A Model Driven Development of Platform-neutral Agents

Inmaculada Ayala Viñas, Mercedes Amor Pinilla_y
Lidia Fuentes Fernández
{ayala, pinilla, lff}@lcc.uma.es

Grupo CAOSD
http://caosd.lcc.uma.es/

MATES 2010
Leipzig, September

Outline

Motivation

- There are a plenty of proposal that try to bridge the classical gap between design and implementation in Multi-Agent Systems (MAS)
  - MaSE
  - Tropos
  - DSML4MAS
  - …

They generate code only for a small set of agent platforms (only one in the most of cases)

Acceptable for some systems

A serious limitation for Ambient Intelligence

GISUM

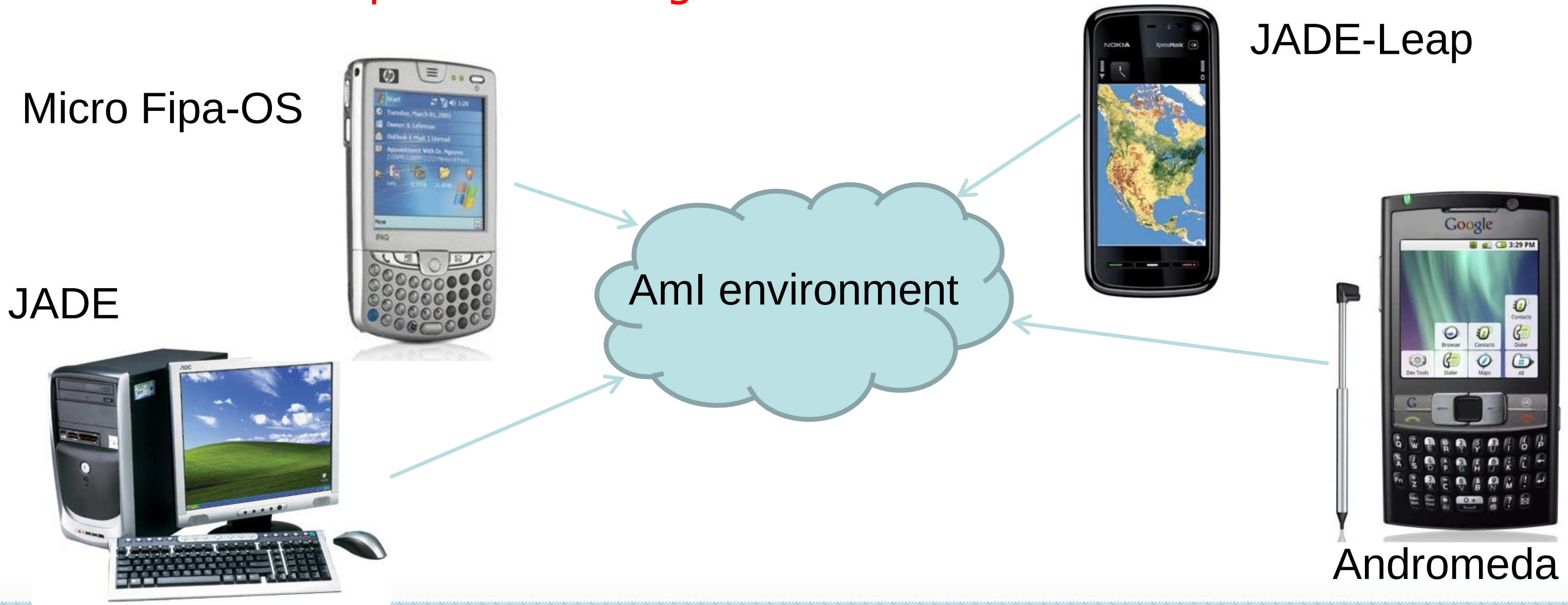
Motivation

- Ambient Intelligence (AmI) is a paradigm which proposes the development of intelligent environments.

  - Smarthomes
  - Education
  - Health
  - Transport
    - Vehicular Ad-Hoc Network (VANET)

Motivation

- AmI environments are composed of heterogeneous devices (desktop computers, mobile phones, sensors, PDAs, …)
  - Agent platforms (APs) provide versions adapted to lightweight devices
  - It is expected that new APs for new devices will appear in the next few years
- Solutions to bridge the gap between design/implementation must consider an open set of target APs



G I S U M

Motivation

- Agents of an AmI system must be able to communicate and cooperate with agents running on devices with different APs.
  - VANET as an example
    - Standardization is not possible
    - All devices will not have the same AP installed
      - E.g. on board computer, mobile phones, GPS, Web services/servers, etc.
  - JADE-leap versions (J2SE and MIDP)
    - Interoperability is not guaranteed

“The development of MAS for AmI environments is not affordable using the traditional agent development processes that normally consider the generation of agents for a single AP”

Our Goal

- Our Goal
  - Automatic derivation of agents running on top of different agent platforms, in AmI environments
- Model driven development (MDD), helps to bridge the gap between design/implementation
  - Source model: platform independent models (PIM)
    - Our PIM is PIM4Agents (a platform independent AP)
  - Target model: platform specific models (PSM)
    - Our PSM is Malaca (a platform neutral agent model)
  - Transformation between PIM and PSM
    - From PIM4Agents to Malaca

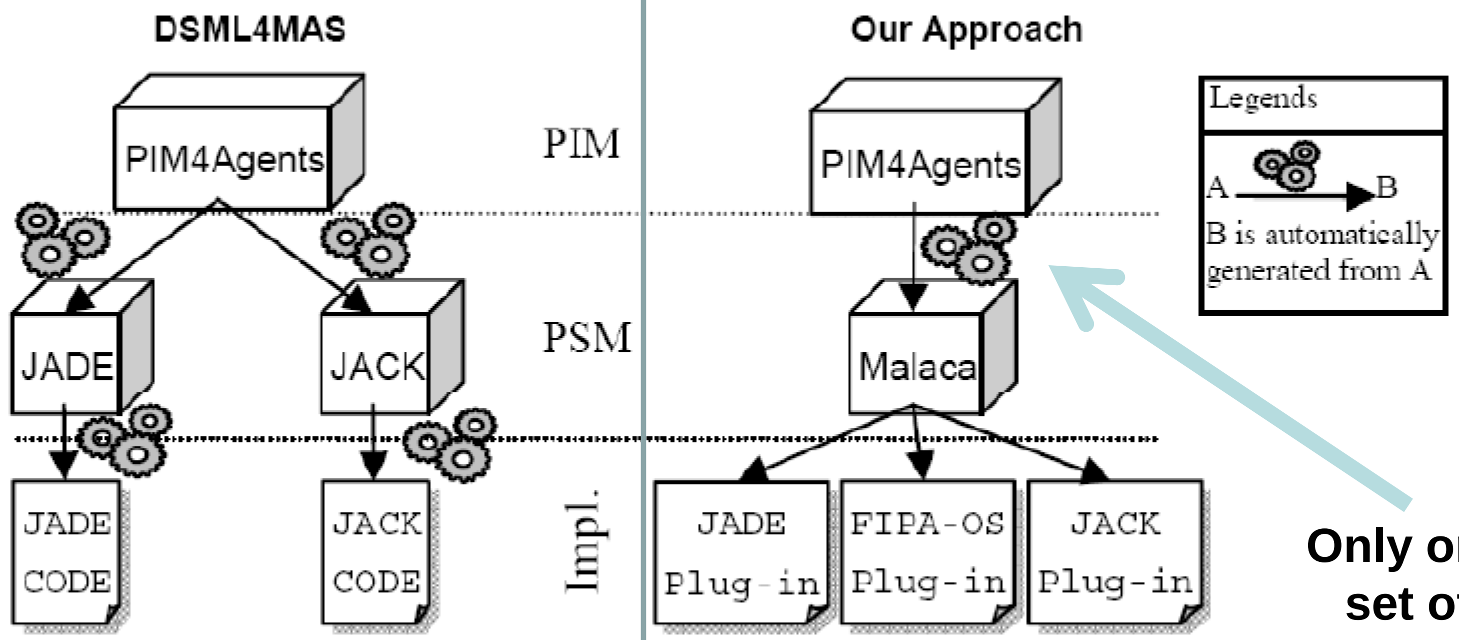
Our Goal

- Why PIM4Agents ?
  - It is possible to represent concepts from different agent types
  - It is easy to specify MAS for different domains.
  - The DDE Tool helps to specify different views of MAS.

- Why Malaca ?
  - A platform-neutral specific metamodel (PNSM)
  - Execution on top of any FIPA compliant platforms
  - MAD Tool

Our approach

Classical MDD approach



Only one set of transformation rules !!!

One simple plug-in per AP

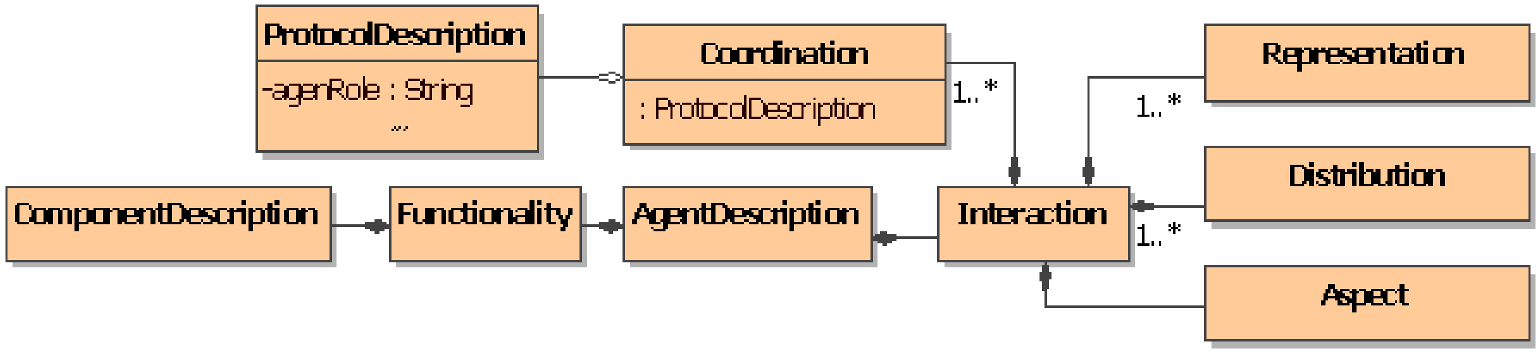
Our approach

- The internal architecture of a Malaca agent represents separately application-specific functions from extra-functional agent properties
  - Better internal modularization
  - Based on the composition of components and *aspects*[1] (e.g. distribution aspect)
- Malaca agents are configured with two DSLs (Domain specific languages)
  - MaDL (Agent configuration language)
  - ProtDL (Protocol description language)
  - MaDL and ProtDL are part of the target metamodel

[1] AOSD (Aspect-oriented Software Development, http://aosd.net)

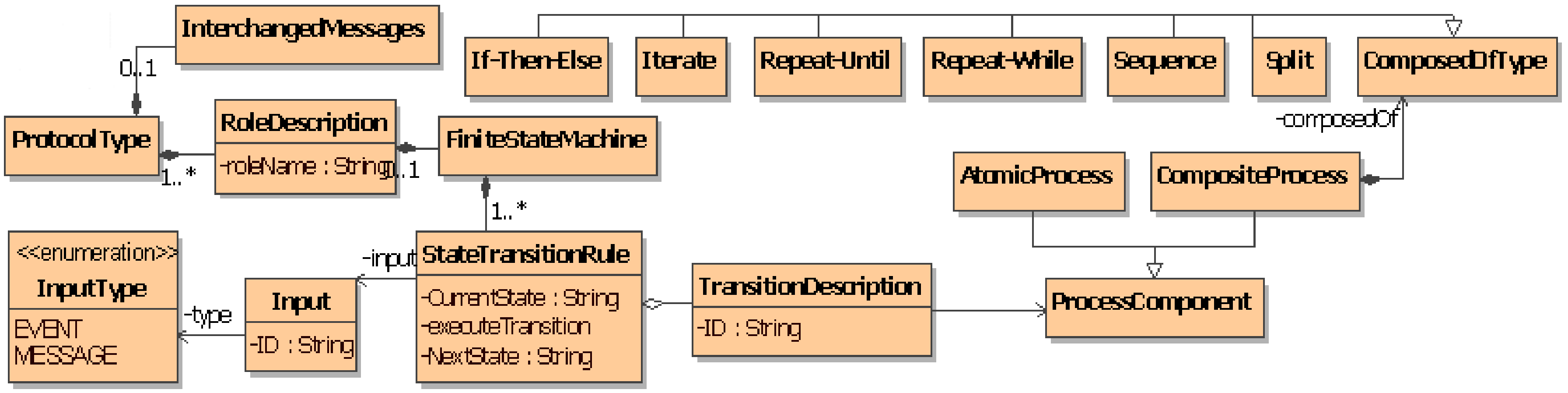
Our approach

- MaDL
  - Description of the agent architecture in view of the agent functionality and the agent interaction
  - Web services are integrated naturally into Malaca Architecture
  - Identified aspects
    - Learning, Context-awareness, **Coordination**, Distribution, Representation



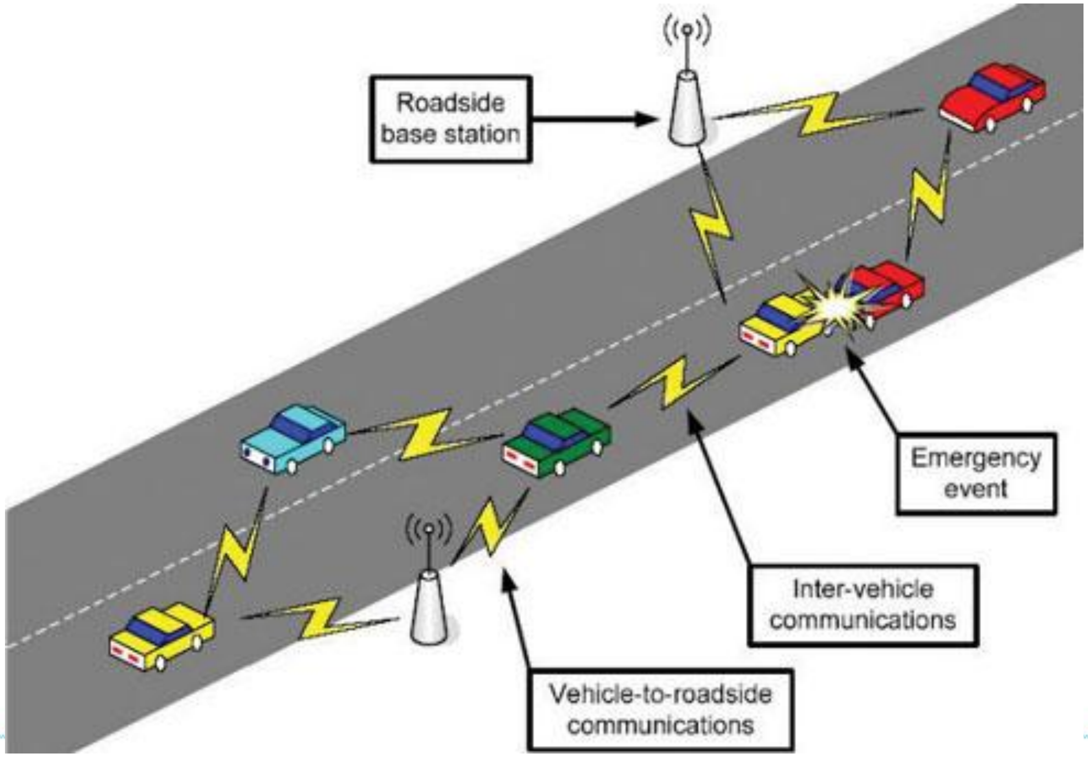
Our approach

- ProtDL
  - Coordination aspect



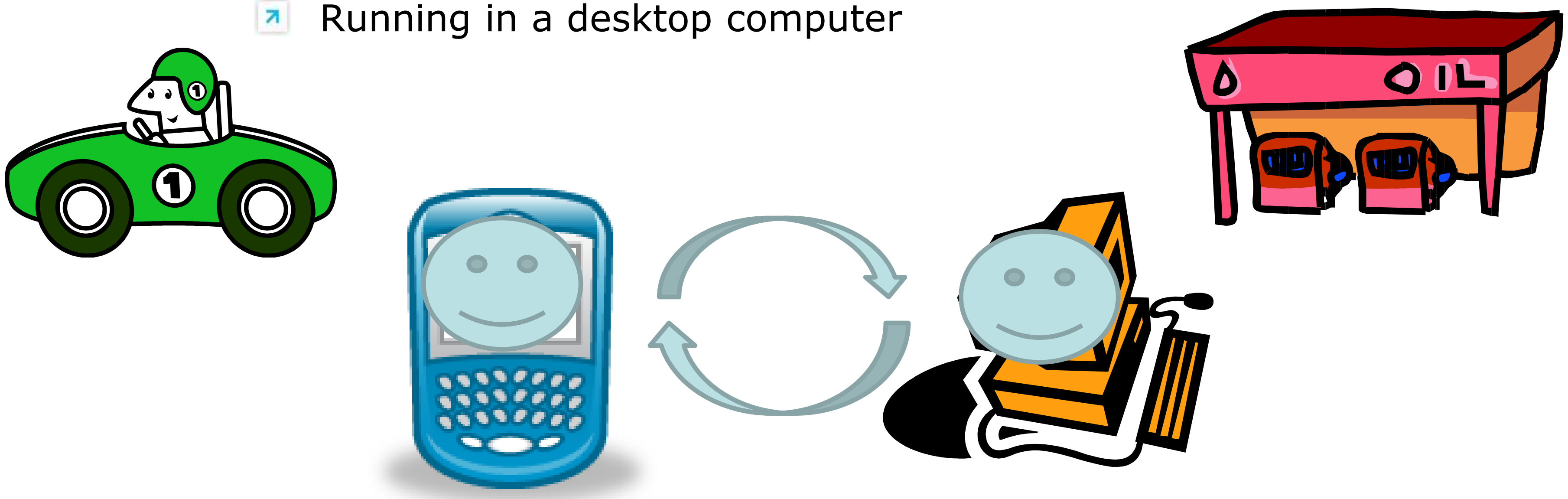
Motivating case study

- VANETs are a form of mobile Ad-hoc network that allows information exchanging between users in vehicles and between users in vehicles and service providers, along roads.


Roadside base station
Emergency event
Inter-vehicle communications
Vehicle-to-roadside communications

Motivating case study

- MAS for VANET
  - Agents for vehicles (*VehicleAgent*)
    - Weather forecast is provided by a web service
    - Location of gas station
    - Running in a on-board computer
  - Agents for gas stations (*GasStationAgent*)
    - Negotiation with agent for vehicle
    - Running in a desktop computer

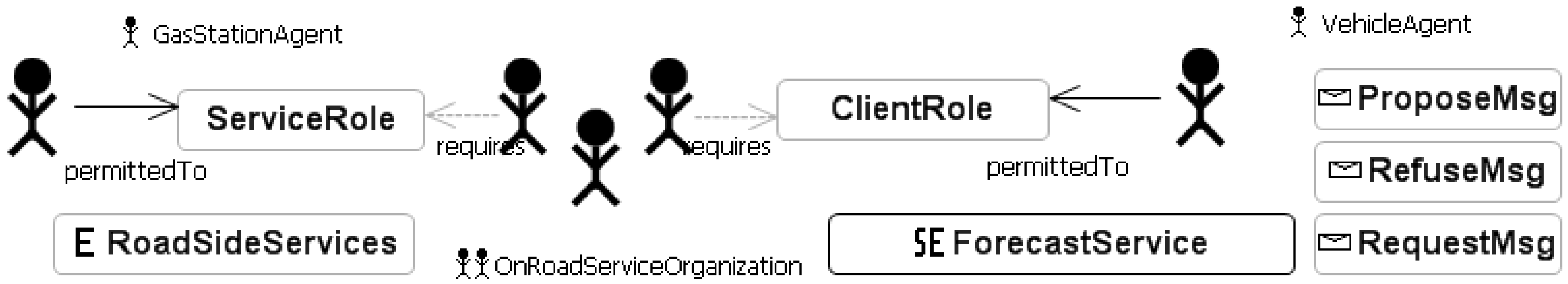


Design with DDE Tool

- PIM4Agents
  - DSML4MAS approach PIM
  - Different views of MAS each focused in a specific aspect of MAS
- DDE Tool
  - IDE for DSML4MAs
    - Graphical DSL
    - Transformation process
  - A diagram for each PIM4Agents view
- Case study
  - 11 diagrams
  - MAS, Interaction and Plan

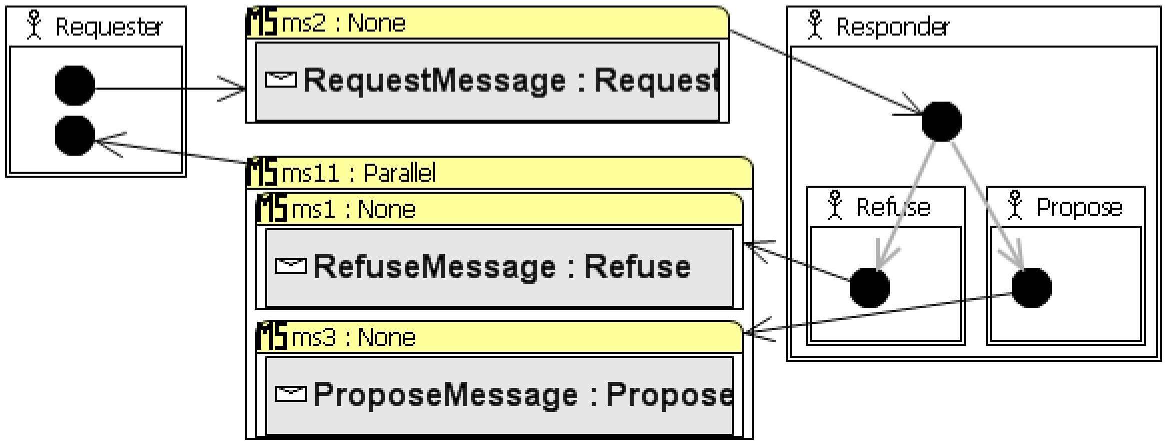
Design with DDE Tool

- Multi-agent System diagram



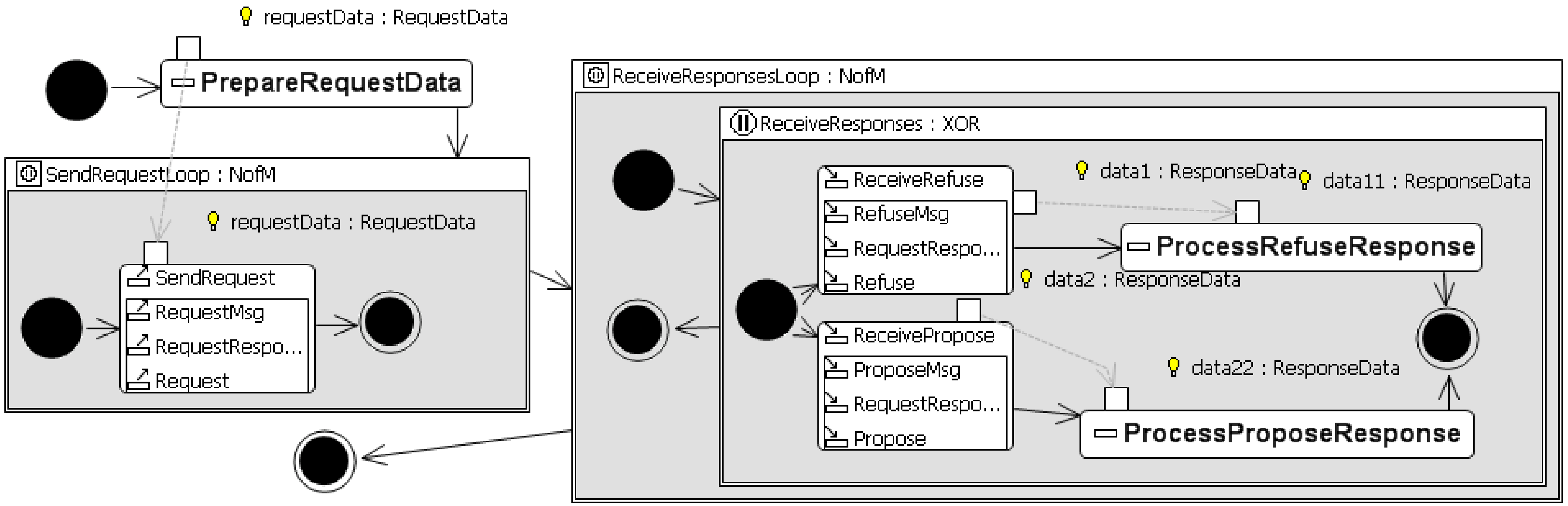
Design with DDE Tool

- Interaction diagram



Design with DDE Tool

- Behavior diagram



Generation of Malaca Agents

- Rules for MaDL

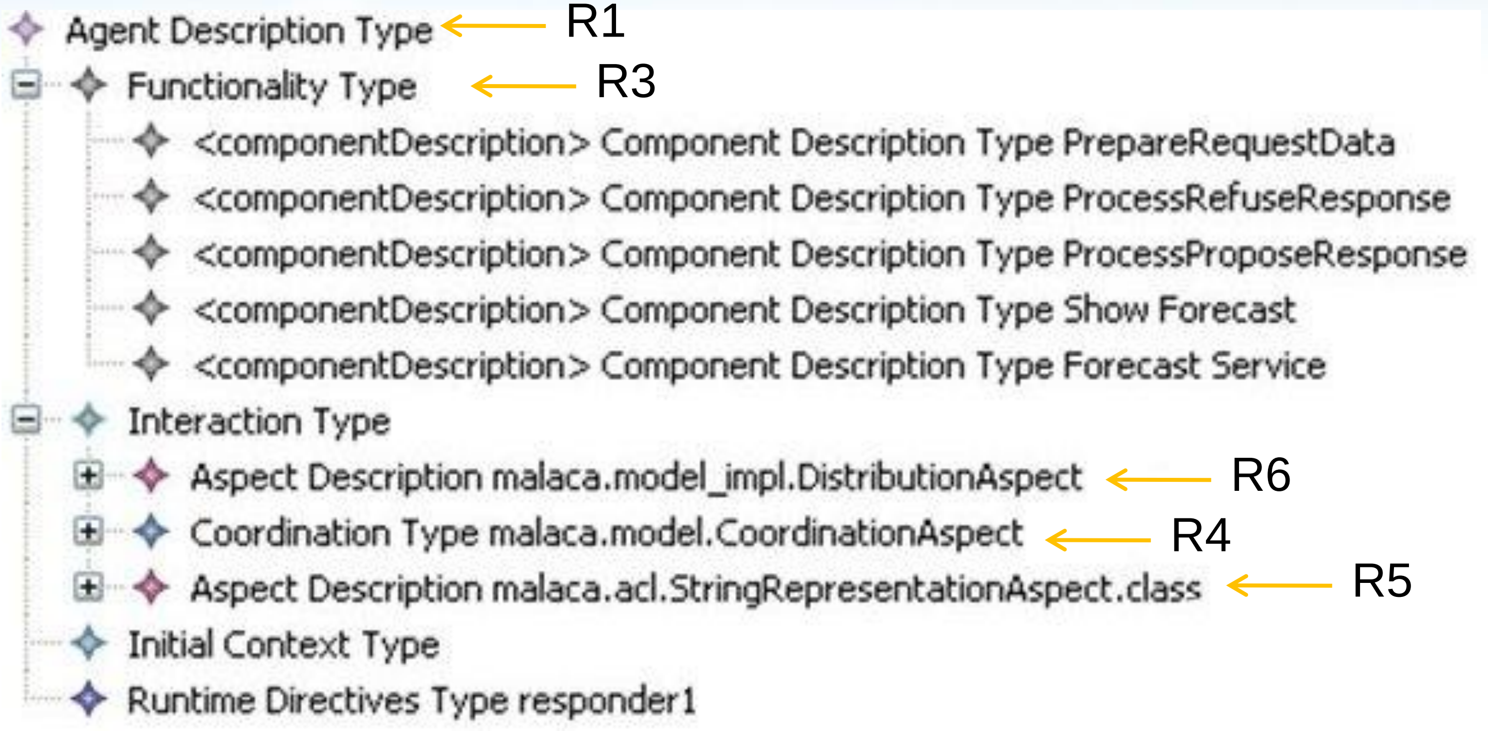
	PIM4Agents	MaDL
R1	Agent	AgentDescription
R2	AgentInstance	AgentDescription
R3	InternalTask	ComponentDescription
R4	DomainRole	Coordintation
R5	Agent	AspectDescription (Representation)
R6	Agent	AspectDescription (Distribution)

Generation of Malaca agents

- Rules for ProtDL

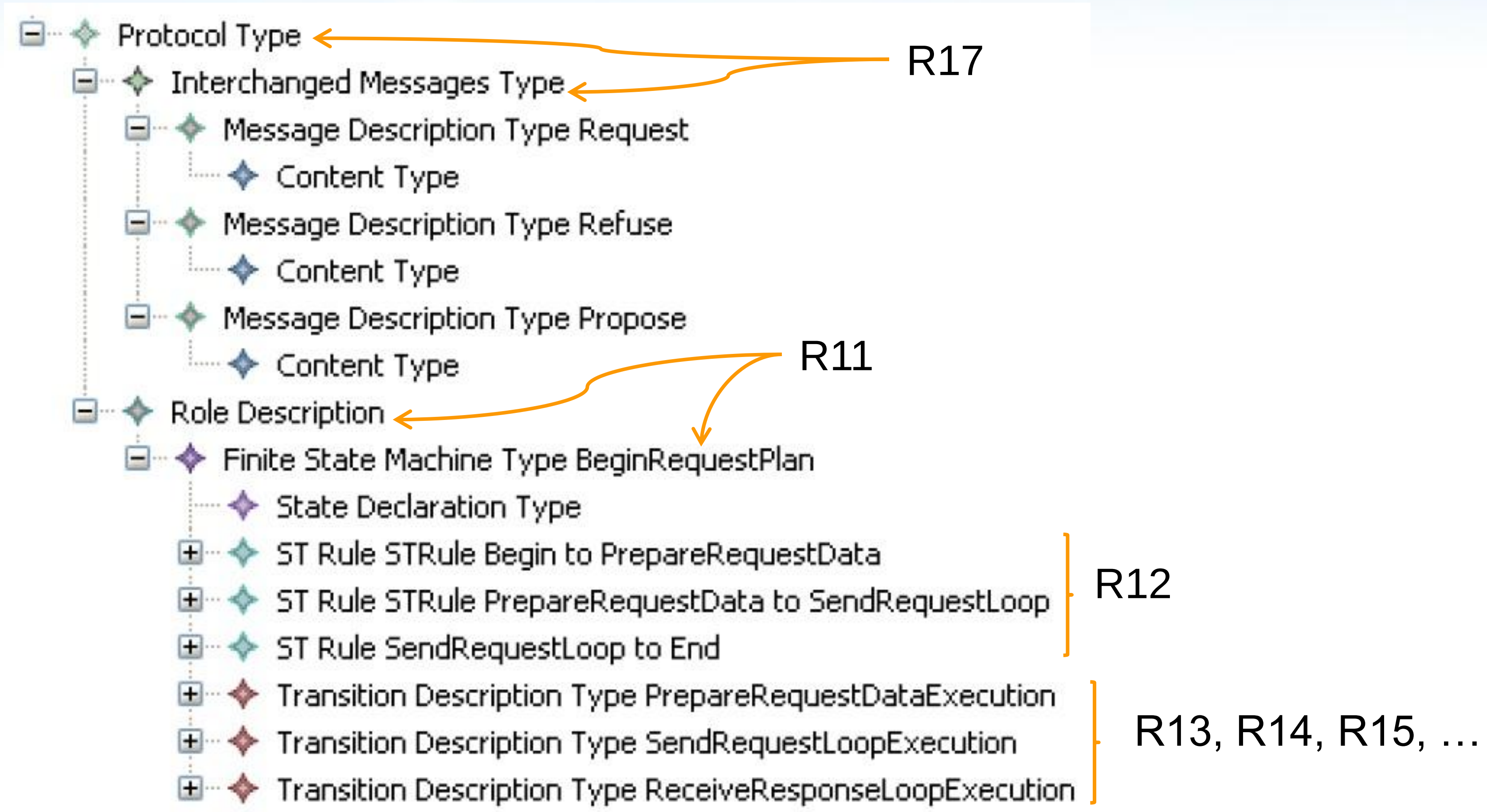
	PIM4Agents	ProtDL
R7	Protocol	Protocol
R8	Actor	RoleDescription
R9	MessageFlow, MessageFlow	StateTransitionRule
R10	MessageFlow, MessageFlow	TransitionDescription
R11	Plan,String	RoleDescription
R12	Activity, Activity	StateTransitionRule
R13	Activity, Activity	TranstionDescription
R14	InternalTask	ProcessComponent
R15	InvokeWS	ProcessComponent
R16	Split	ProcessComponent
R17	Protocol, Organization	Protocol

Generation of Malaca Agents



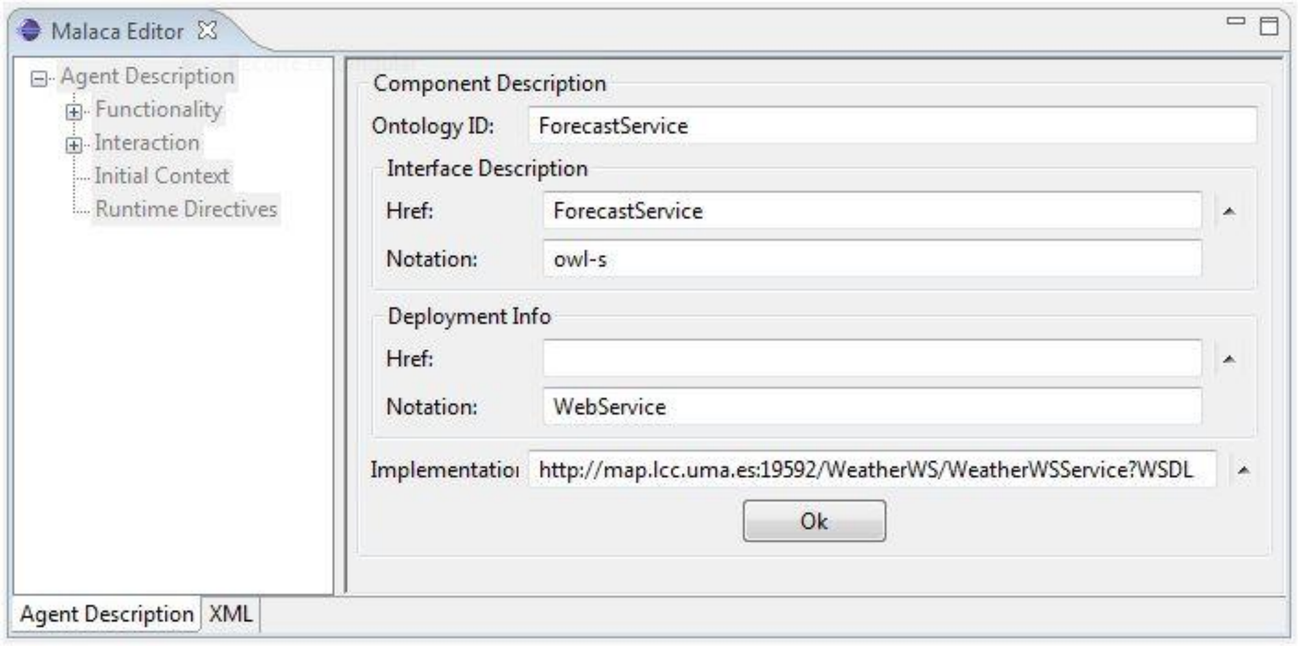
Agent Description Type
Functionality Type
<componentDescription> Component Description Type PrepareRequestData
<componentDescription> Component Description Type ProcessRefuseResponse
<componentDescription> Component Description Type ProcessProposeResponse
<componentDescription> Component Description Type Show Forecast
<componentDescription> Component Description Type Forecast Service
Interaction Type
Aspect Description malaca.model_impl.DistributionAspect
Coordination Type malaca.model.CoordinationAspect
Aspect Description malaca.acl.StringRepresentationAspect.class
Initial Context Type
Runtime Directives Type responder1

Generation of Malaca Agents


Protocol Type
Interchanged Messages Type
Message Description Type Request
Content Type
Message Description Type Refuse
Content Type
Message Description Type Propose
Content Type
Role Description
Finite State Machine Type BeginRequestPlan
State Declaration Type
ST Rule STRule Begin to PrepareRequestData
ST Rule STRule PrepareRequestData to SendRequestLoop
ST Rule SendRequestLoop to End
Transition Description Type PrepareRequestDataExecution
Transition Description Type SendRequestLoopExecution
Transition Description Type ReceiveResponseLoopExecution

Deployment of Malaca Agents

- Malaca Deployment phase consists of configuring the appropriate components and aspects implementation.
  - Configuring the distribution aspect for each agent
  - Selecting the component implementations that provide the agent functionality



Component Description

Ontology ID: ForecastService

Interface Description

Href: ForecastService

Notation: owl-s

Deployment Info

Href:

Notation: WebService

Implementatioı http://map.lcc.uma.es:19592/WeatherWS/WeatherWSService?WSDL

Discussion

- Cost of including a new agent platform
  - PIM4Agents
    - Specify the new metamodel $\longrightarrow$ MOF, Ecore
    - One set of model to model transformation rules $\longrightarrow$ ATL
    - One set of model to code transformation rules $\longrightarrow$ MOFScript, Jet
  - Malaca
    - A new plug-in in charge of instantiating the corresponding platform-dependant communication sub-sytem. $\longrightarrow$ Java

Discussion

- Optimization of the generated code
  - PIM4Agents
    - 91 classes generated for JADE and 34.06% dummy code
  - Malaca
    - 9 classes
    - Malaca Framework optimization

- Limitations of AmI domain
  - AmI environment is made up of different devices, which could contain agents running of top of different APs
    - The interoperability is guaranteed between Malaca Agents
    - The interoperability is not guaranteed between Jade versions

Conclusions and future work

- MDD is the most natural approach to automate the derivation of agent implementations from high level agent models, considering different target APs.

- The process presented in this paper significantly simplifies this process by using Malaca, a platform-neutral agent architecture.
  - Malaca is able to execute on top of different AP.
  - New devices/APs implies only the implementation of a simple plug-in.
  - Malaca agents interoperate with Web services

Conclusions and future work

- Comparing our approach with a DSML4MAS-like process
  - Including a new AP require less effort and user skills in our approach
  - Generated agents are more optimized than in other approaches
  - Malaca agents are interoperable even with different profiles of AmI devices

- As future work, we plan to extend PIM4Agents with new properties (modelled as *aspects*) like context-awareness and learning

The end

Thank you for your attention