F13: Model-driven Development of Agent Systems: A Platform-Independent Model for Agents to improve Interoperability between Platforms

Christian Hahn
Multiagent System Group
German Research Center for Artificial Intelligence (DFKI GmbH), Saarbrücken

Guest researcher, SINTEF, Oslo, as part of the INTEROP project
Structure

• Motivation
• Platform-independent model for agents (PIM4Agents)
• Metamodel for JACK (JackMM)
• Metamodel for JADE (JadeMM)
• Platform-independent model for SOA (PIM4SOA)
• Transformations
  – Horizontal transformations: From SOA to agents
  – Vertical transformations:
    • PIM4Agents to JackMM
    • PIM4Agents to JadeMM
• Conclusion
Introduction
Major Obstacle: Lack of Interoperability
Motivation

• Various agent-oriented methodologies and metamodels exist to describe multiagent systems (MAS) in an abstract manner
• Specialise on particular parts of the MAS and only few work has been invested to derive a common standardisation
• Limits the impact of agent-related systems in commercial applications
• Present a metamodel for agent systems (PIM4Agents) that abstracts from existing agent-oriented methodologies and frameworks
• Agent-oriented software development process in accordance to model-driven development (MDD)
Two Objectives

1. PIM4Agents should bridge the gap between the various existing agent-oriented modelling methods and thus improve the interoperability between agent frameworks

2. PIM4Agents should define a platform-independent abstraction that can be used to integrate and define mappings from service-oriented architectures (SOA), Web services, peer-to-peer (P2P) and Grid execution platforms

Solution:

Metamodelling & model-driven development
Interoperability between Platforms

Symbols
- Metamodel
- Concept
- Relationship
- Correspondence
Metamodelling
Metamodelling

M3: Meta-Metamodel
M2: Metamodel
M1: (User-) Model
M0: Instance / Real System

<<instance of>>
e.g. MOF
**Agent** is the autonomous entity capable of acting in the environment.

**Cooperation** between agents.

**Role**: domain-specific & application-specific roles.

**Message** exchange is described in protocols.

**Behaviour** describes the internal processes.

**Protocol** describes interaction between agents/organisations.
**Team** specifies the structure of one or more entities (Teams/Agents)

**Role** specifies a role as a type by a listing the types of the events the role can deal with

**Events**: type of stimuli a team, role, or team plan reacts to or posts

**Agent** handles events and executes plans

**Plan/Team Plan** specifies the behaviour of an agent/team in reaction to a specific event
Agent is described by an aggregation of the concepts Message-Queue, AgentStates and Scheduler. Behaviour is the aggregation of CompositeBehaviour and SimpleBehaviour. The behaviour implements the agent tasks and intentions.
**ServiceProvider** specifies an entity describing and specifying in its turn services, roles and constraints.

**CollaborationUse**: the model element to represent a usage of a service.

**Behaviour**: abstract class for the specification of messages sequences within a service.

**Role**: the model element to represent a usage of a service.

**Collaboration** represents a pattern of interaction between participating roles. A binary collaboration specifies a service.
Model Transformations
Model-Driven Development

- Current state of the art in Model Driven Engineering (MDE) is much influenced by the ongoing standardisation activities around the OMG Model Driven Architecture® (MDA®)
- MDA is a framework which defines a model-driven approach to software systems development
- MDA encapsulates many important ideas - most notably the notion that real benefits can be obtained by using visual modelling languages to integrate the huge diversity of technologies used in the development of software systems
MDA’s main abstraction Levels

- **Computation independent model (CIM)**
  - The computational independent viewpoint is focused on the environment of the system and on the specific requirements of the system.
  - Hides the structural details and the details related to the targeted platform.

- **Platform independent model (PIM)**
  - A platform independent model is a view of the system from a platform independent viewpoint.
  - Exhibits platform independence and is suitable for use with a number of different platforms of similar types.
  - Gathers all the information needed to describe the behaviour of the system in a platform independent way.

- **Platform specific model (PSM)**
  - A platform specific model is a view of the system from the platform specific viewpoint.
  - Combines the specifications in the PIM with the details that specify how the system uses a particular type of platform.
  - Represents the PIM taking into account the specific platform characteristics.
Horizontal Transformations: From PIM4SOA to PIM4Agents

- **PIM4SOA**
  - Collaboration
  - ServiceProvider
  - Process
  - Message
  - Role
  - Collaboration

**Transformation**

**PIM4Agents**

- Organisation
- Agent
- Behaviour
- Message
- Role
- Protocol
Vertical Transformations (I): From PIM4Agents to JackMM

- **PIM4Agents**
  - Organisation
  - Agent
  - Behaviour
  - Message
  - Role
  - Protocol

- **Transformation**

- **JackMM**
  - Team
  - Team
  - TeamPlan
  - Event
  - Role
  - Event
Vertical Transformations (II): From PIM4Agents to JadeMM

- **PIM4Agents**
  - Agent
  - Organisation
  - Message
  - Protocol
  - Role
  - Behaviour

- **Transformation**

- **JadeMM**
  - Agent
  - ACLMessage
  - Behaviour
Eclipse-based Framework

Eclipse-based integrated development environment (IDE):
Rational Software Modeler + Atlas Transformation
Language transformation engine + MOFScript
Conclusion

• Defined a platform independent metamodel for agents (called PIM4Agents)
• Illustrated how this metamodel could be used in a MDD scenario to simplify the implementation of agents systems
• Described how to transfer service-oriented architectures to executable agent systems,
  – Illustrated how concepts of a platform independent model for SOA (PIM4SOA) can be transformed to agent-oriented concepts described by the PIM4Agents
  – Illustrated how abstract agent-related concepts could be mapped to agent-oriented execution platforms.
• Benefit of the PIM4Agents:
  Vertical mappings to agent-related frameworks can be reused and shared for different application-oriented metamodels.
Framework

Eclipse-based integrated development environment (IDE):
Rational Software Modeler + Atlas Transformation
Language transformation engine + MOFScript

ATL engine

Source metamodel (EMF)
Target metamodel (EMF)

Source metamodel (EMF)
Target metamodel (EMF)

Service Model

PIM4Agents model

Jack code

MOFScript engine

Source metamodel (EMF)
MOFScript code
rule ServiceProvider2Team

from serviceProvider : SOA!ServiceProvider

to team : AgentMM!Team

    name <- serviceProvider.name,
    Roles_Performed <- rolePerforms,
    Plans <- plans,
    Roles_Required <- rolesRequired

plans : AgentMM!TeamPlans

    uses_plan <- teamPlan

rolePerforms : AgentMM!Roles_Performed

    performs_role <- 'roleAgent'

TEAMPlan : AgentMM!TEAMPlan

    name <- serviceProvider.name+'TEAMPlan',
    Roles <- roles

rolesRequired : AgentMM!Roles_Required

    requires_role <- roleRequired
Metamodelling
**Agent** is the autonomous entity capable of acting in the environment

**Behaviour** describes the internal processes

**Protocol** describes interaction between agents/organisations

**Message** exchange is described in protocols

**Cooperation** between agents

**Role**: domain-specific & application-specific roles
**Agent** handles events and executes plans

**Team** specifies the structure of one or more entities (Teams/Agents)

**Events:** type of stimuli a team, role, or team plan reacts to or posts

**Plan/Team Plan** specifies the behaviour of an agent/team in reaction to a specific event

**Role** specifies a role as a type by a listing the types of the events the role can deal with
**Agent** is described by an aggregation of the concepts Message-Queue, AgentStates and Scheduler.

**Behaviour** is the aggregation of CompositeBehaviour and SimpleBehaviour. The behaviour implements the agent tasks and intentions.
**ServiceProvider** specifies an entity describing and specifying in its turn services, roles and constraints

**Collaboration** represents a pattern of interaction between participating roles. A binary collaboration specifies a service

**CollaborationUse**: the model element to represent a usage of a service

**Behaviour**: abstract class for the specification of messages sequences within a service

**Role**: the model element to represent a usage of a service