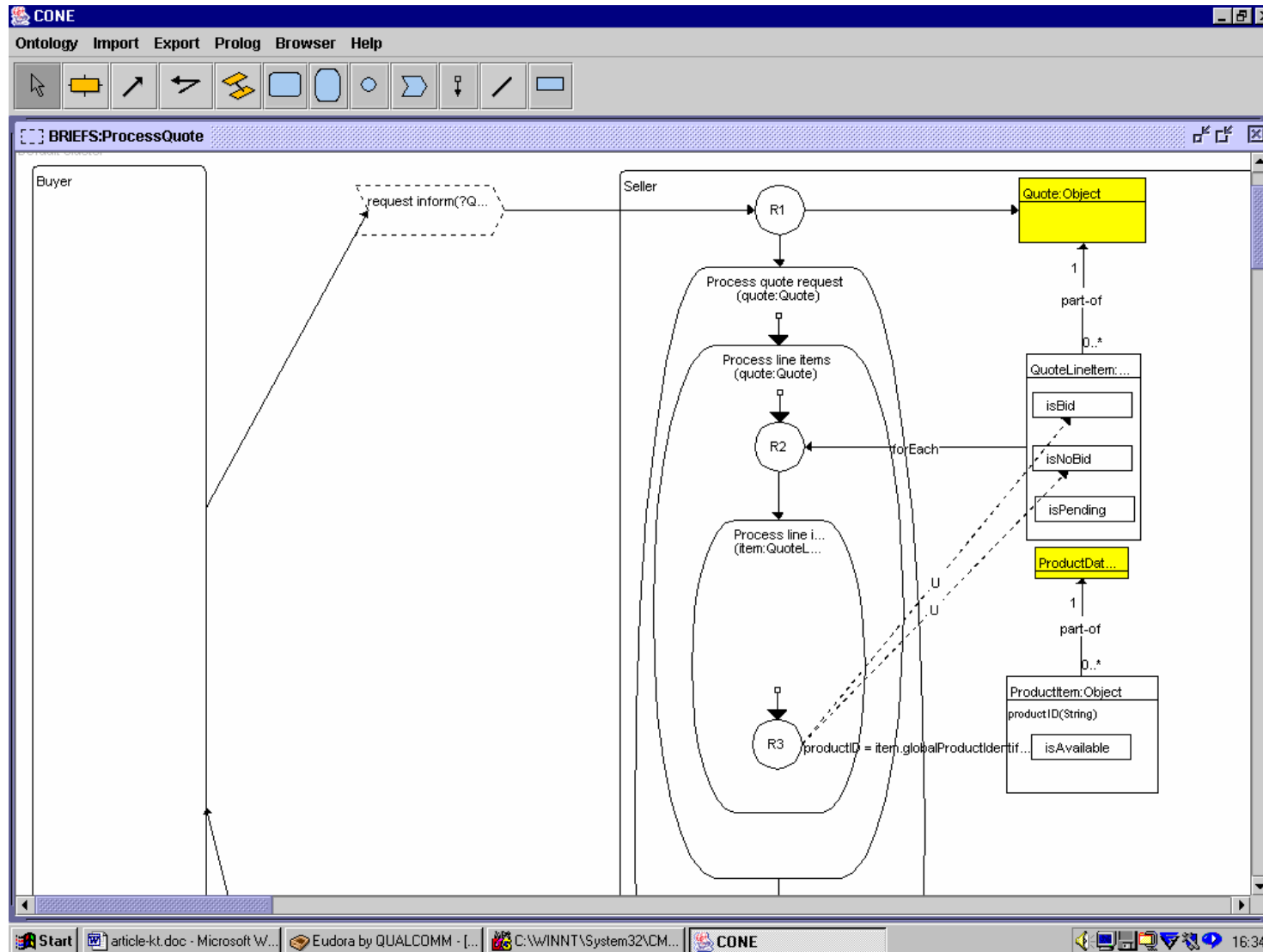
Simulation and implementation II

Notion of RAP/AOR	Object class in JADE	Object method of JADE (if applicable)
Object type	java.lang.Object	-
Agent type	jade.core.Agent	-
Elementary activity type	jade.core.behaviours. OneShotBehaviour	-
Sequential activity type	jade.core.behaviours. SequentialBehaviour	-
Parallel activity type	jade.core.behaviours. ParallelBehaviour	-
Execution cycle of a KPMC agent	jade.core.behaviours. CyclicBehaviour	-
Waiting for a message to be received	jade.core.behaviours. ReceiverBehaviour	-
Starting the first-level activity	jade.core.Agent	public void addBehaviour (Behaviour b)
Starting a sub-activity	jade.core.behaviours. SequentialBehaviour	public void addSubBehaviour (Behaviour b)
Starting a parallel sub-activity	jade.core.behaviours. ParallelBehaviour	public void addSubBehaviour (Behaviour b)
Start-of-activity activity border event type	jade.core.behaviours. OneShotBehaviour	public abstract void action()
Start-of-activity activity border event type	jade.core.behaviours. SequentialBehaviour, jade.core.behaviours. ParallelBehaviour	public abstract void onStart()
End-of-activity activity border event type	jade.core.behaviours.Behaviour	public int onEnd()
Agent message	java.lang.acl.ACLMessage	-

Demonstration



Tools





Alternative approach to implementation

- 1 Executable AOR domain models are transformed into equivalent XML-based *subjective* representations that are then interpreted and executed by software agents.
- 1 Advantage: design once, implement multiple times!

XML-schema for representing business process types (based on RuleML)

```
<xs:element name="businessProcess" type="businessProcessType"/>
<xs:complexType name="businessProcessType">
  <xs:sequence>
    <xs:element name="processTypeName" type="nameType"/>
    <xs:element name="perspective" type="nameType"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="reactionRule" type="reactionRuleType"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="reactionRuleType">
  <xs:all>
    <xs:element name="ruleName" type="nameType" minOccurs="0"/>
    <xs:element name="event" type="eventTermType"/>
    <xs:element name="body" type="bodyTermType" minOccurs="0"/>
    <xs:element name="head" type="actionEffectTermType" />
  </xs:all>
</xs:complexType>
```

Schema-based representation of a business process type

```
<reactionRule>
  <event>
    <startOfActivity>
      <activityTypeName>Process line items</activityTypeName>
    </startOfActivity>
  </event>
  <body>
    <forEach>
      <entityTypeName>QuoteLineItem</entityTypeName>
      <scope>
        <Var>quote</Var>
      </scope>
    </forEach>
  </body>
  <head>
    <mainAction>
      <startActivity>
        <activityTypeName>Process line item</activityTypeName>
        <Var type="QuoteLineItem"/> item</Var>
        <activityMode> sequential</activityMode>
      </startActivity>
    </mainAction>
  </head>
</reactionRule>
```



Summary

- 1 It should be possible to describe problems at the level of conceptual domain modelling.
- 1 Tasks of planning, coordination and learning should be “raised” to the domain level.
- 1 Transformation rules between levels should be fully elaborated.