# MODEL-DRIVEN AGENT ORIENTED SOFTWARE ENGINEERING

Dr Nikolaos Spanoudakis
Teaching/Research Staff
School of Production Engineering and Management
Applied Mathematics and Computers Laboratory
Technical University of Crete
Tel. : +30 28 21 03 77 44
Email: nikos@science.tuc.gr
www : users.isc.tuc.gr/~nispanoudakis

EASSS 2015

26 July, Catania, Italy

# Tutorial Summary



- Motivation and Background
  - Software Engineering
  - Model-driven engineering
- Agent Oriented Software Engineering
- AOSE Methodologies
  - Protocols Specification
- Model-Driven AOSE with ASEME
  - The Domain-Specific Language
  - The process
  - The tools

# Motivation and Software Engineering



- I know how to program, look, this is a simple “Hello world”, I did it in 10 seconds!!

- Why do I need Software Engineering?
- Why should I even spend my time drawing models?

# Teamwork needs



- Process
- Vocabulary
- Role assignment
- Documentation
- Planning
- Progress Assessment
- …

# The Scientific Domain



- The **Software Engineering** (IEEE, 1990) discipline:
  - applies a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software
  - studies such approaches

# The Scientific Domain (2)

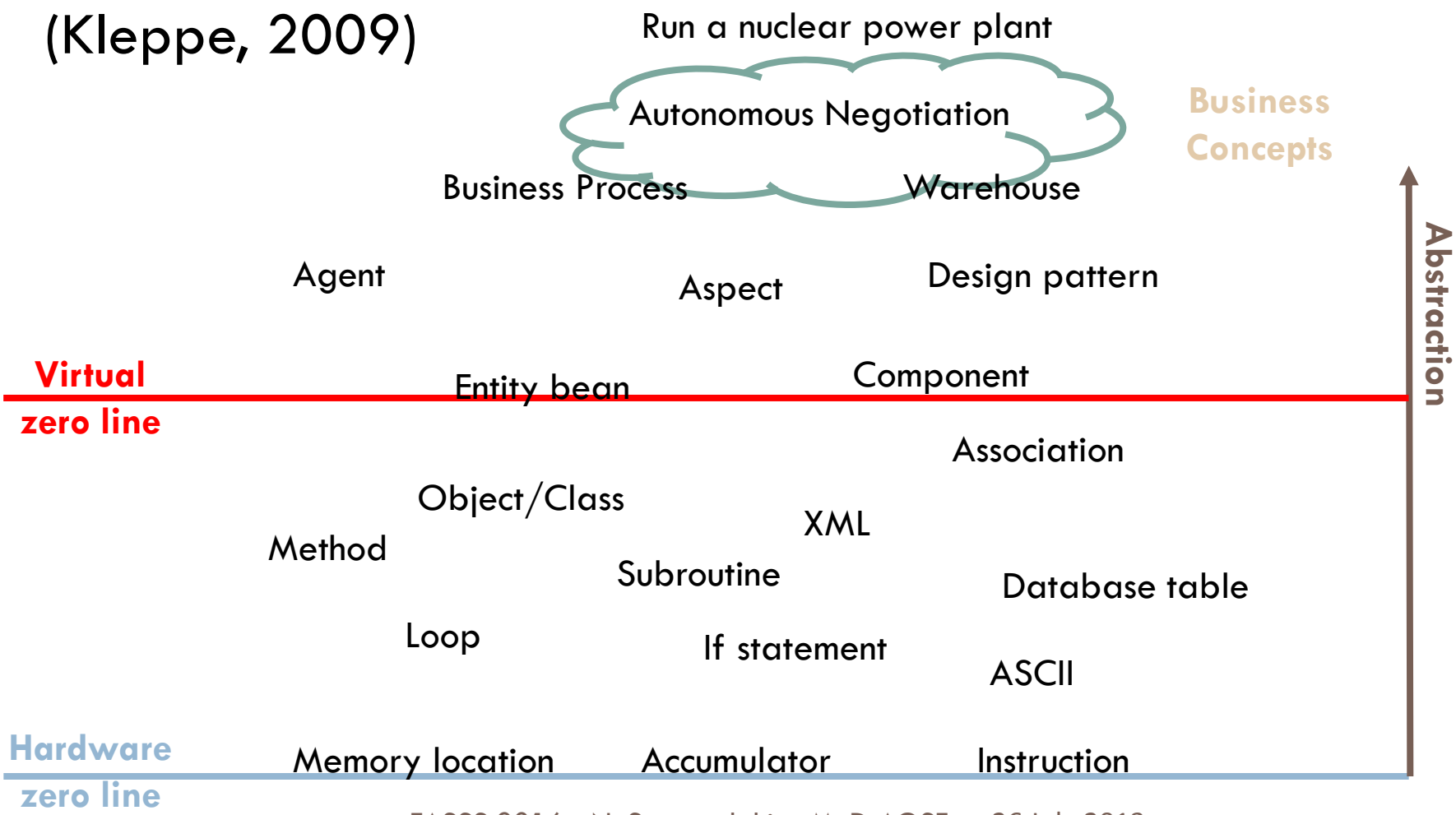


- A **software process** provides answers to the following questions (Boehn, 1988):
  - "What shall we do next?
  - How long shall we continue to do it?"
- A **software methodology** is "A predefined and organized collection of techniques and a set of rules which state by whom and in what order the techniques are used" (Tolvanen, 1998)

# SE models abstractions



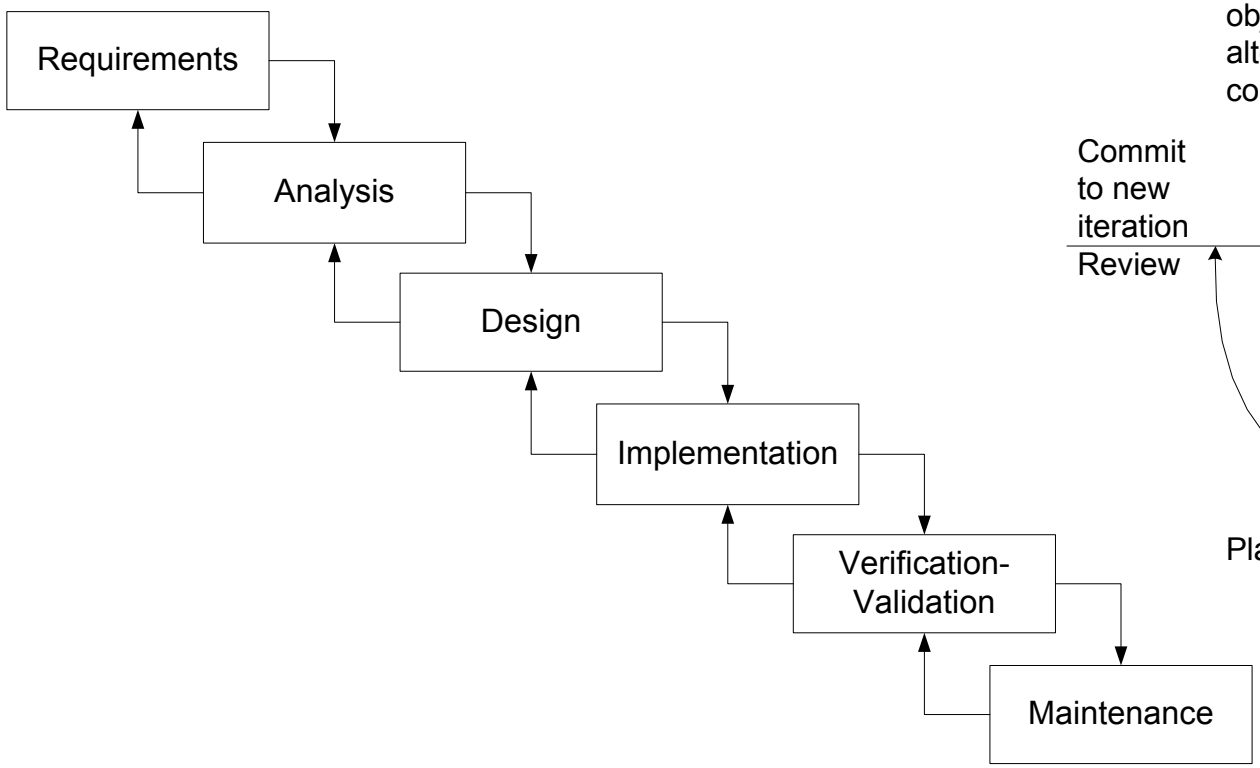
(Kleppe, 2009)

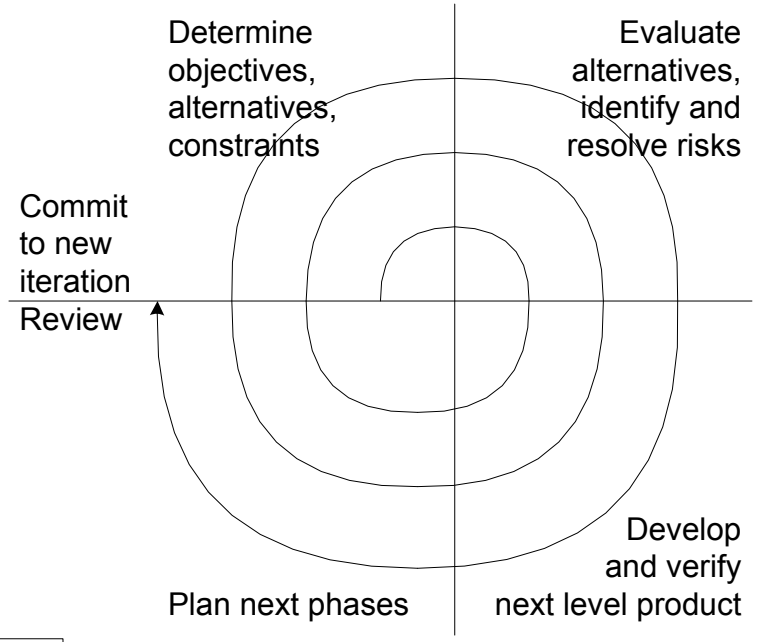


# Foundations of SE



- Waterfall and Iterative Process

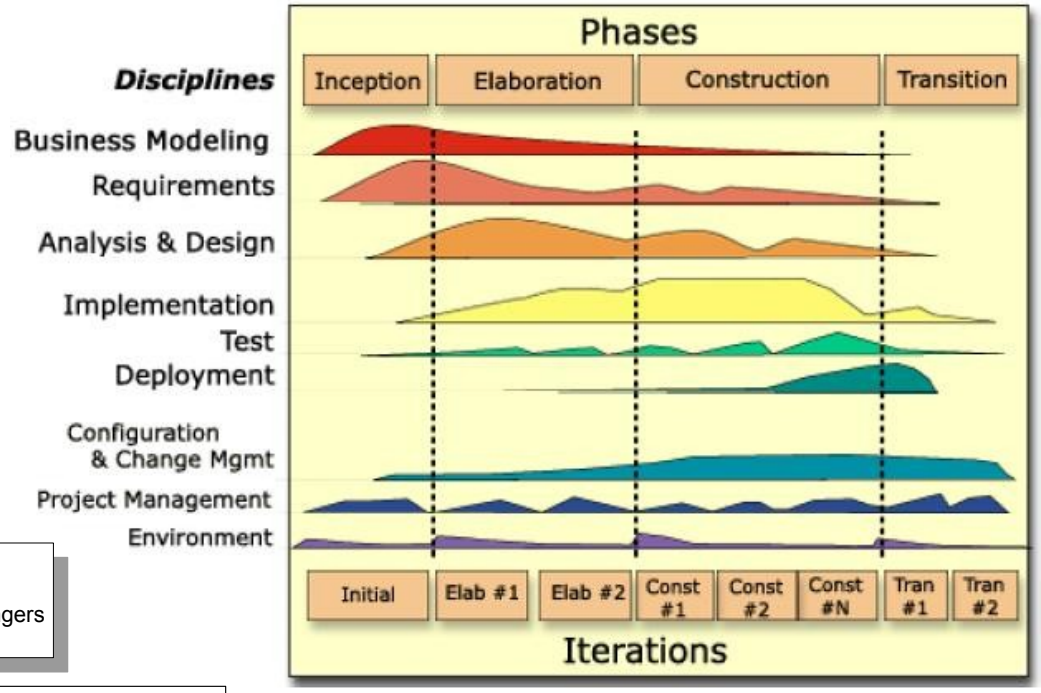


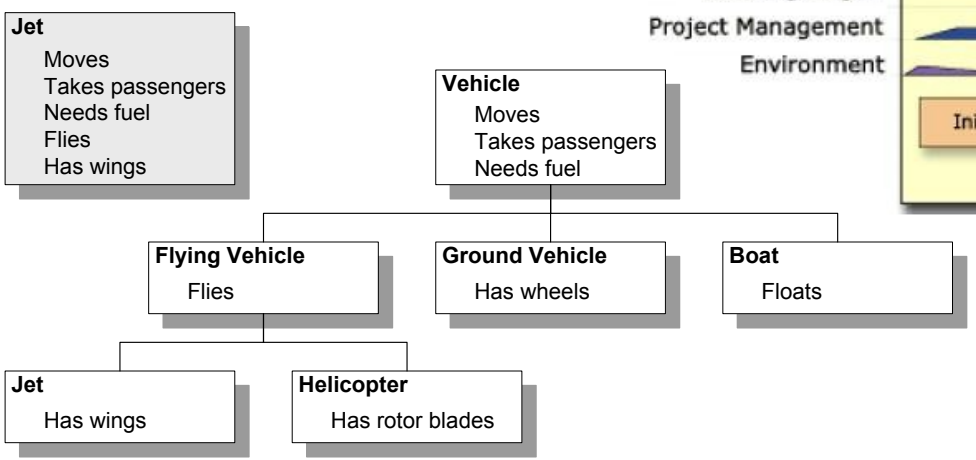



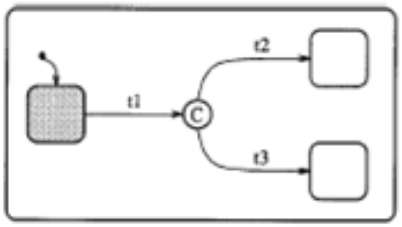
# Mainstream SE



- RUP
- UML
- Statecharts
- …



# Trends in Software Engineering



- Agility (Fowler and Highsmith, 2001)
  - New development needs
    - addressing continuously changing requirements
    - continuous evaluation
    - motivated individuals
    - less bureaucracy
  - Representatives
    - XP (Beck, 2000)
    - Scrum (Schwaber and Beedle, 2002)

# Trends in Software Engineering (2)



- Modularity
  - Give importance to interfaces
  - Heterogeneous programs interoperability
  - Assembling components
- Service Oriented Computing
  - Service description
  - Service discovery
  - Service orchestration

# Trends in Software Engineering (3)



- Autonomic Systems
  - Self-configuration: Automatic configuration of components;
  - Self-healing: Automatic discovery, and correction of faults;
  - Self-optimization: Automatic monitoring and control of resources to ensure the optimal functioning with respect to the defined requirements;
  - Self-protection: Proactive identification and protection from arbitrary attacks.
  - … (more self-* properties)

# Trends in Software Engineering (4)



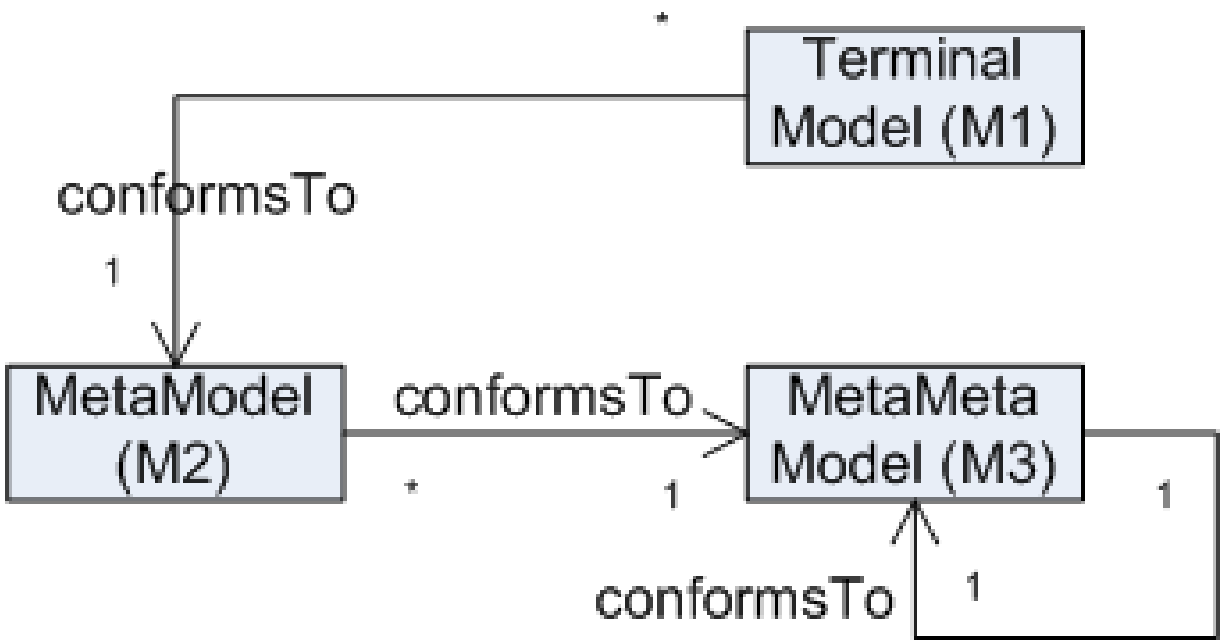
- Model-Driven Engineering (MDE)
  - Proposes the systematic use of models as primary engineering artifacts (Beydeda et al., 2005) allowing for
    - portability
    - interoperability
    - reusability
  - Model Driven Architecture (MDA) of OMG (2001)
    - Computation Independent Model (CIM)
    - Platform Independent Model (PIM)
    - Platform Specific Model (PSM)
  - Domain-Specific Languages (DSLs) and Metamodeling


Data
MOF
Querying Service
analysis
Visualisation Systematic
i-DSML
Domain
Verification
Transformation
Variability Identifying Quality
Heterogeneous Support
Lässig
flexible
Interpretation
variability Towards
Framework
Level-Agnostic
architecture
commonality
Family-based
reimplementation
Software Language-Independent
Efficient
Synchronization Elements Composition-based OCL
Languages Normalizing
DSL
Modular
Designation Traceability
Pattern-based Process
Measure
Adaptation
Alloy4SPV Description
Formal Expressive OCLR
Construction Temporal
Defining QVT
View linguistic Approach
families Generated Sensor e-Motions
Large Palladio
Extension Scale
Metrics
TGG

# Model Driven Engineering (MDE)



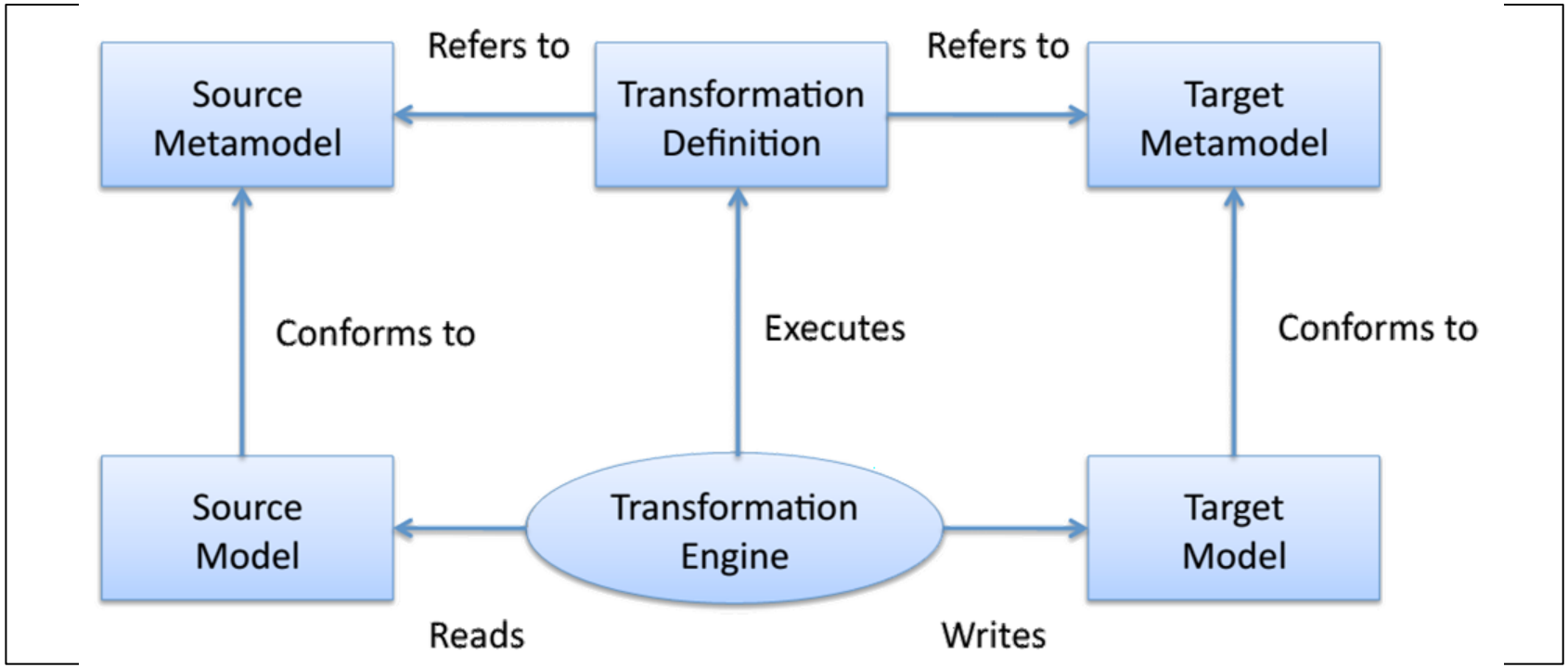
- Model driven engineering relies heavily in model transformation
- and to metamodeling (Jouault and Bézivin, 2006)



# Model Transformation



- Four types of tranformations (Langlois et al., 2007)
  - model to model (M2M)
  - model to text (M2T)
  - text to model (T2M)
  - text to text (T2T)

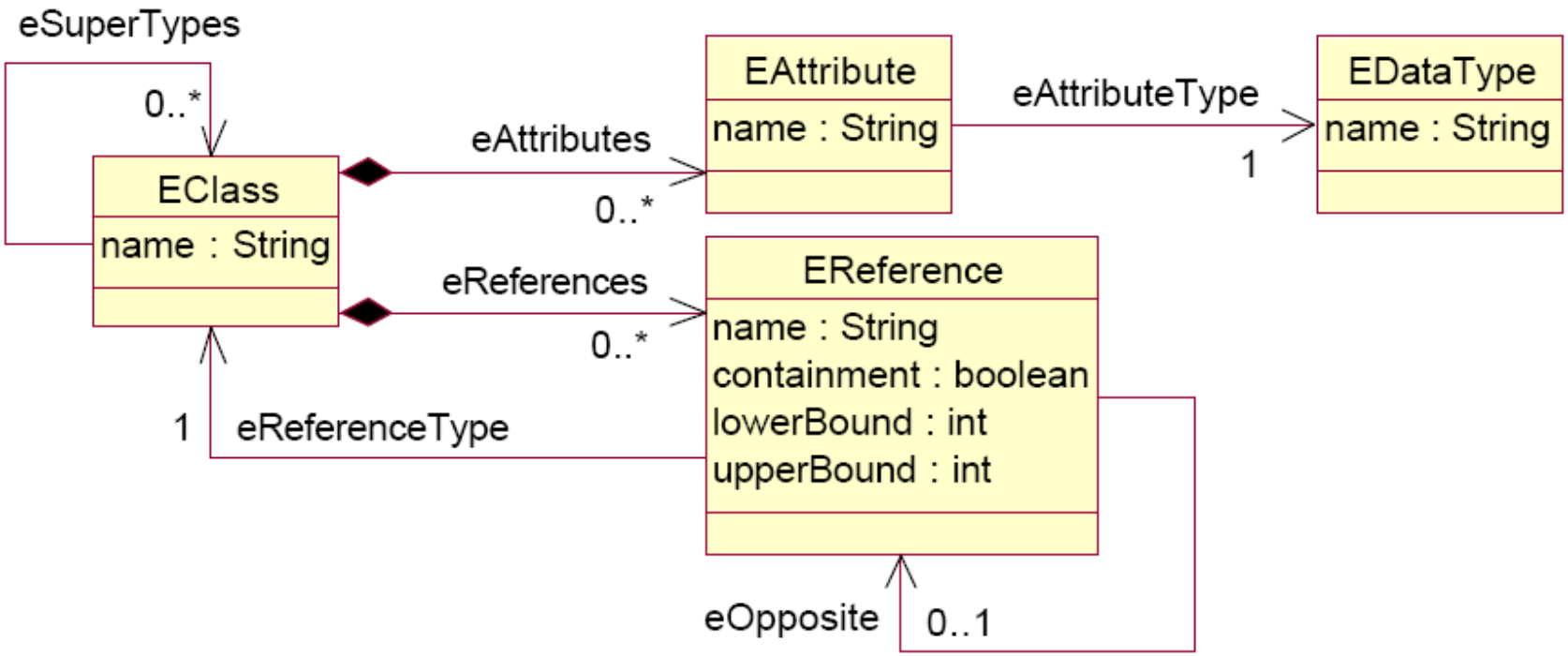


# Transforamtion requirement: The metamodel

- The metamodel allows
  - To write formally a transformation process
  - To use the Eclipse Modeling Framework which automates a number of tasks, and,
  - allows the use Java (and other languages - DSLs) for manipulating models and for writing transformations

# The ecore metametamodel

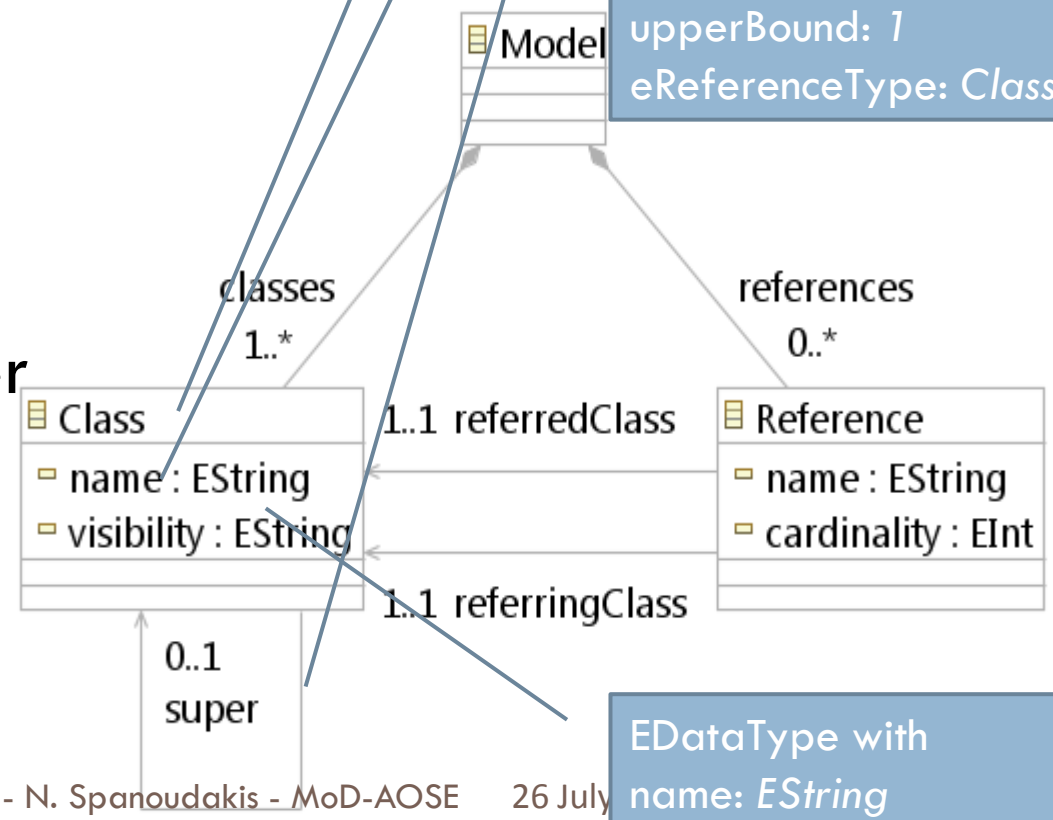
- of Eclipse Modeling Framework (EMF)



# Sample metamodel



- A class diagram contains one or more classes
- Classes have a name and visibility
- Moreover, classes can reference other classes
- Finally, they can have a super class




EClass with
name: *Class*

EAttribute with
name: *name*

EReference with
name: *super*,
containment: *false*,
lowerBound: *0*,
upperBound: *1*
eReferenceType: *Class*

EDataType with
name: *EString*

# Sample model (graphical editor)

Modeling - ClassDiagram/model/ClassDiagramModel.xmi - Eclipse

File Edit Navigate Search Project Sample Reflective Editor Run Window Help

Quick Access | Java | Modeling

Model Explorer

type filter text

- AUML_AIP
- ClassDiagram
  - Project Dependencies
  - src
  - JRE System Library [JavaSE-1.8]
  - Plug-in Dependencies
  - META-INF
  - model
    - classDiagram.aird
    - classDiagram.ecore
    - classDiagram.ecore_diagran
    - classDiagram.genmodel
    - ClassDiagramModel.xmi
- Prosumer

classDiagram... | ClassDiagram...

- platform:/resource/ClassDiagram/model/ClassDiagramModel.xr
  - Model
    - Reference follows
    - Class Vehicle
    - Class Car
    - Class PoliceCar
    - Class Motorbike
- platform:/resource/ClassDiagram/model/classDiagram.ecore

Properties | Problems

| Property | Value |
|---|---|
| Cardinality | 1 |
| Name | follows |
| Referred Class | Class Vehicle |
| Referring Class | Class PoliceCar |

Selected Object: Reference follows

Session saving

# Sample model (text-XML editor)



C:\Users\Nikos\My Programs\eclipse-modeling-mars-2\workspace\ClassDiagram\model\ClassDiagramModel.xmi - Notepad++

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

ClassDiagramModel.xmi

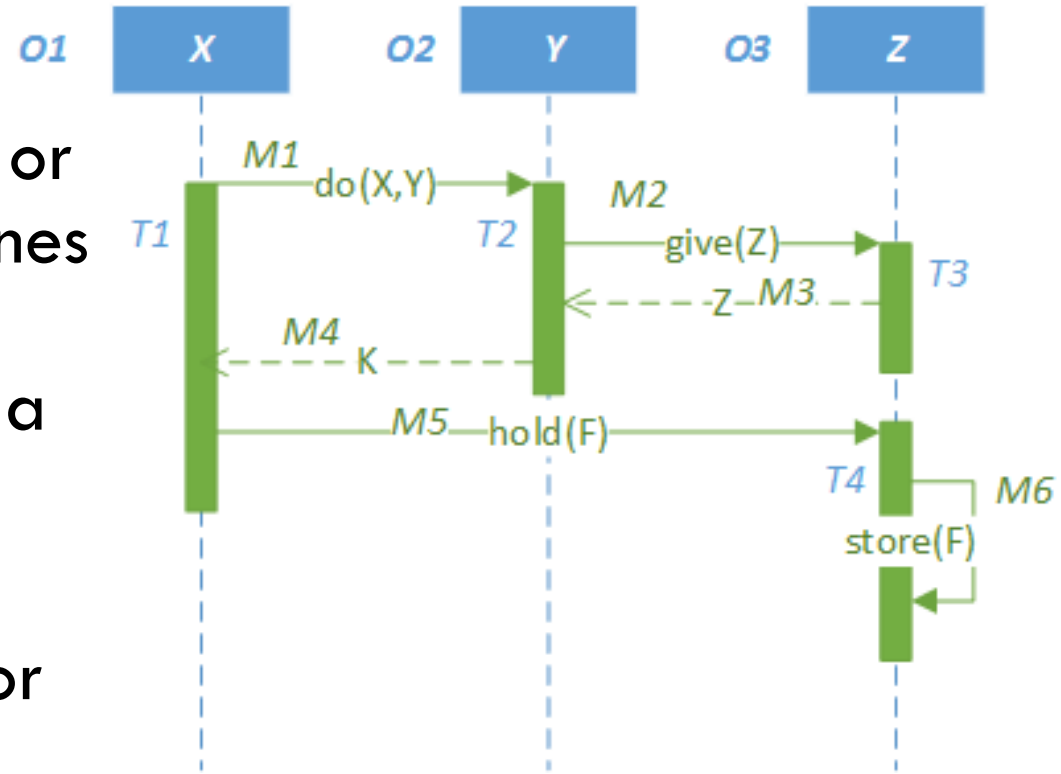
```xml
<?xml version="1.0" encoding="UTF-8"?>
<classDiagram:Model
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:classDiagram="http://www.example.org/classDiagram"
    xsi:schemaLocation="http://www.example.org/classDiagram classDiagram.ecore">
  <references
      referringClass="//@classes.2"
      referredClass="//@classes.0"
      name="follows"
      cardinality="1"/>
  <classes name="Vehicle"/>
  <classes name="Car"
      super="//@classes.0"/>
  <classes name="PoliceCar"
      super="//@classes.1"/>
  <classes name="Motorbike"
      super="//@classes.0"/>
</classDiagram:Model>
```

eXtensible M length : 685 lines : 21 Ln : 7 Col : 81 Sel : 0 | 0 Dos\Windows UTF-8 INS

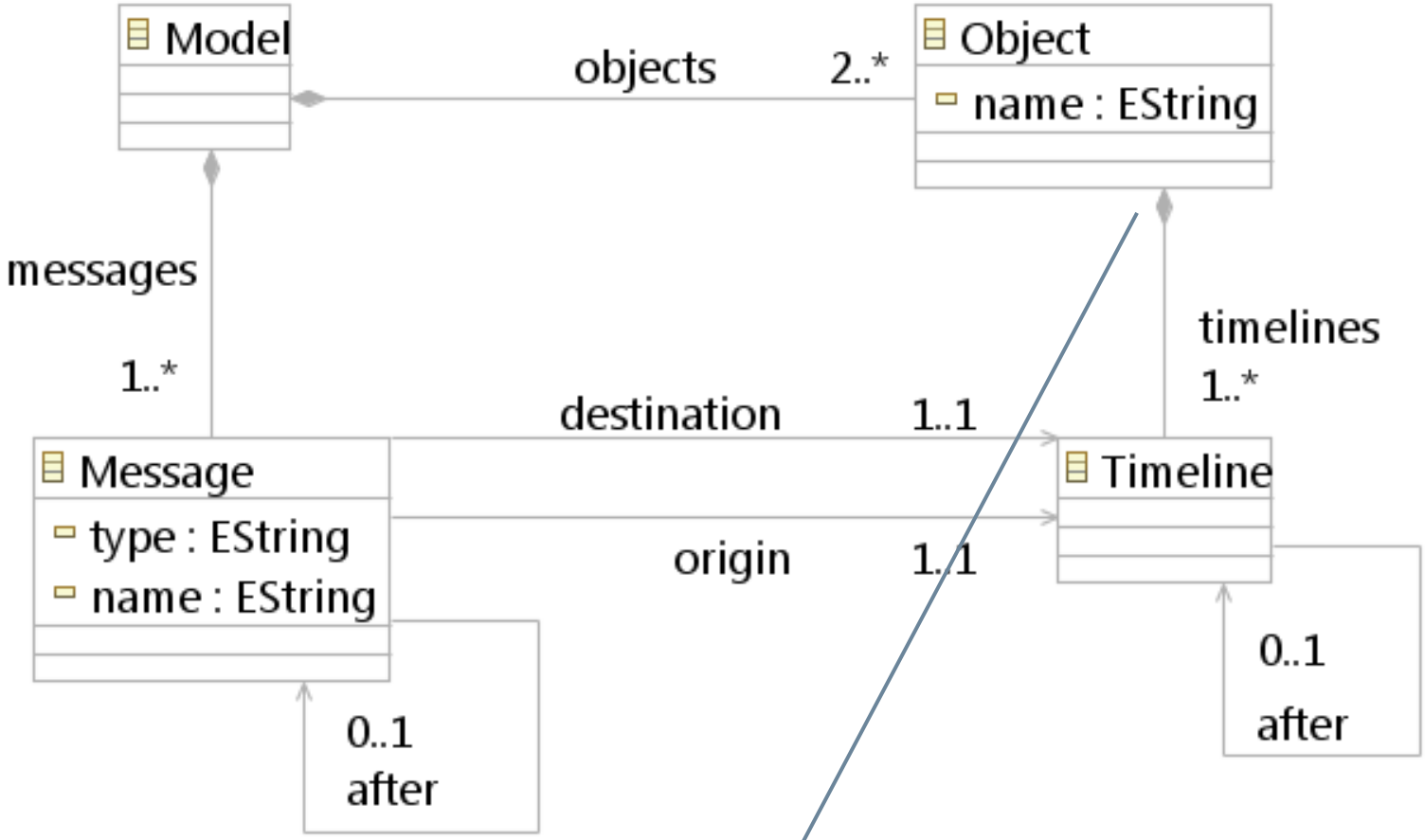
# Now, let's create a metamodel



- Sequence diagrams are common in SE
- Each object has one or more ordered timelines and can send and receive messages in a timed sequence
- Messages can be method invocations or return messages



# A Sequence Diagram metamodel




All classes must be contained by another except the Model class

# A UML sequence diagram model



Modeling - AUML_AIP/model/UML_SD_Model.xmi - Eclipse

File Edit Navigate Search Project Sample Reflective Editor Run Window Help

Quick Access | Java | Modeling

Model Explorer

type filter text

- AUML_AIP
  - Project Dependencies
  - src
  - JRE System Library [JavaSE-1.8]
  - Plug-in Dependencies
  - META-INF
  - model
    - AUML_AIP.aird
    - AUML_AIP.ecore
    - AUML_AIP.ecore_diagram
    - AUML_AIP.genmodel
    - UML_SD_Model.xmi
    - UML_SD.ecore
      - AUML_AIP
    - UML_SD.ecore_diagram

Outline

An outline is not available.

UML_SD.ecore_diagram | UML_SD_Model.xmi

- platform:/resource/AUML_AIP/model/UML_SD_Model.xmi
  - Model
    - Object X
      - Timeline
    - Object Y
    - Object Z
    - Message do(X,Y)
    - Message give(Z)
    - Message Z
    - Message K
    - Message hold(F)
    - Message store(F)
- platform:/resource/AUML_AIP/model/UML_SD.ecore

Properties | Problems

| Property | Value |
|---|---|
| After | Message K |
| Destination | Timeline |
| Name | hold(F) |
| Origin | Timeline |
| Type | call |

Selected Object: Message hold(F)

# XML

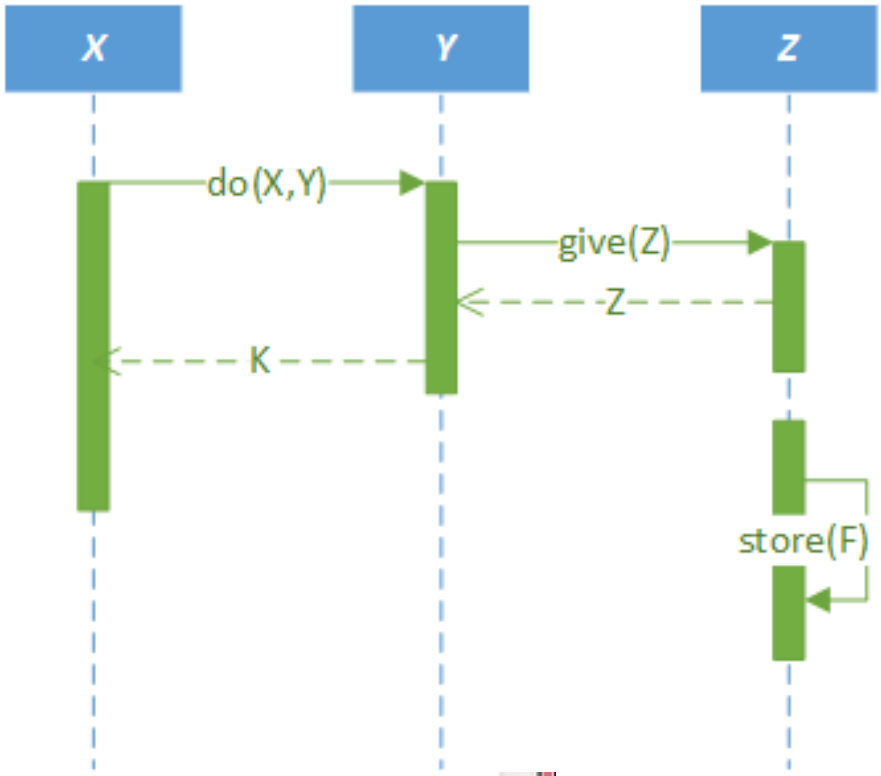
C:\Users\Nikos\My Programs\eclipse-modeling-mars-2\workspace\AUML_AIP\model\UML_SD_Model.xmi - Notepa...

File Edit Search View Encoding Language Settings Macro Run Plugins Window ?

UML_SD_Model.xmi

```xml
<?xml version="1.0" encoding="UTF-8"?>
<AUML_AIP:Model
    xmi:version="2.0"
    xmlns:xmi="http://www.omg.org/XMI"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:AUML_AIP="http://www.example.org/AUML_AIP"
    xsi:schemaLocation="http://www.example.org/AUML_AIP UML_SD.ecore">
  <objects name="X">
    <timelines/>
  </objects>
  <objects name="Y">
    <timelines/>
  </objects>
  <objects name="Z">
    <timelines/>
    <timelines
        after="//@objects.2/@timelines.0"/>
  </objects>
  <messages
      origin="//@objects.0/@timelines.0"
      destination="//@objects.1/@timelines.0"
      type="call"
      name="do(X,Y)"/>
  <messages
      origin="//@objects.1/@timelines.0"
      destination="//@objects.2/@timelines.0"
      type="call"
      name="give(Z)"
```

length : 1555 lines : 55 Ln : 7 Col : 71 Sel : 0 | 0 Dos\Windows UTF-8 INS



# Domain Specific Languages (DSLs)



- A DSL is a computer language specialized to a particular application domain
- The line between general-purpose languages and DSLs is not always sharp, e.g.:
  - Parallel programming language (Ada)
  - Query language SQL
  - 3D models representation (CAD)
- We use EMF and metamodels for defining DSLs

# Computer-Aided Software Engineering



- Computer-Aided Software Engineering (CASE) tools improve productivity and quality in software development
  (Baik and Boehm, 2000)
- Tools aid the software development process and lead to code generation
- Model transformation prevents errors of understanding or copying

# Knowledge Engineering



- Is about
  - Knowledge Representation
  - Inference Engines and Tools
- Trends
  - Ontology
    - Development
    - Merging
    - Sharing
  - Web (OWL, RDF)

# Agent Oriented Software Engineering

I have the knowledge on how to program an agent (method/know-how, programming language, etc). Look, here is my ground-breaking argumentation-based negotiating agent…

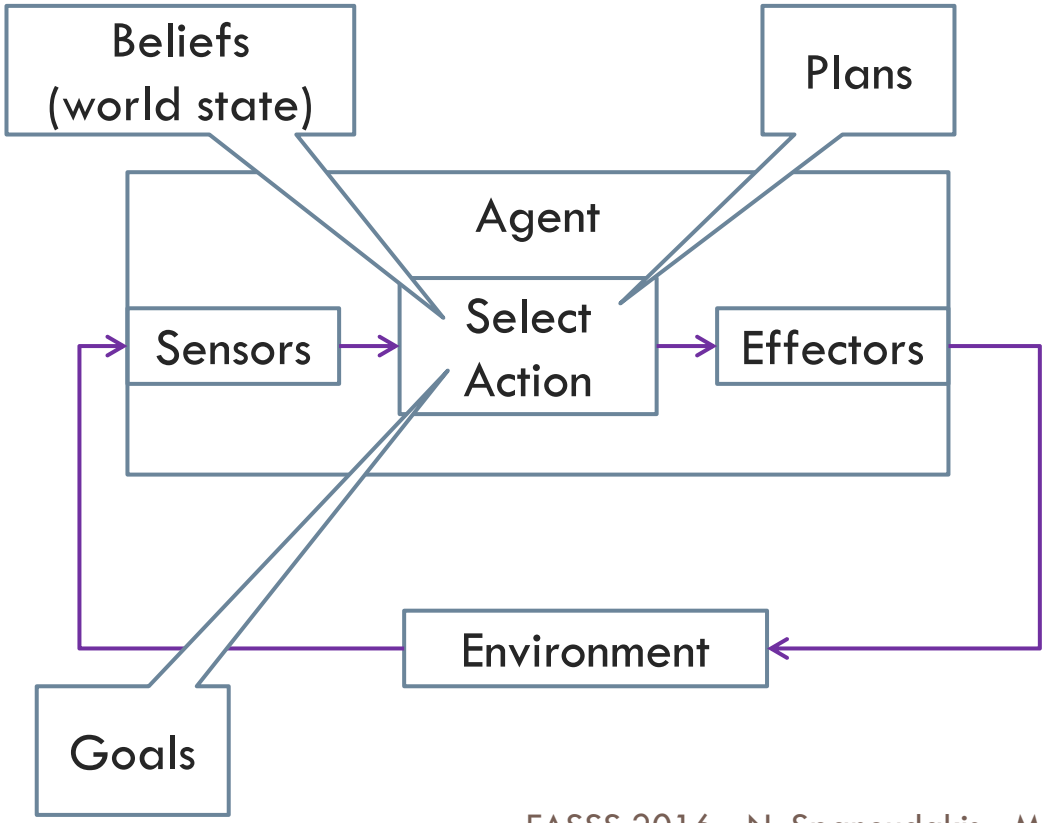
Why do I need AOSE?

What is it about?

# Agents vs Objects



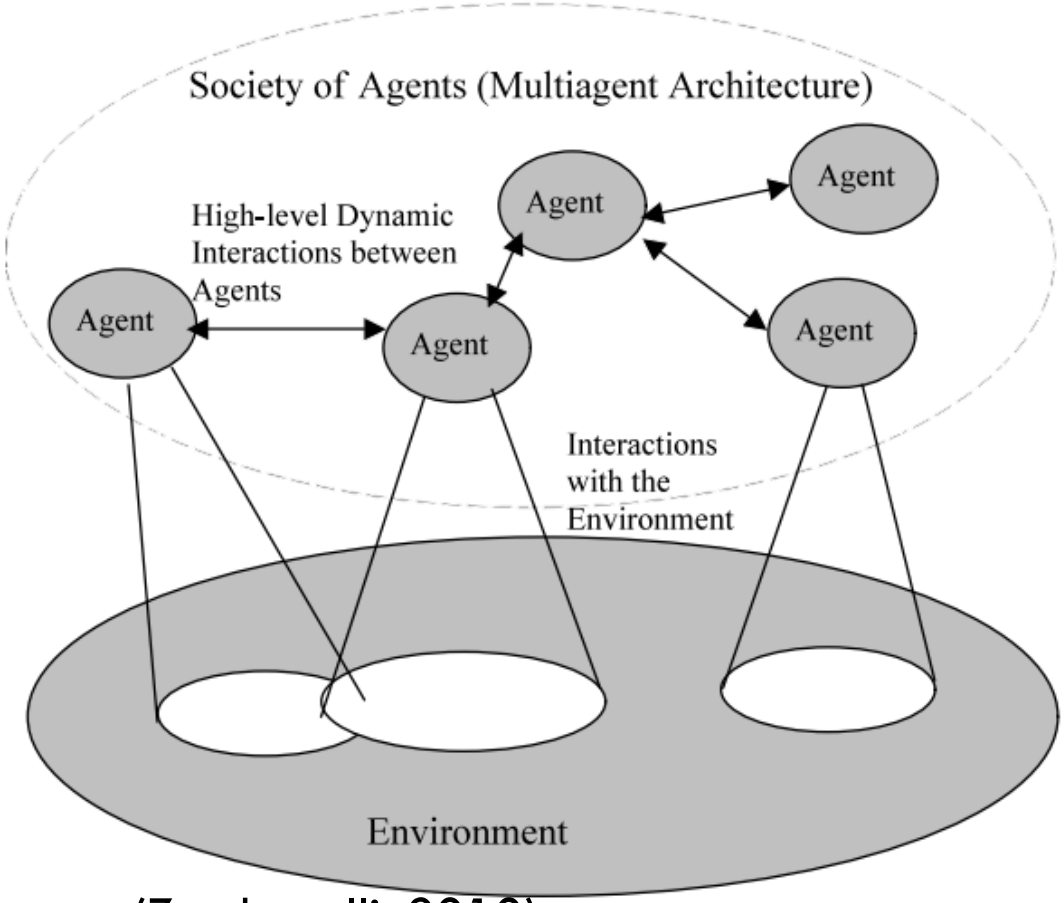
Agent

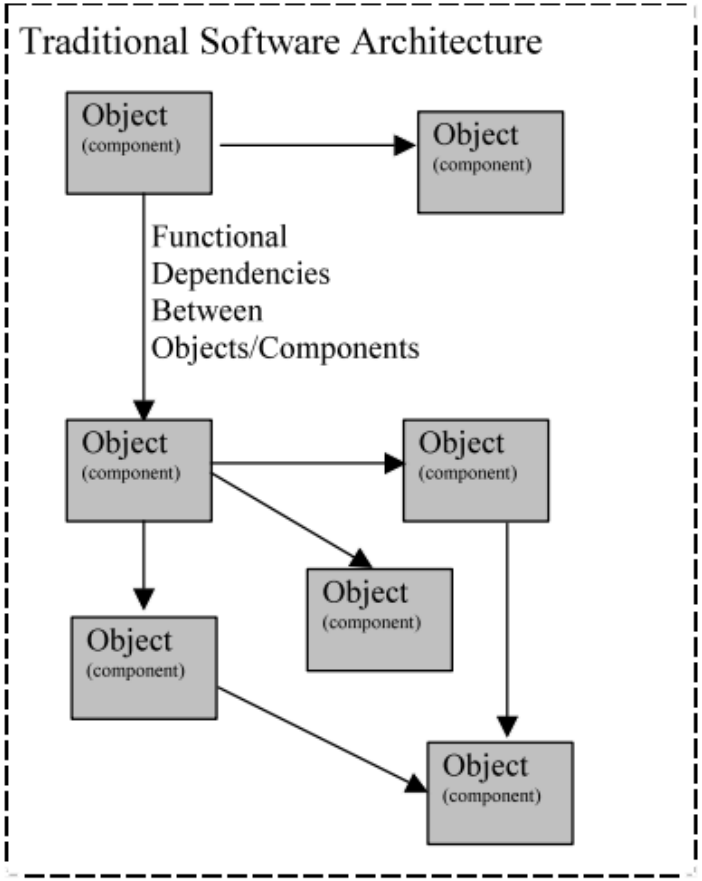



Class

# Agents vs Objects (2)






(Zambonelli, 2010)

# Agent Oriented Programming (AOP)



- Events
  - From the environment
  - From peers
- Goals
  - Select next goal
- Plans
  - Select plan
  - Revise plan
- Beliefs
  - Environment state
  - Belief revision
- Knowledge
  - Domain knowledge
- Actions
  - Act on environment
  - Send message

# Agent Oriented Software Engineering



- Research a methodology for defining software that is (Wooldridge and Jennings, 1995; Weiss, 2003):
  - Autonomous
  - Social
  - Reactive
  - Pro-active
  - Adaptive
  - Persistent
- The domain is maturing (more than 15 years)
  - OOD invented around 1982, first UML version in 1997
- Most important for Multi-Agent Systems (MAS)

# AOSE Considerations



- What, how many agents?
- How to structure of agent?
- Model of the environment?
- Communication?
- Relationships?
- Coordination?
- Protocols?

(Hexmoor and Brainov, 2002)

# Intrinsic Attributes (Weiss, 1999)



- Life span (temporary, permanent)
- Knowledge level (reactive, deliberative)
- Construction (declarative method, procedural method)
- Mobility (static, mobile)
- Adaptiveness (rigid, self-learning, learning)
- Models (self-model, world-model, other agents models)

# Extrinsic Attributes (Weiss, 1999)

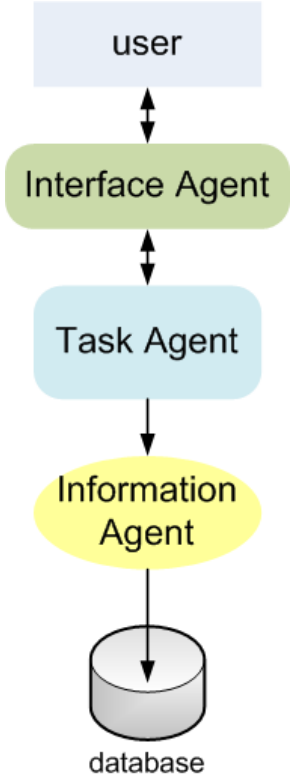


- Locality (local, remote)
- Autonomy (autonomous, controlled)
- Sociality (autistic, informed, responsible, team member)
- Friendliness (cooperative, antagonistic, truthful, benevolent)
- Interactions
  - How (directly, through facilitators or mediators or non-agents)
  - Whom (other agents, people, the world)
- Semantics (declarative or linear communications)

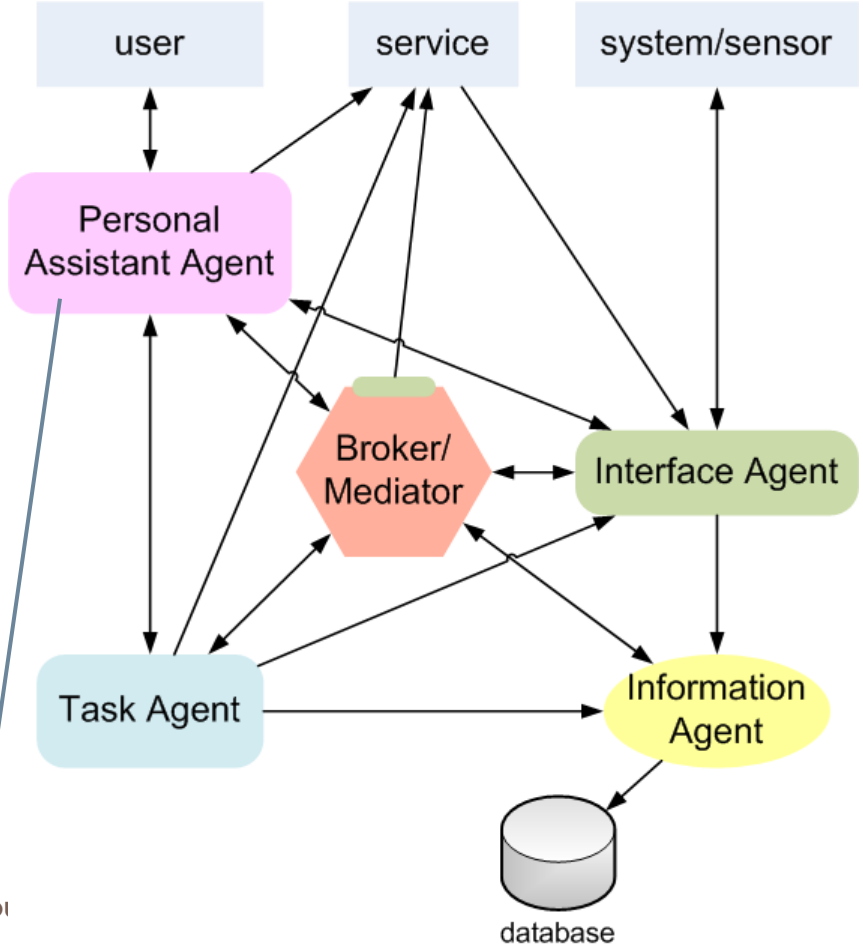
# Software Agent Types



Sycara and Zeng (1996)




And since then…




Or even Intelligent Personal Assistant (IPA)

N. Spanoı

# Platforms



- 1990: AGENT-0 (Shoham)
- 1993: PLACA (Thomas; AGENT-0 extension with plans)
- 1996: AgentSpeak(L) (Rao; inspired by PRS)
- 1997: 3APL (Hindriks et al.)
- 2000: JACK (Busetta, Howden, Ronnquist, Hodgson)
- 2000: CLAIM (Amal El FallahSeghrouchni)
- 2001: JADE (Bellifemine, Poggi, Rimasa)
- 2002: Jason (Bordini, Hubner; implementation of AgentSpeak)
- 2003: Jadex (Braubach, Pokahr, Lamersdorf)
- 2008: 2APL (successor of 3APL)

(extended from Koen Hindriks, 2011)

# Some AOSE Methodologies



- Multi-agent Systems Engineering (Deloach et al., 2001, 2006)
- The Gaia Methodology (Wooldridge et al., 2000 ; Zambonelli et al., 2003)
- Agent UML (Odell et al., 2000)
- Vowels (Ricordel and Demazeau, 2002)
- PASSI (Burrafato and Cossentino, 2002)
- Prometheus (Padgham and Winikoff, 2003)
- Adelfe (Bernon et al., 2003)
- Ingenias (Pavón and Gómez-Sanz, 2003)
- Tropos (Bresciani et al., 2004)
- ASEME (Spanoudakis and Moraitis, 2009)
- Malaca, Gormas, ….

# AOSE Methodologies

Why are there so many?

Which one should I use?

OOD had a number of proposals before UML became a standard

# MaSE

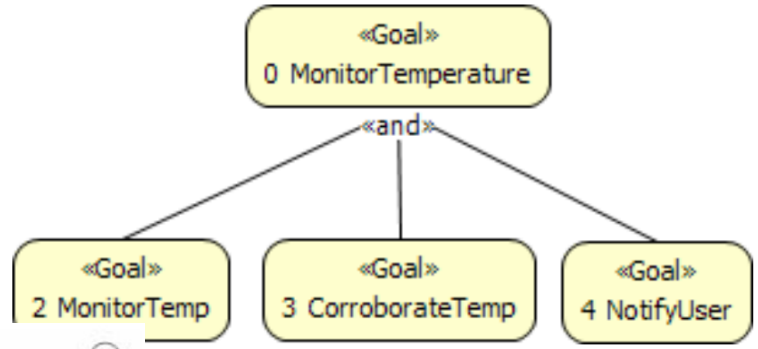


- A UML-based Methodology
- MaSE defines a system goal oriented MAS development methodology
- The authors define for the first time inter and intra-agent interactions that must be integrated
- O-MaSE (Deloach, 2005) introduced the organization concept, introducing the use of AUML in MaSE

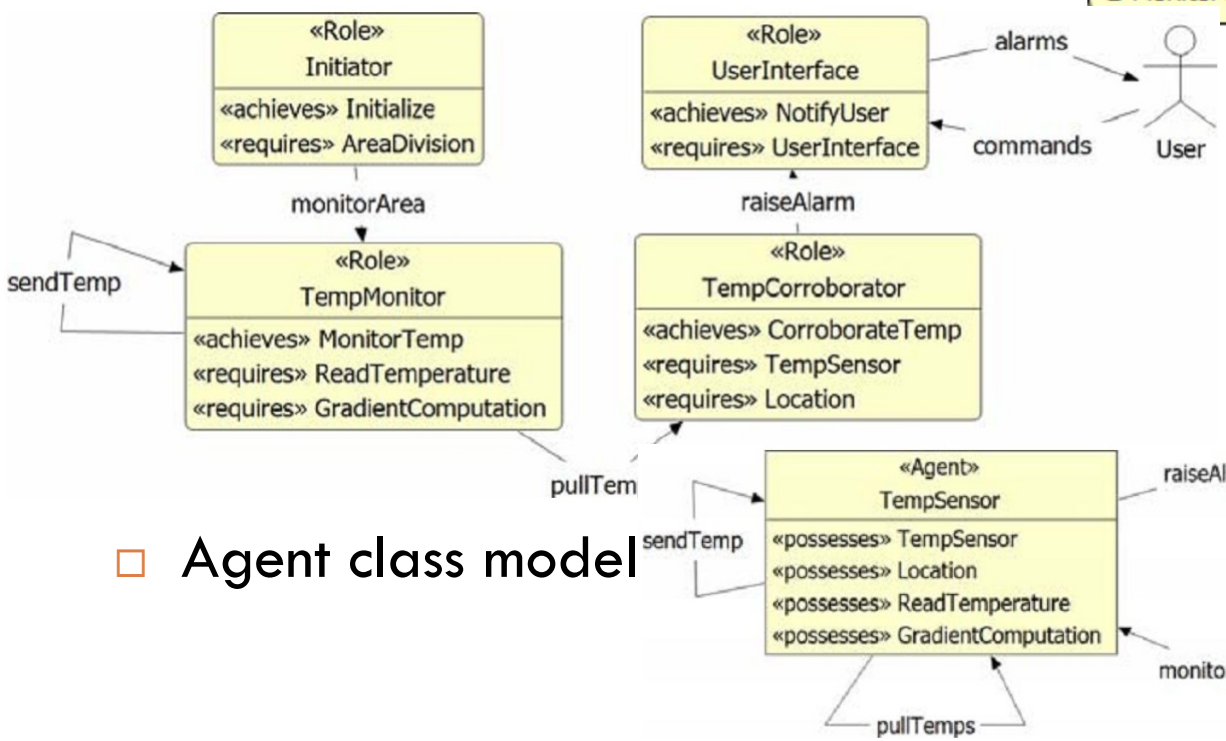
# Goal-based role definition



- Goal Hierarchy Diagram




- Role model

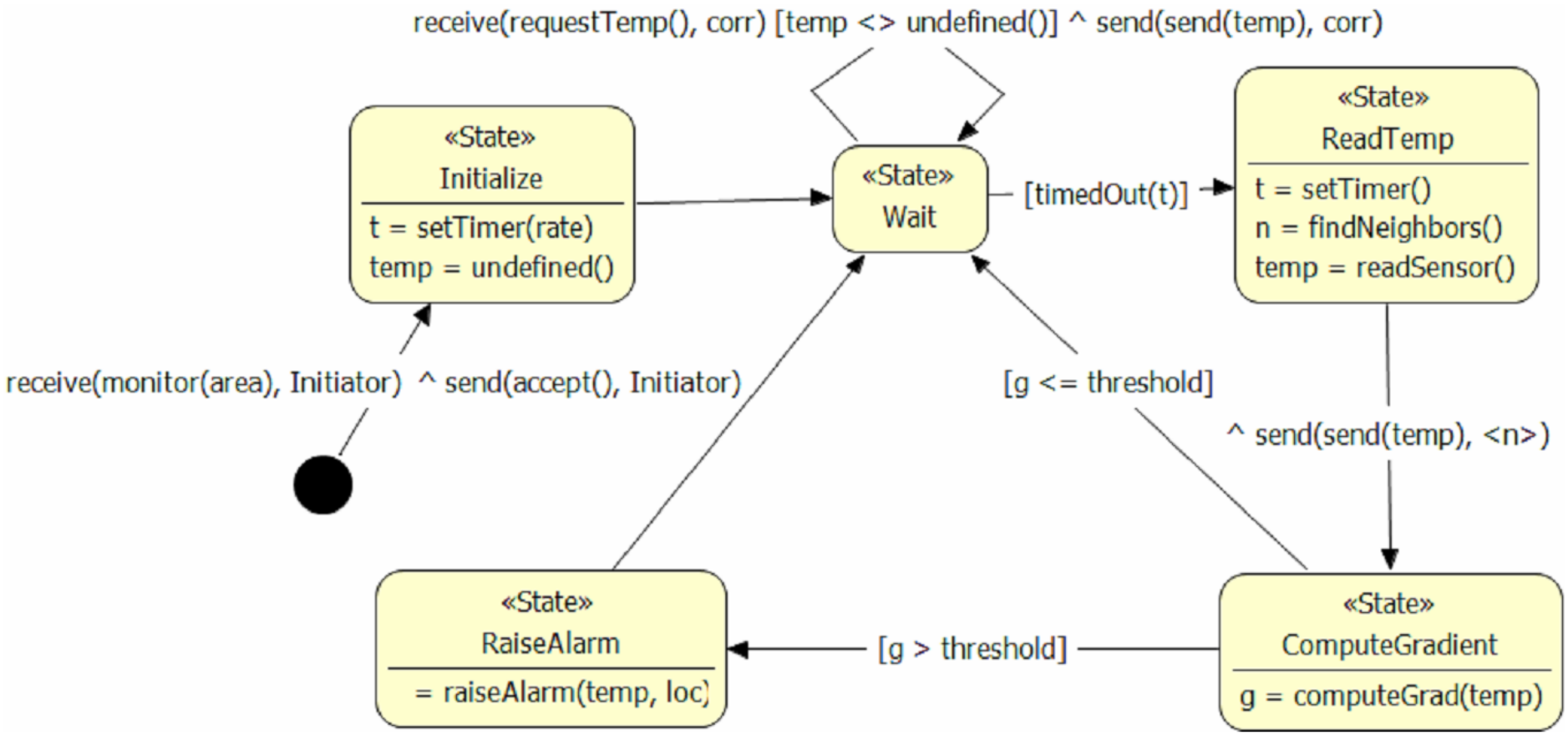



- Agent class model

# Plan Model



- Get Temperature plan



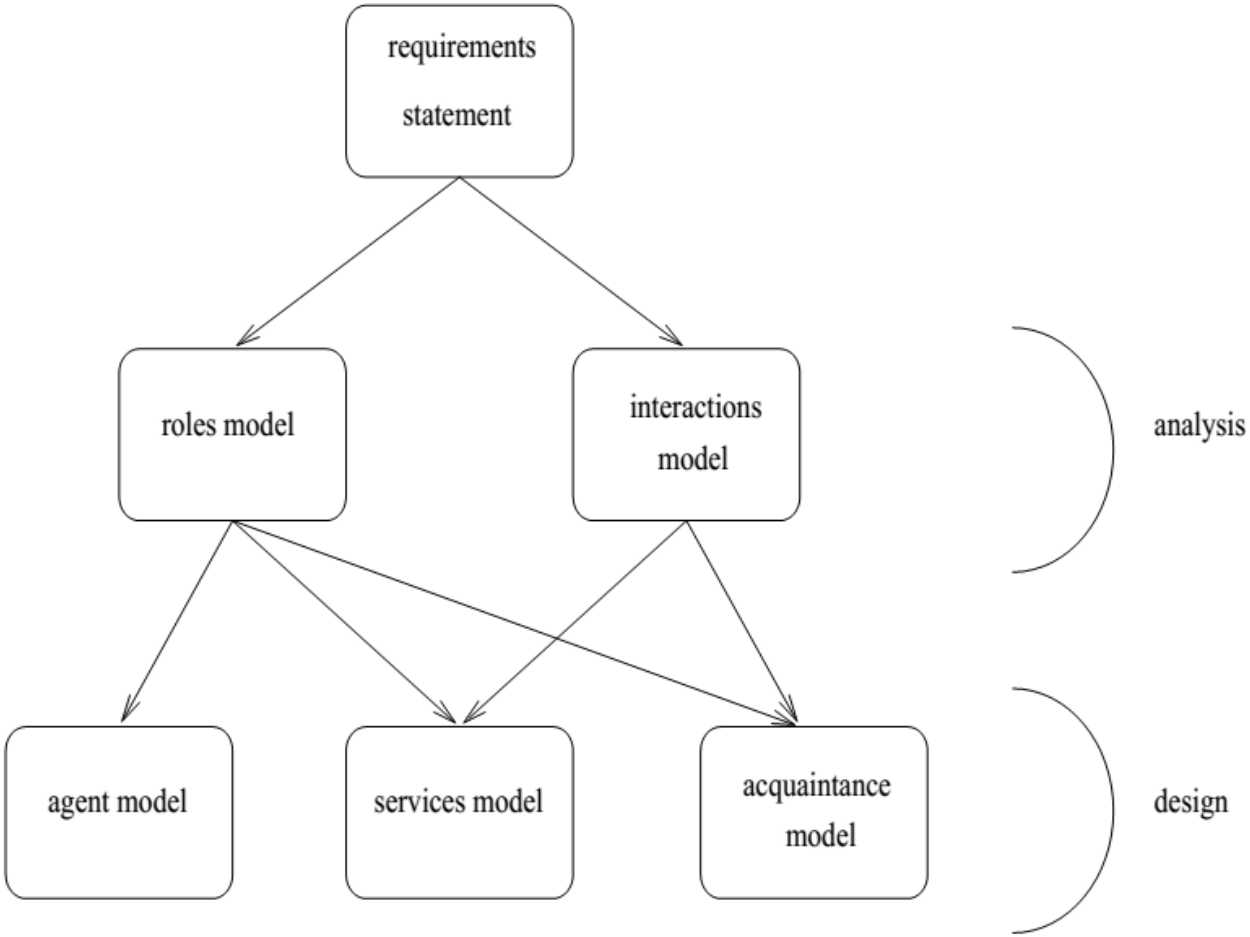
# The Gaia Methodology



- The Gaia methodology (Wooldridge et al., 2000) is an attempt to define a general methodology that it is specifically tailored to the analysis and design of MAS.
- Gaia defines the structure of a MAS in terms of a role model.
- The model identifies the *roles* that agents have to play within the MAS and the interaction protocols between the different roles.

# The Gaia process





# The Gaia Analysis phase



- Roles consist of:
- *responsibilities,*
  - Liveness
  - Safety
- *permissions,*
- *Activities,* and
- *protocols.*

| Operator | Interpretation |
|---|---|
| **x . y** | x followed by y |
| **x \| y** | x or y occurs |
| **x*** | x occurs 0 or more times |
| **x+** | x occurs 1 or more times |
| **x ω** | x occurs infinitely often |
| **[x]** | x is optional |
| **x \|\| y** | x and y interleaved |

Role Schema: CUSTOMER (CUST)

Description:
Organisation or individual requiring a service quote.

Protocols and Activities:
MakeCall, GiveRequirements

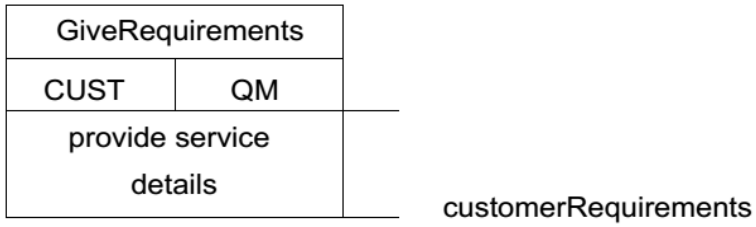
Permissions:

```
generates  customerDetails        // Owner of customer information
           customerRequirements   // Owner of customer requirements
```

Responsibilities
Liveness:

CUSTOMER = ( MakeCall. GiveRequirements)+

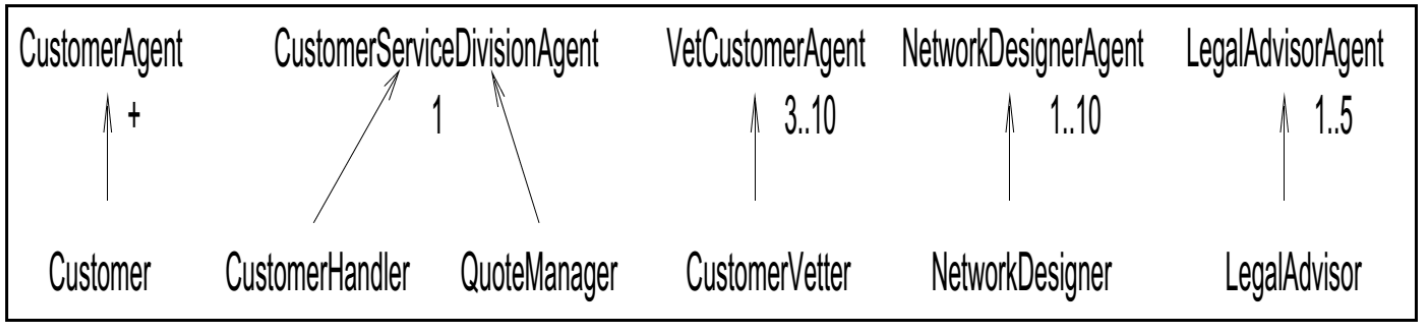
Safety:
- **true**



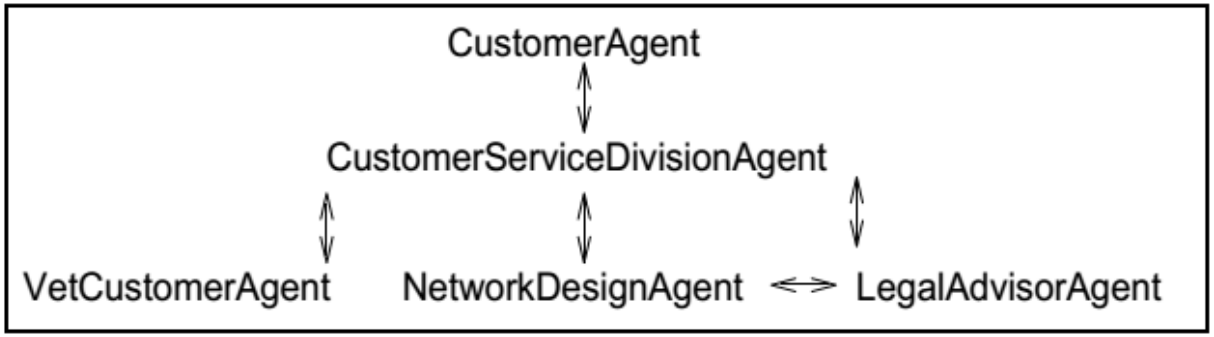
# The Gaia design phase



- Agent types aggregate roles




- … and have acquaintances



# Extensions



- The Gaia2JADE process (Moraitis and Spanoudakis, 2006) proposed a method for generating code for the JADE framework.
- Another work (Bittencourt et al., 2009) automated this selection using semantic web technologies but stopped at defining each Gaia activity as a generic JADE Behaviour
- ROADMAP proposed an hierarchy of roles for integrating roles
- Zambonelli et al. (2003) proposed
  - the environmental model
  - organizational structure with institutional rules

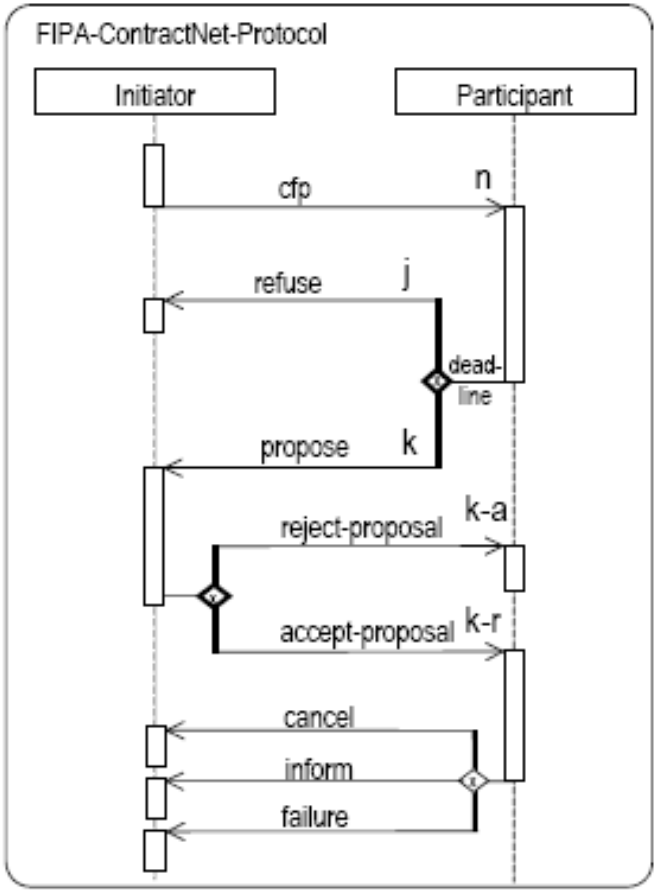
# Agent UML (AUML)



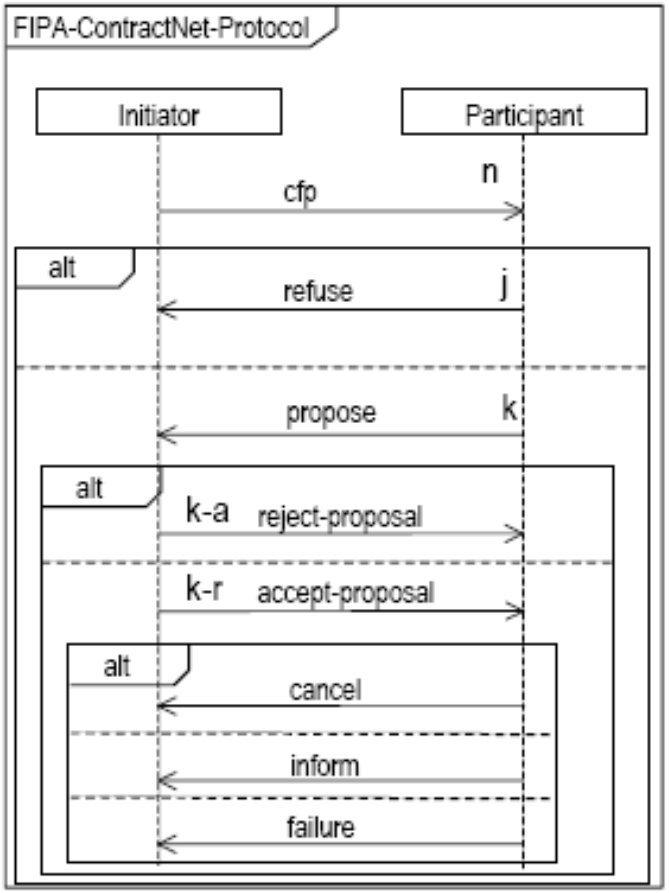
- Is an extension of UML to capture agent technology concepts such as protocol, role, etc
- MAS development can be a top-down decomposition process starting from the roles and protocols (Odell et al., 2001)
- The Agent Interaction Protocol (AIP) model is used by a number of methodologies and practitioners becoming de-facto standard

# Agent Interaction Protocol (AIP)





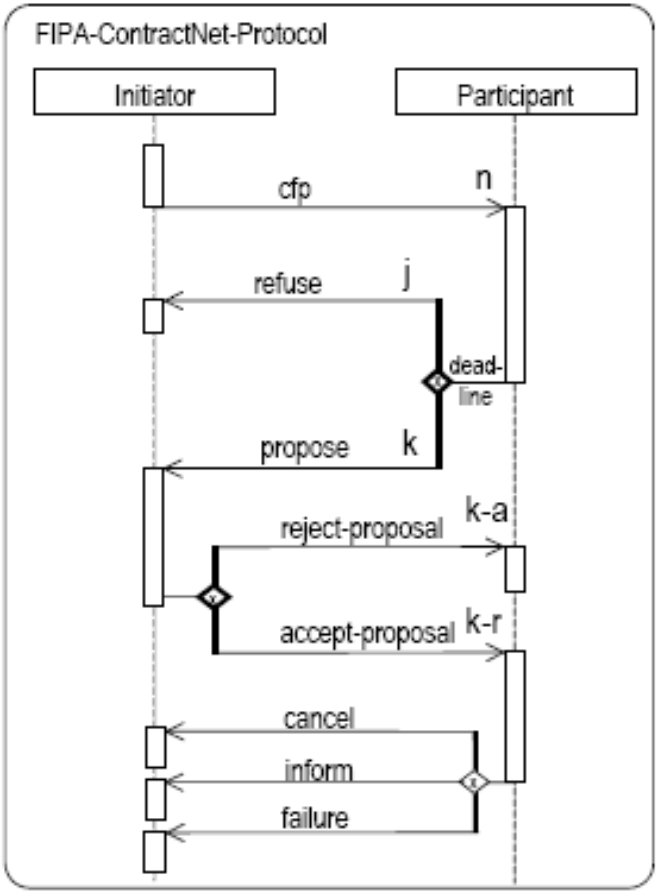

(a) UML 1.x with extensions




(b) UML 2.0

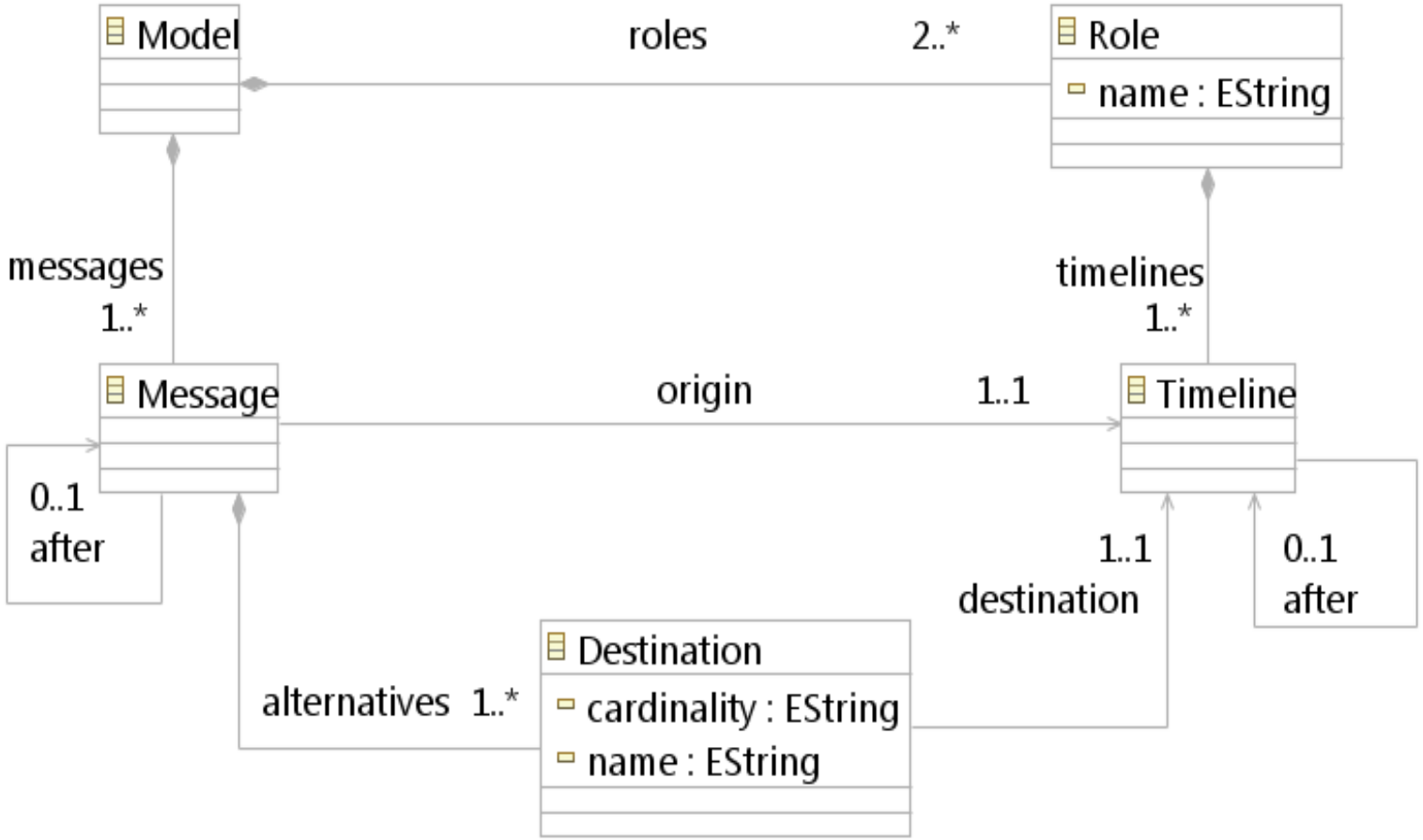
# A metamodel of AIP






(a) UML 1.x with extensions

- Let's extend the Sequence diagram metamodel:
  - A message can have cardinality
  - A message can have alternative paths and names
  - We have roles instead of objects

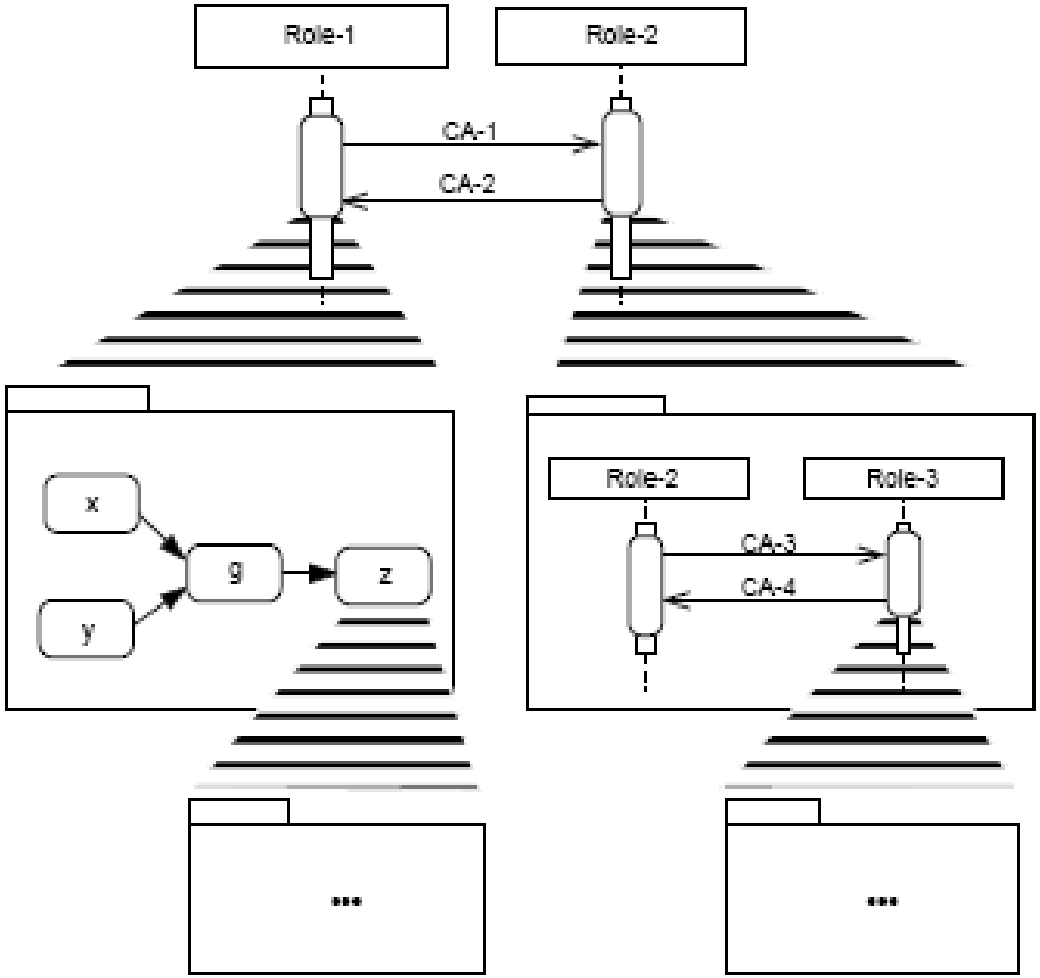





# An AIP metamodel

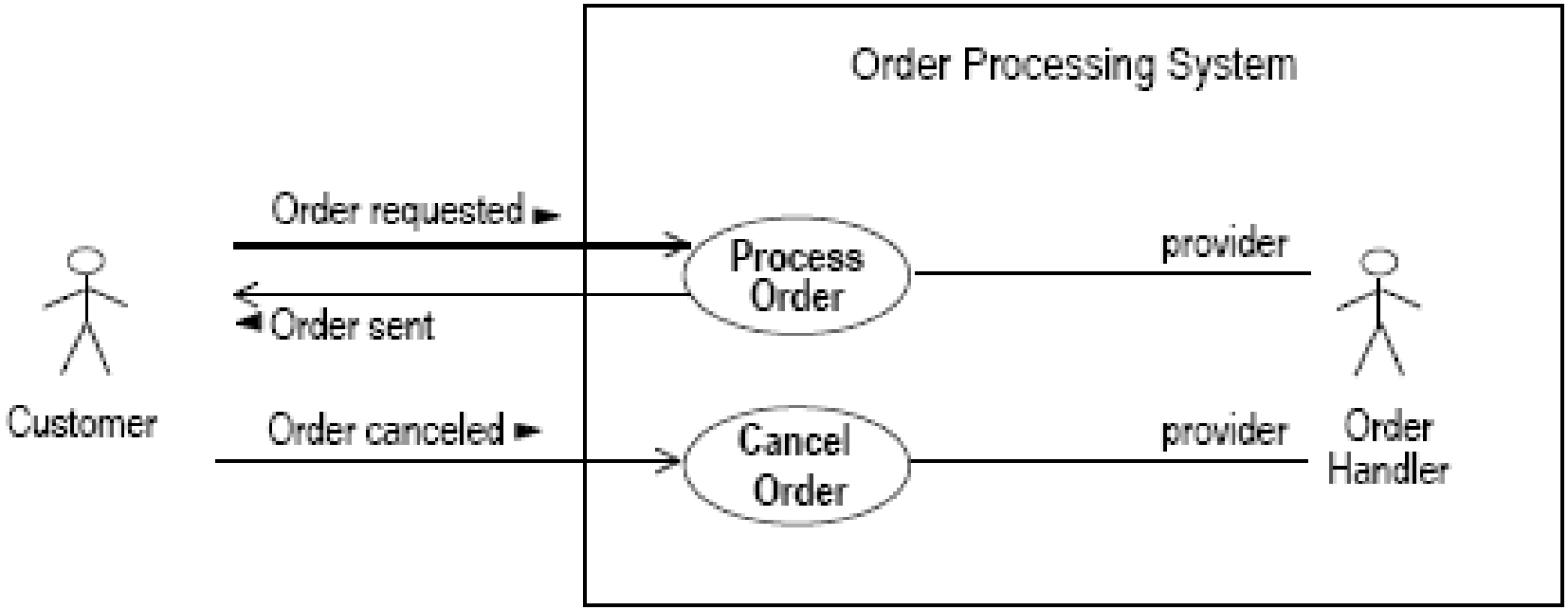
# Top-down decomposition process





# AUML Use Case Diagram





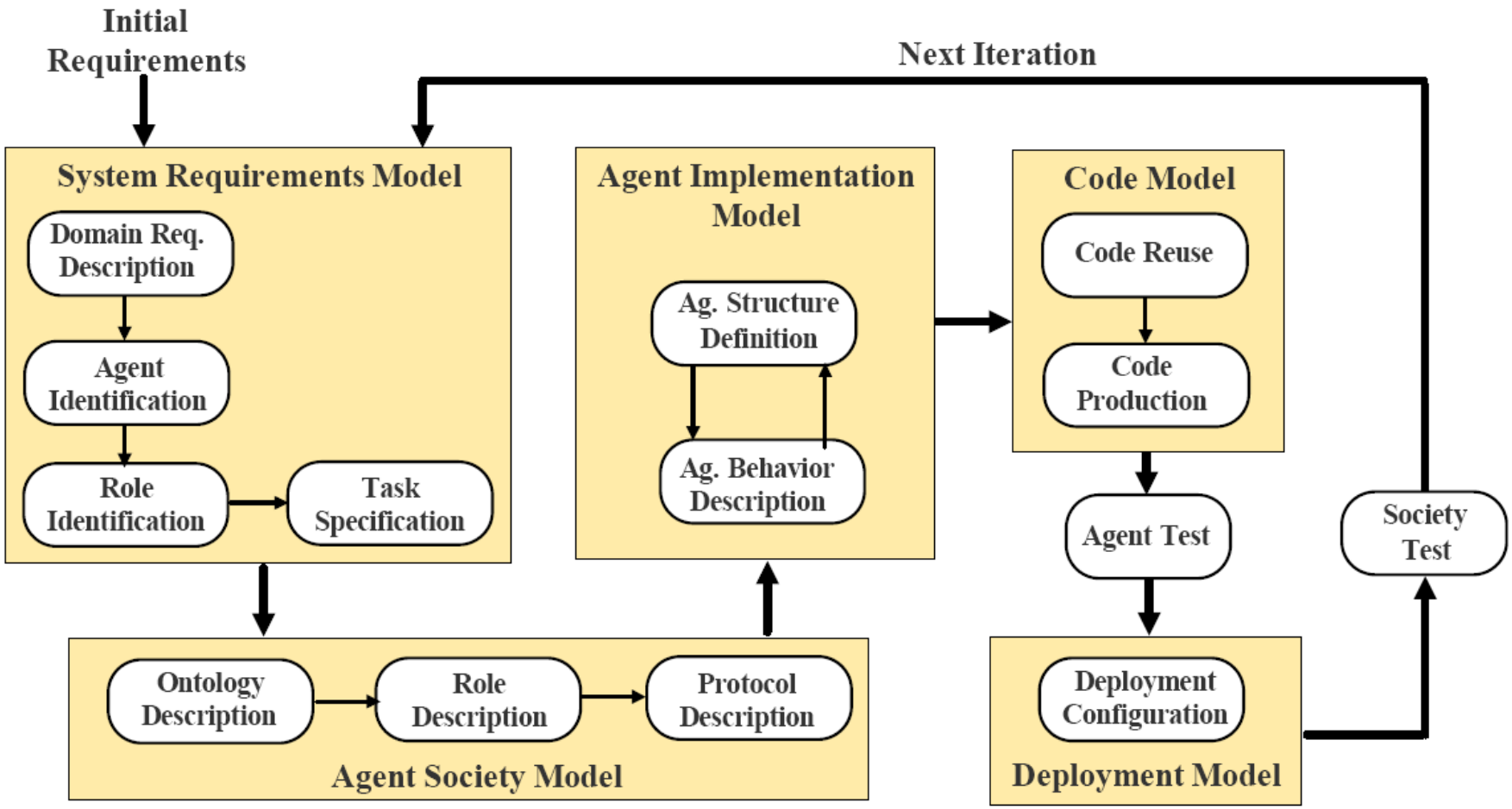
# Vowels Methodology



- The main idea of the vowels methodology is that a MAS is consisted of four major component types
  - the Agent
  - the Environment
  - Interactions
  - Organization
- It is a methodology that introduced these four different aspects in MAS development for the first time in a modular architecture.

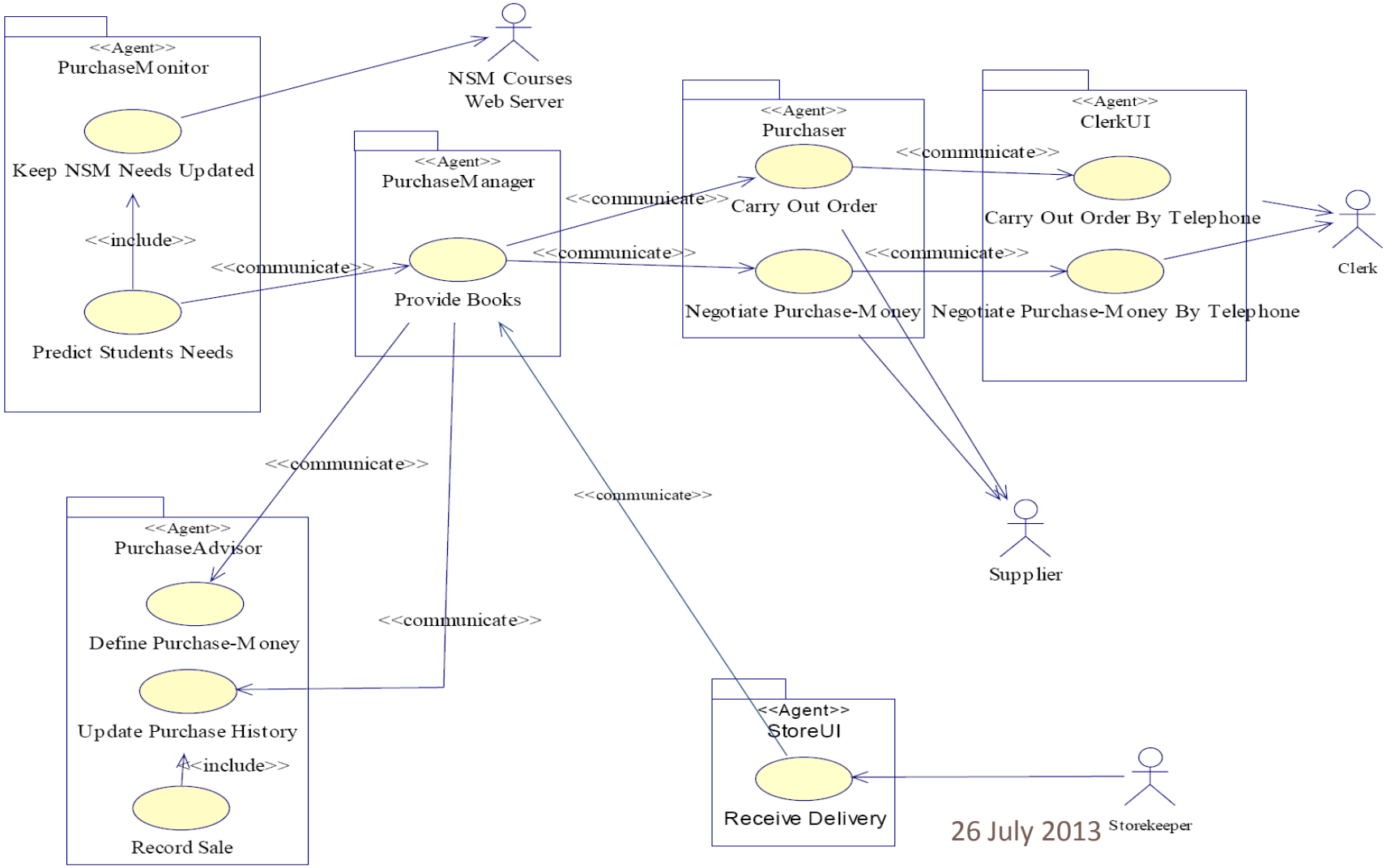
# PASSI models and phases





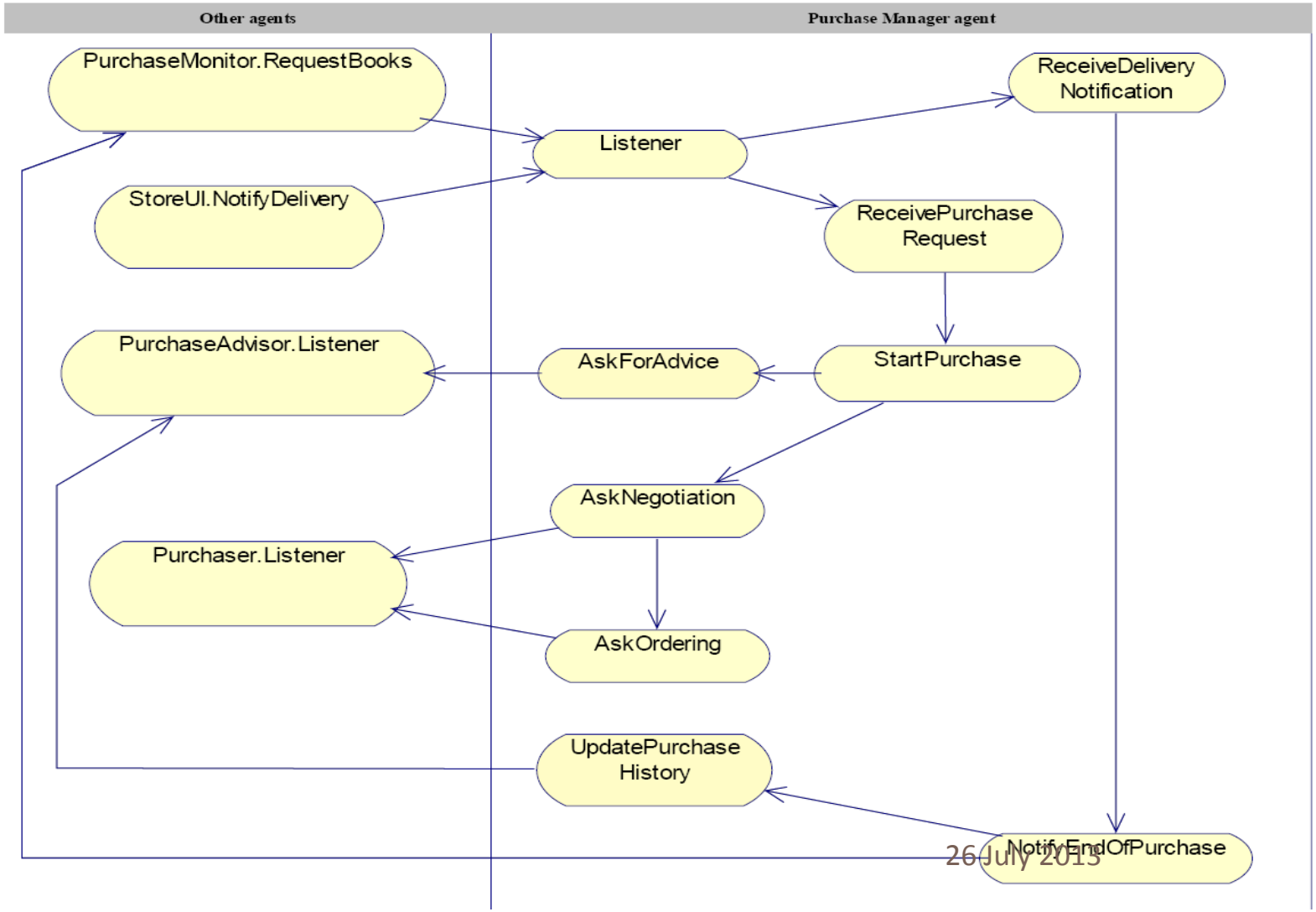
# Agents Identification Diagram





# Activity Diagram





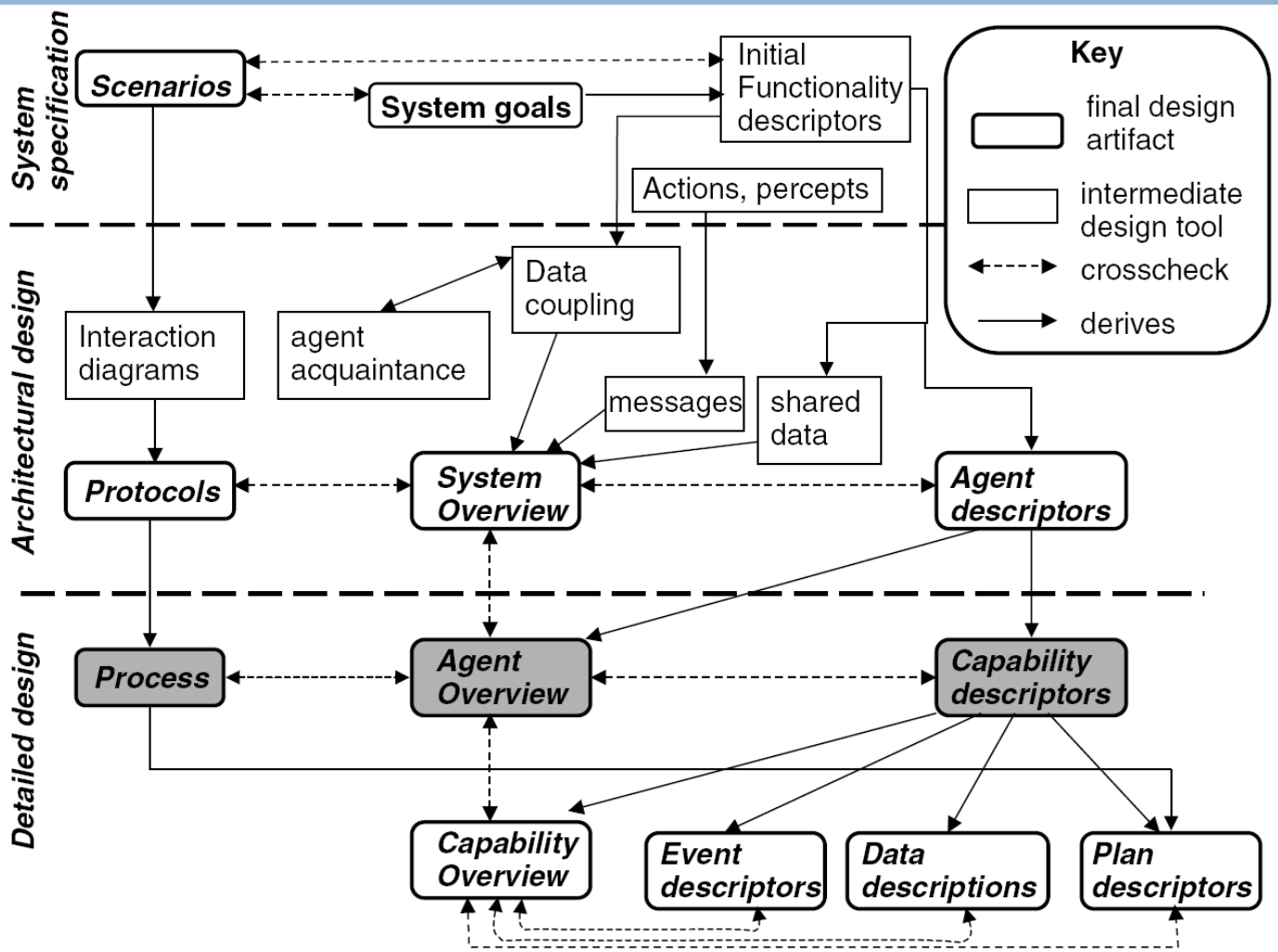
# Prometheus



- Prometheus supports the development of *intelligent* agents, entities with
  - Goals
  - Beliefs
  - Plans
  - Events
- Uses the popular JACK Intelligent Agents Platform (Winikoff, 2005) for implementation
- Uses AUML AIP diagrams for protocols definition

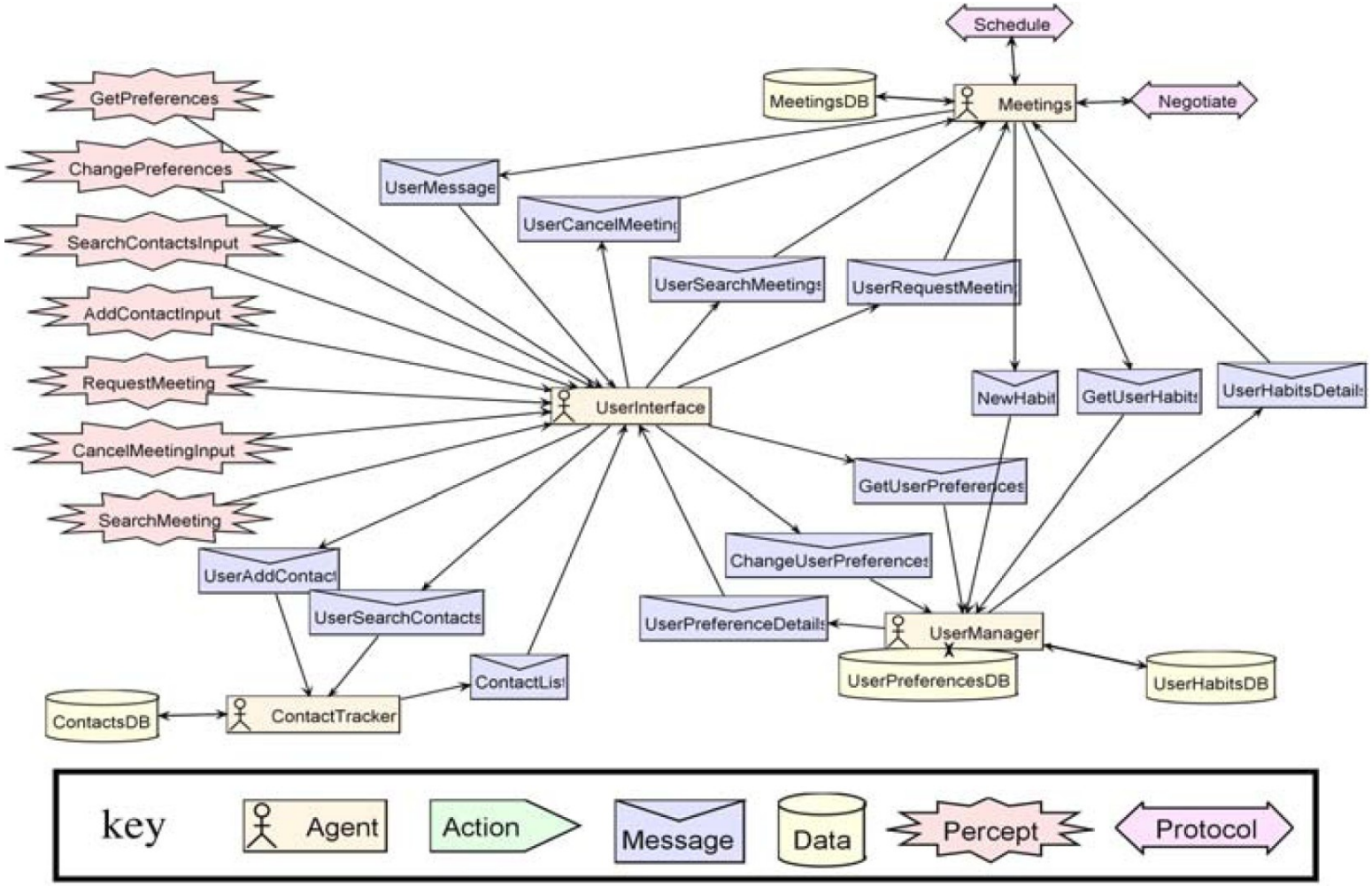
# Prometheus Phases





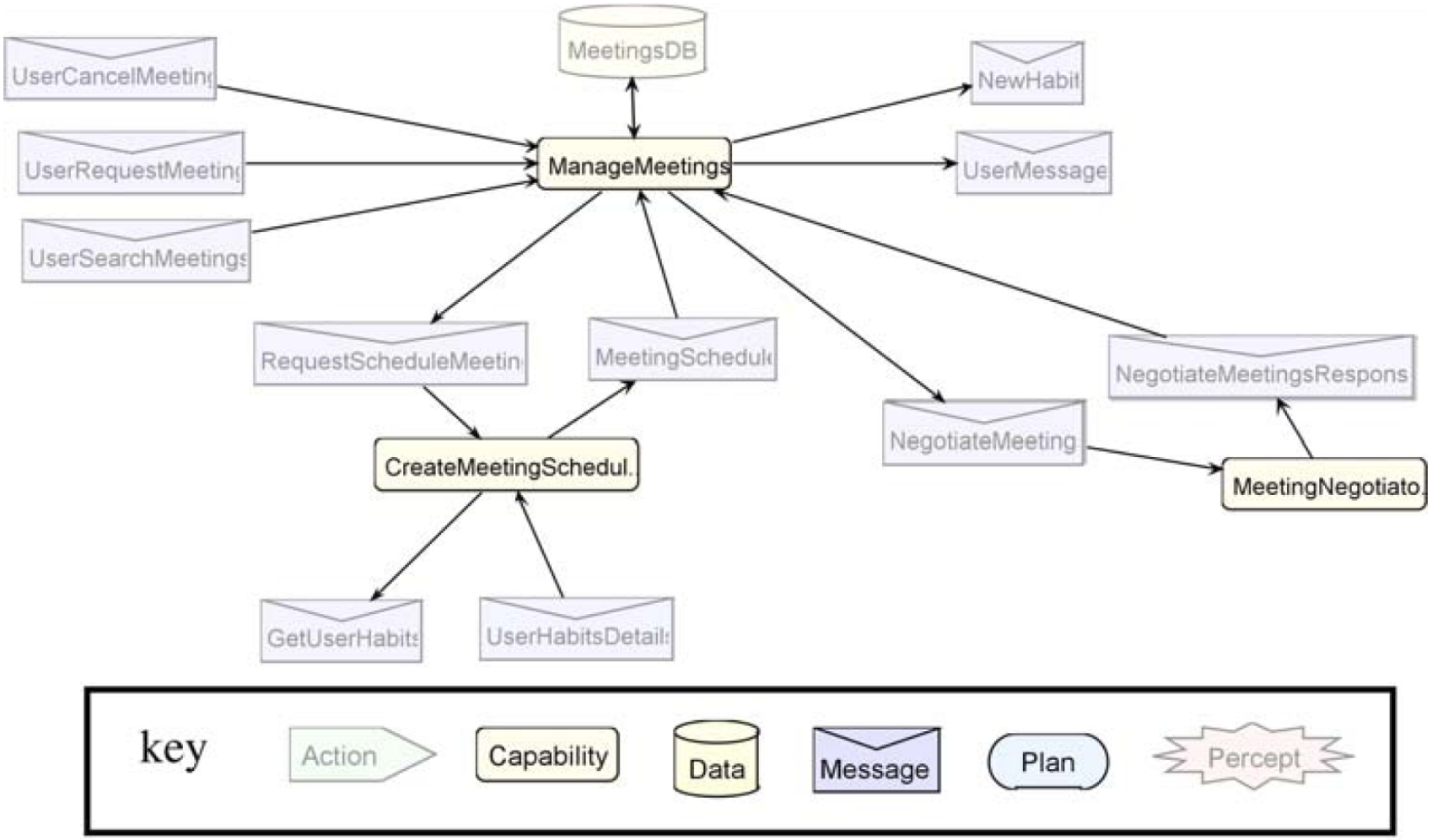
# System Overview Diagram





# Agent Overview Diagram

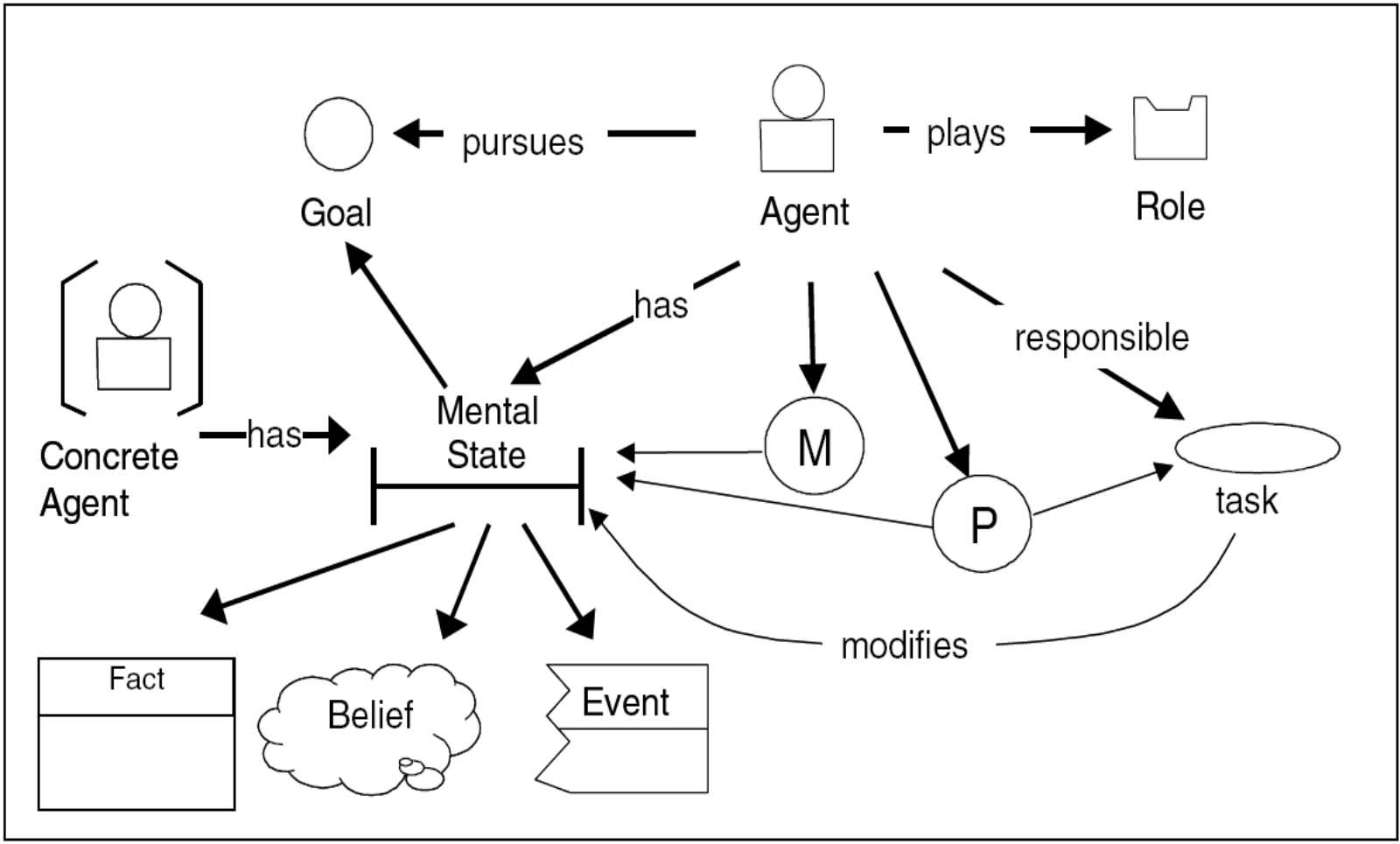




# Ingenias



- UML and BDI – based agent modeling
- AUML can be used for defining protocols
- Interesting tools are available for a quite complex but also detailed methodology (more than 100 activities)
- A tool allows for model transformation by example for aiding the developers to automate transition between different phases models






# Elements of the agent viewpoint

(Pavón, Gómez-Sanz & Fuentes, 2005)

# Tropos

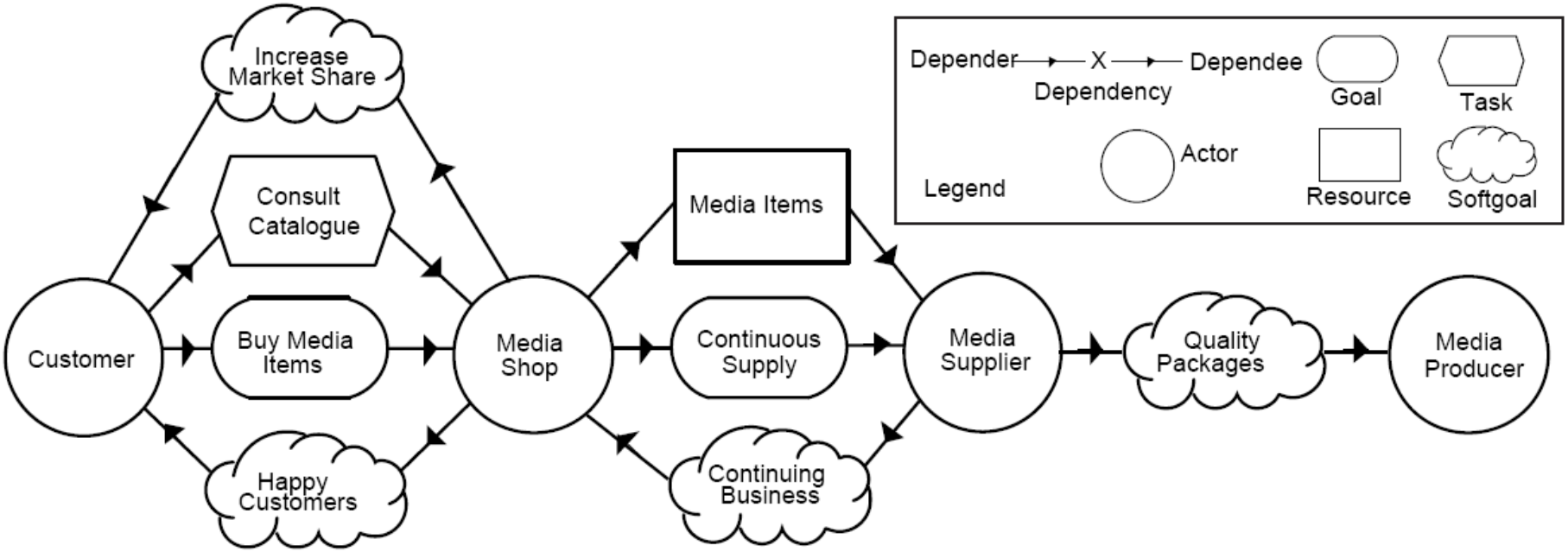


- Based on ideas brought by requirements-driven software engineering methods (i*)
- Allows for capturing non-functional requirements
- Allows for mapping Tropos concepts to BDI concepts
- And development using JACK
- Temporal logic-based formal Tropos
- AUML AIP for protocols design

# Actor diagram



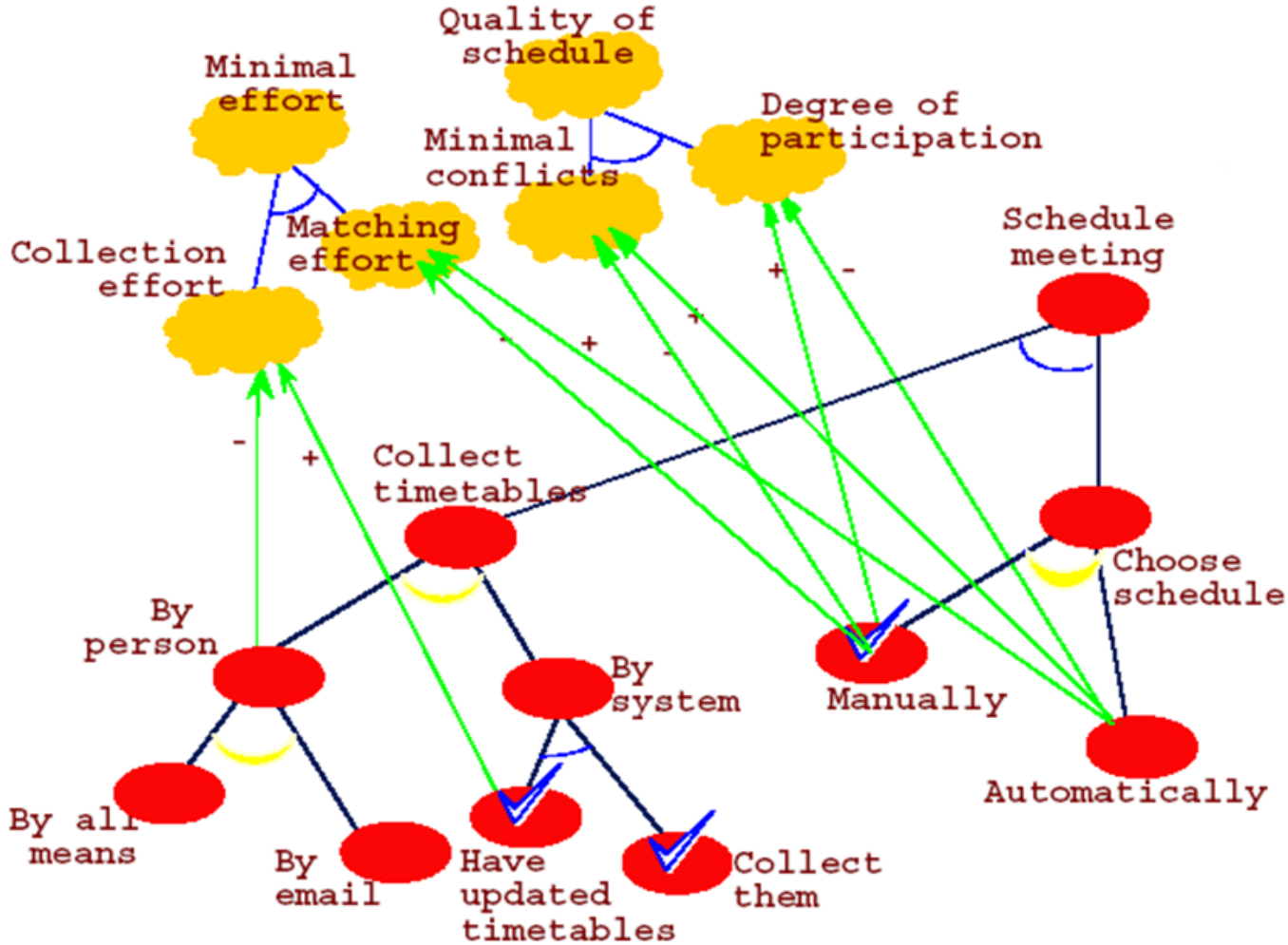
- Early requirements capture



# Goal Analysis



- Decompose
- Allow for adaptive modeling
- Aid or hinder a soft goal




(Bresciani et al., 2004)

# Protocols Specification

A key issue in AOSE is protocols specification

How is a protocol defined?

What are its properties?

# Agent Communication Language



- the message types of ACLs (or *performatives*) are understood as *speech acts* (Austin 1975)
- Performatives express the intent of an agent when it sends a message to another agent
- FIPA defined an ACL message by the atom:

*performative(sender, receiver, content)*

# Moore's Conversation Policies



- Moore (2000) makes the assumption that developers that adopt models can understand a formal specification and implement it as they see fit
- For example, the message
  *request(sender, receiver, action)*
  expresses that:
  - The *receiver* believes that the *sender* wants him to do the *action*
  - The *receiver* believes that the *sender* wants the *receiver* to want to do the *action* (Moore, 1999)

# Definition of a CP

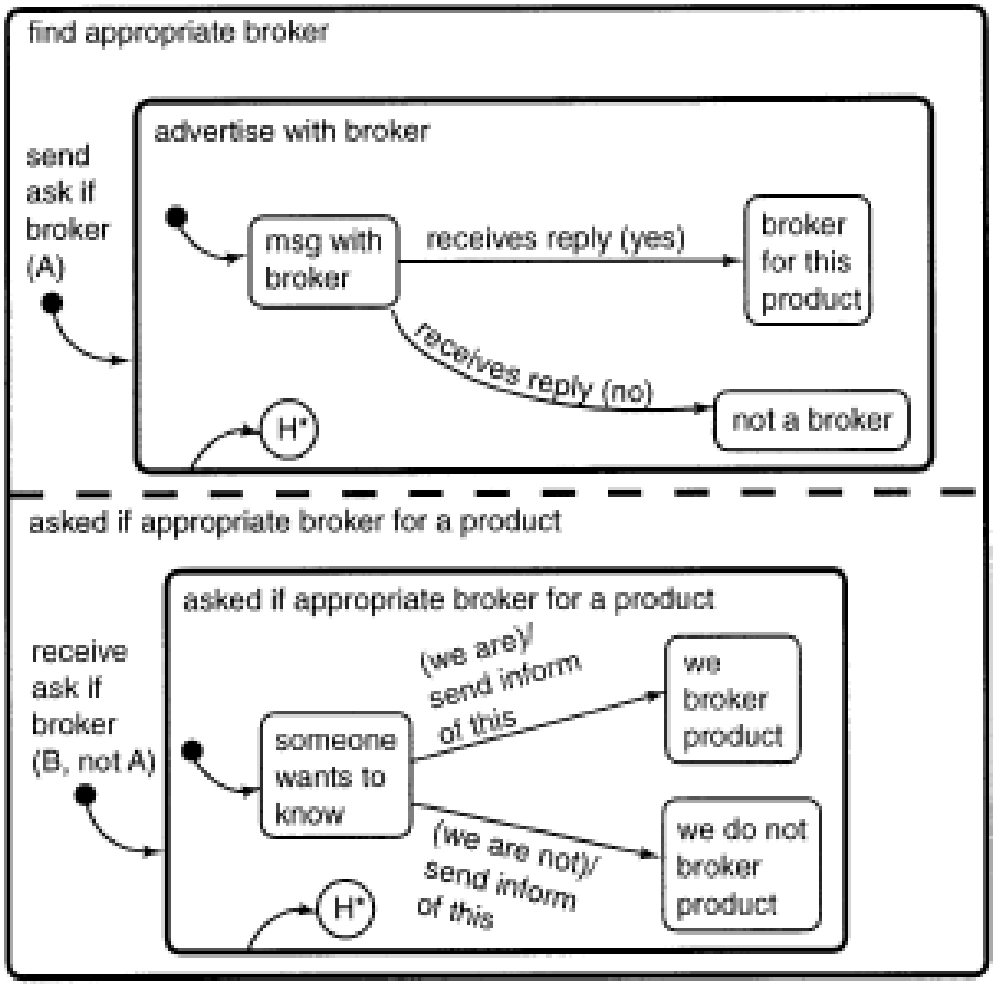


- According to the work of Moore, the conversation policies are implementation independent.
- A conversation policy (CP) defines:
  - how one or more conversation partners respond to messages they receive,
  - what messages a partner expects in response to a message it sends, and,
  - the rules for choosing among competing courses of action.

# A conversation



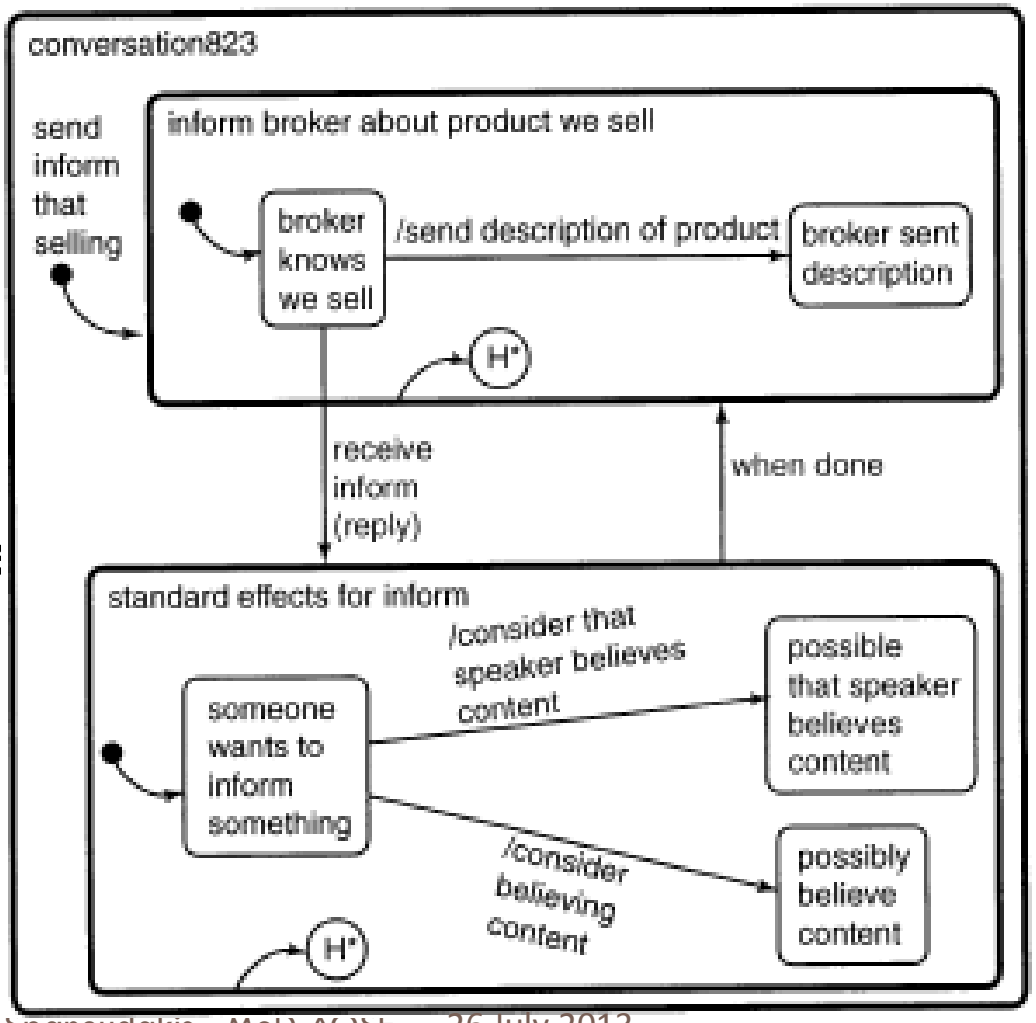
- Modeled as a statechart
- Participants are orthogonal components



# CP with a subdialog



- CPs are available in the agent's repository
- While in a CP an unexpected message in the same conversation can cause a search for another CP



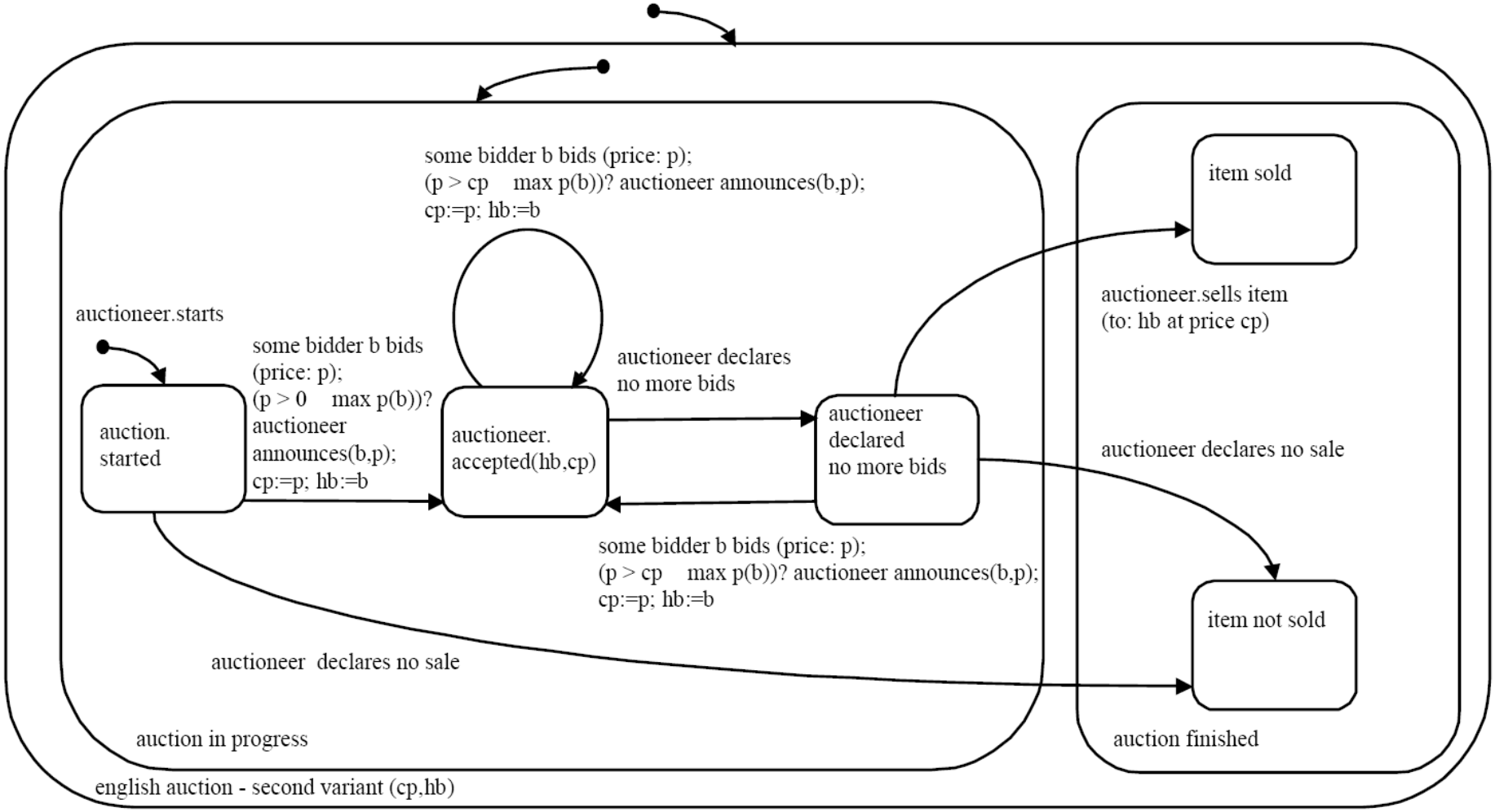
# Propositional Statecharts



- Paurobally et al. (2004) propose that an inter-agent protocol should be:
  - correct (having no contradictory states),
  - unambiguous (what each agent should do),
  - complete (defining all possible outcomes)
  - verifiable (its properties can be verified).
- they combine the
  - Statecharts
  - Agent Negotiation Meta-Language (ANML)
    - based on Propositional Dynamic Logic (PDL)

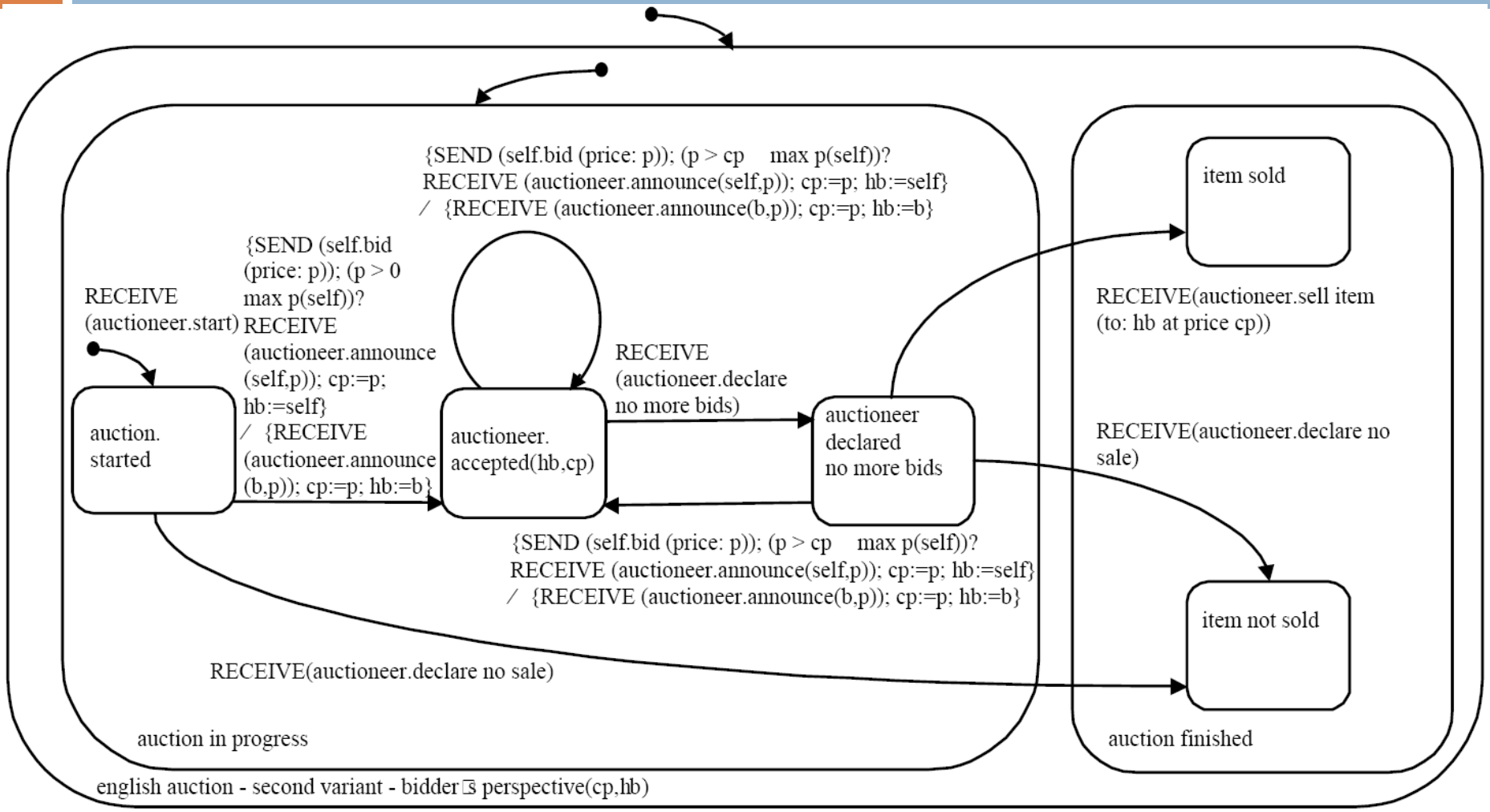
# View Both Participants





# Individual Point of View





# Commitment-based ACL



- A commitment has (Formara and Colombetti, 2003):
  - debtor
  - creditor
  - content (a proposition)
  - a list of propositions that have to be satisfied in order for the commitment to become *active*;
  - state {*unset, cancelled, pending, active, fulfilled, violated*}
  - a timeout valid only for *unset* commitments
- Protocols are presented as interaction diagrams
  - States: System states
  - Transitions: speech-acts / environment events

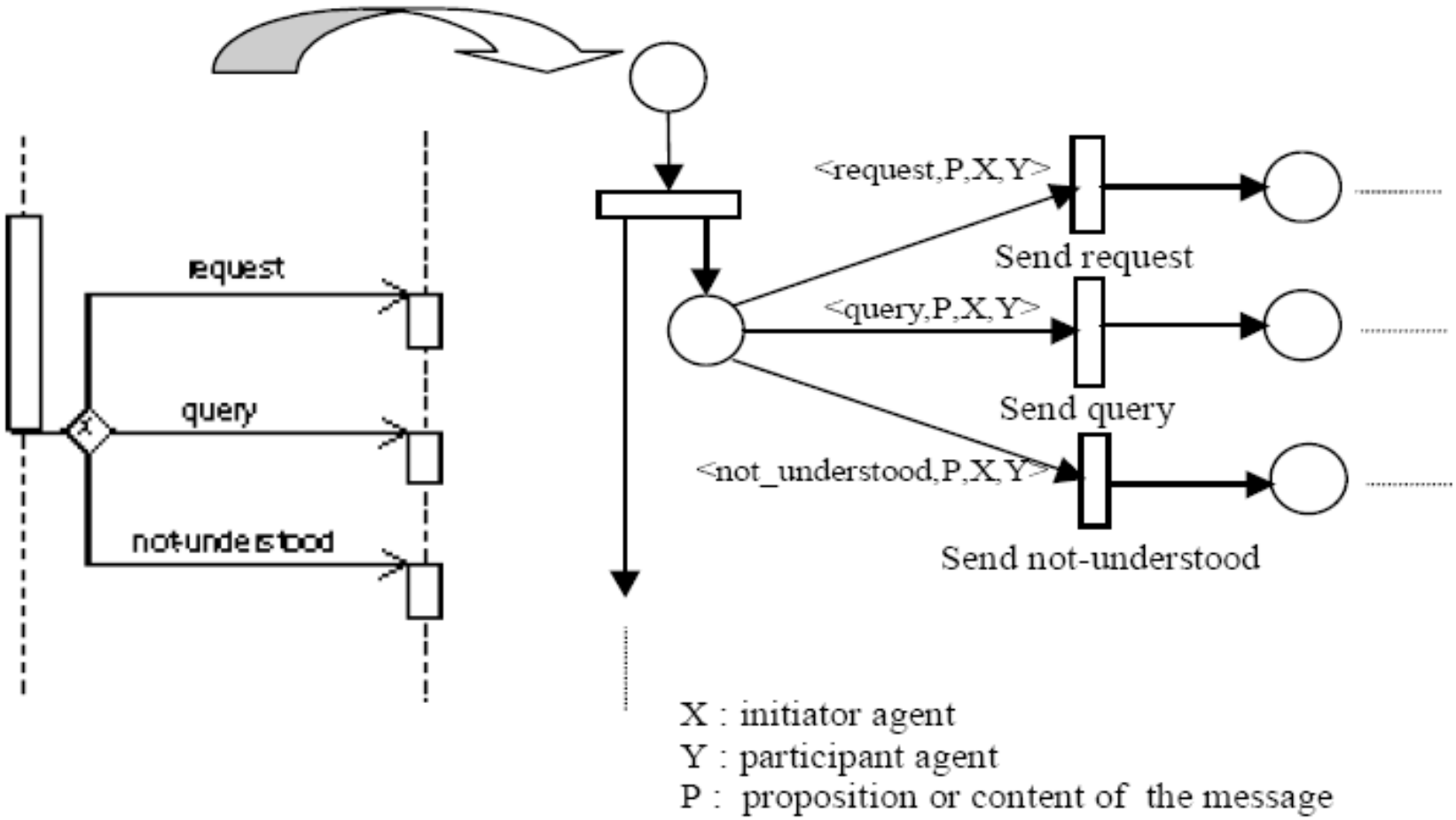
# Colored Petri Nets (CPN)



- Petri nets are well suited for modeling the concurrent behavior of distributed systems
- Formal model allowing for verification
- Mazouzi et al. (2002) showed how to transform an AUML interaction diagram to a CPN
  - Complexity remains tractable
  - Can become very complex
  - Limited reusability and abstraction

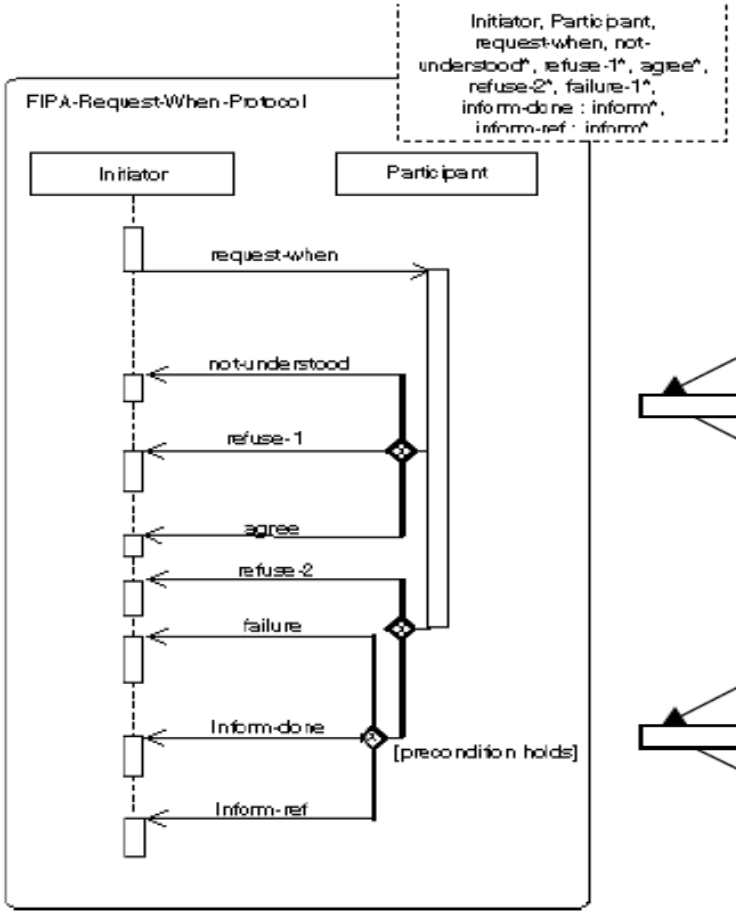
# Exclusive OR part of an AIP to CPN

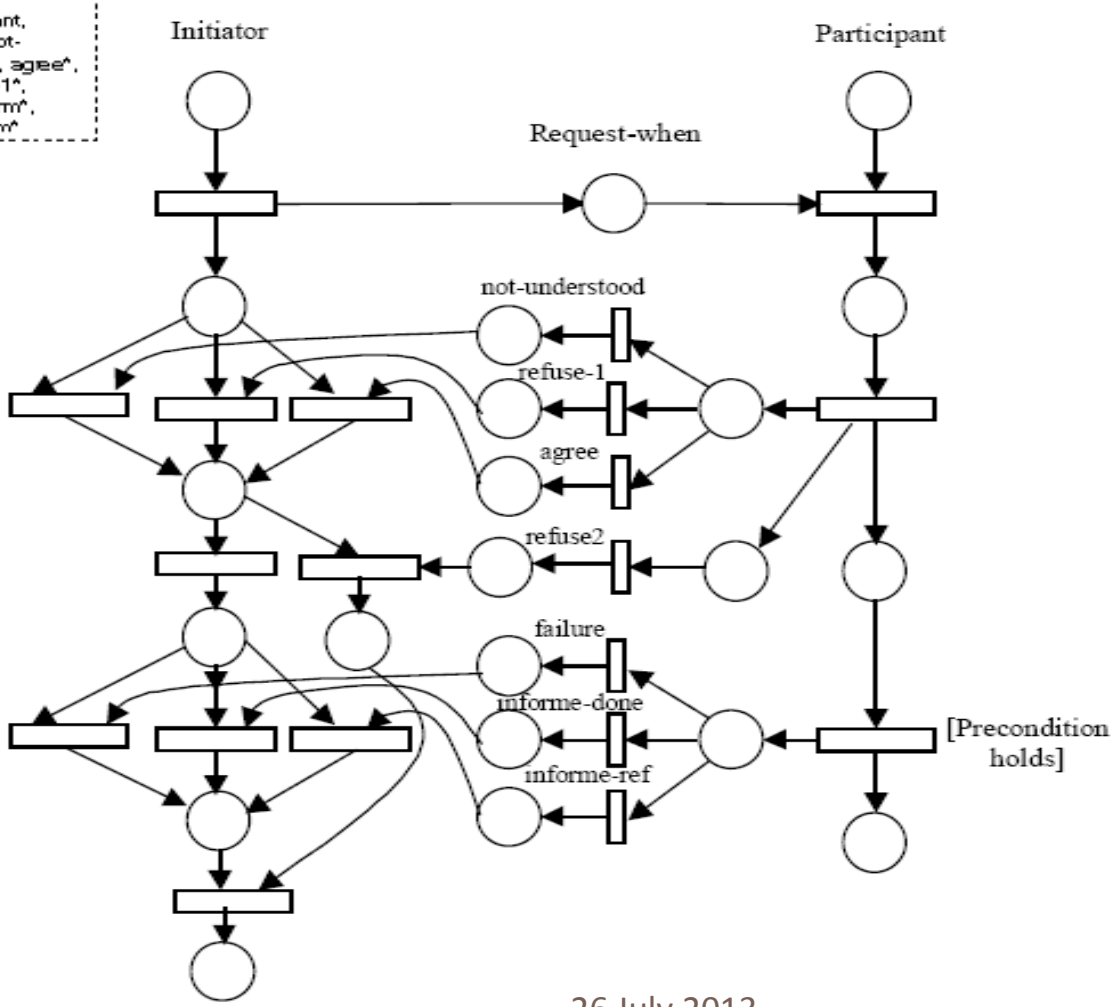




# AIP to CPN

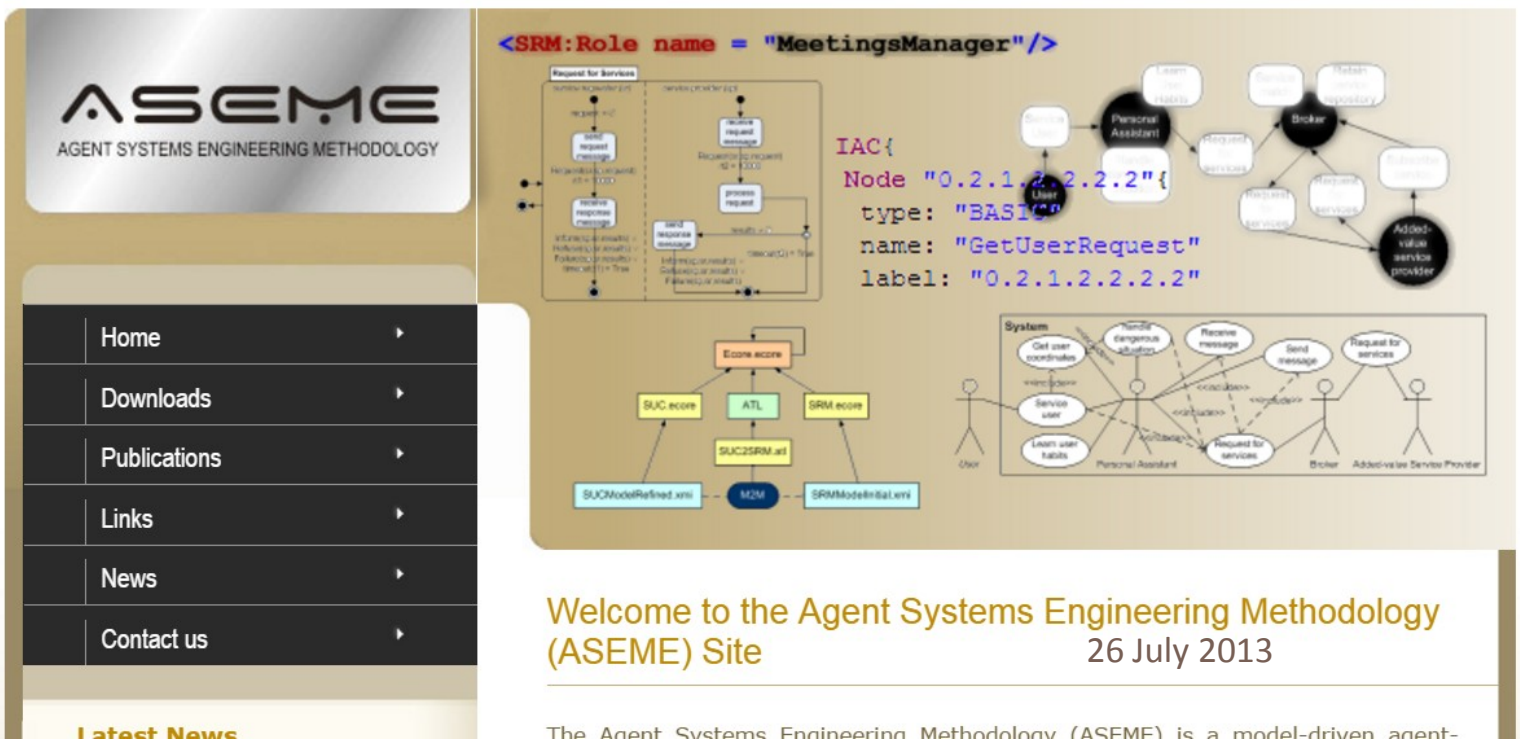



Initiator, Participant,
request-when, not-
understood*, refuse-1*, agree*,
refuse-2*, failure-1*,
inform-done : inform*,
inform-ref : inform*
FIPA-Request-When-Protocol
Initiator
Participant
request-when
not-understood
refuse-1
agree
refuse-2
failure
Inform-done
[precondition holds]
Inform-ref


Initiator
Participant
Request-when
not-understood
refuse-1
agree
refuse2
failure
informe-done
informe-ref
[Precondition
holds]

# Model-Driven AOSE with ASEME

http://aseme.tuc.gr



# ASEME

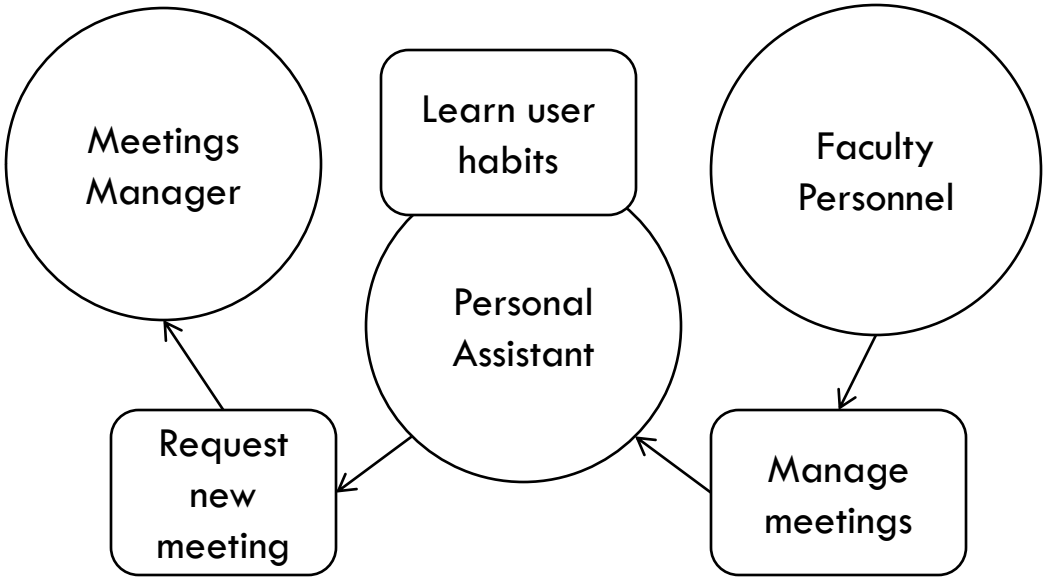


- ASEME can be considered as a Hybrid methodology using ideas from other methodologies such as Tropos, AUML and Gaia but also from the work of Moore
- We use it to demonstrate how an AOSE Methodology uses
  - The modeling language
  - The process
  - The tools

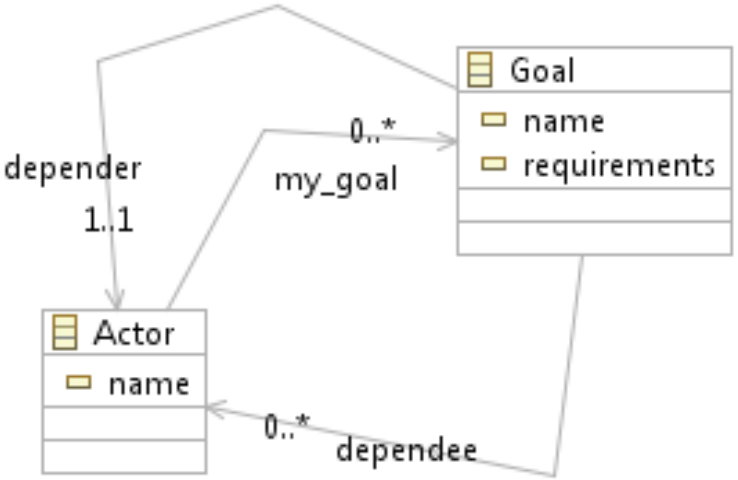
# AMOLA Model: Requirements Analysis



- The **System Actors-Goals model (SAG)** based on the Tropos actor diagram



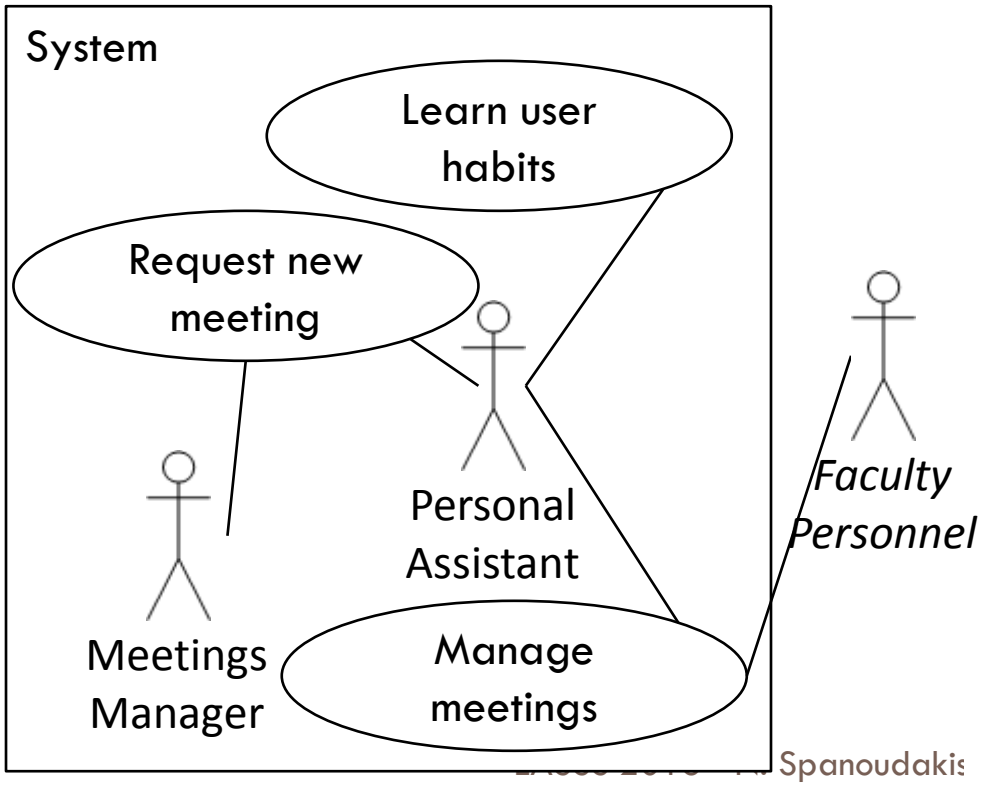

- And its metamodel definition using Ecore, the Eclipse Modeling Framework 's (EMF) model of a model



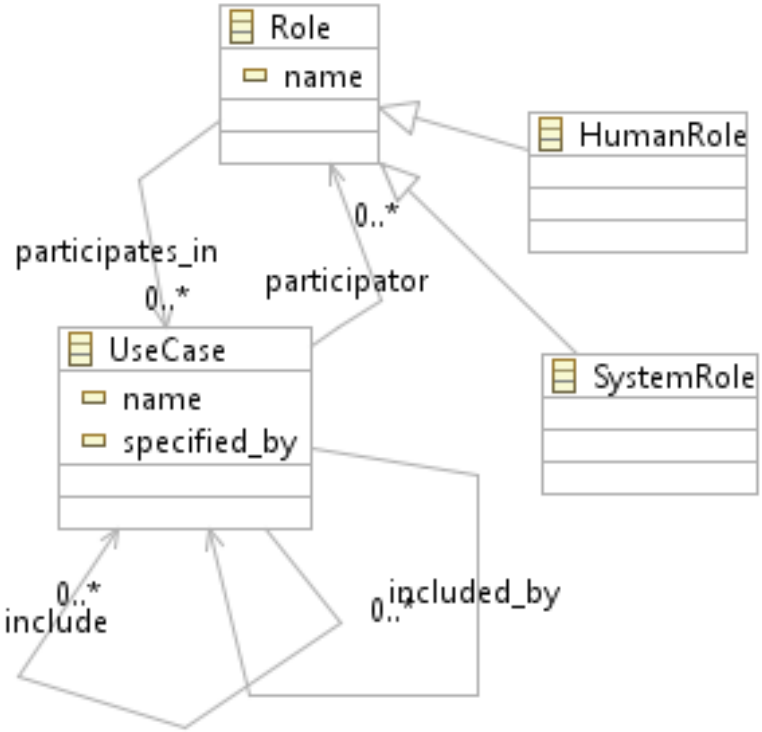
# AMOLA Model: Analysis



- The **System Use Cases model (SAG)** based on the UML use cases




- And its metamodel definition



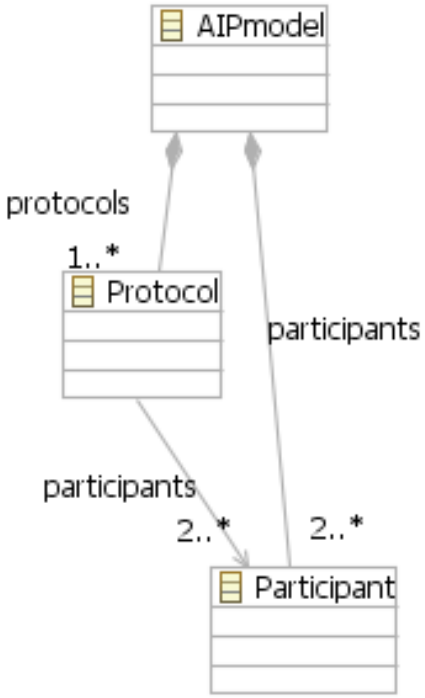
# AMOLA Model: Analysis (cont)



- The **Agent Interaction Protocol model (AIP)**

| | **Request new meeting** | |
|---|---|---|
| ***Participants*** | *Personal Assistant* | *Meetings Manager* |
| ***Rules for engaging*** | He needs to create a meeting | - |
| ***Outcomes*** | He has received a confirmation of the meeting creation | - |
| ***Process*** | request new meeting = send new request. receive new results | request new meeting = receive new request. send new results |

- And its metamodel definition



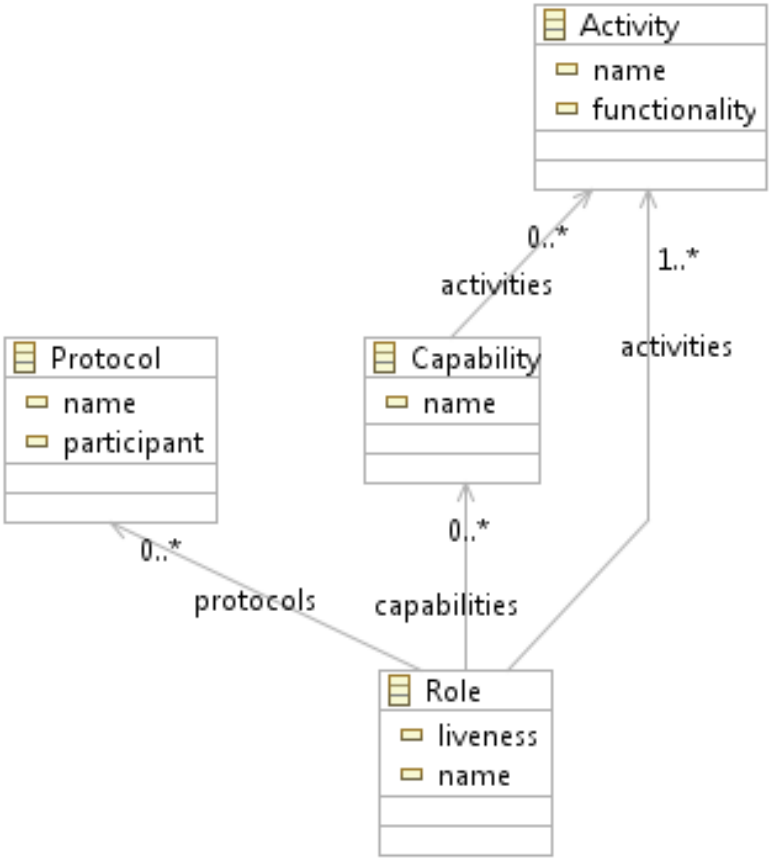
# AMOLA Model: Analysis (cont)



- The **System Roles model (SRM)** based on the Gaia roles model

**Role:** Personal Assistant
**Capabilities and Protocols:**
learn user habits, request new meeting: personal assistant, …
**Activities:**
learn user preference, update user preferences, …
**Liveness:**
personal assistant = request new meeting || learn user habits
learn user habits = learn user preference. update user preferences
…

- And its metamodel



# Grammar metametamodel

- A grammar defines the syntax of a language, such is EBNF (Wirth, 1996)

BnfGrammar → { Rule }

Rule → NonTerminal → Expression

Expression → Alternative | Composition | PExpression

PExpression → Optional | AtLeastOne | Arbitrary | Symbol | ( Expression )

Alternative → PExpression | (PExpression | Alternative)

Composition → PExpression +

Optional → [ Expression ]

AtLeastOne → PExpression +

Arbitrary → { Pexpression }

Symbol → Terminal | NonTerminal

Defines itself

# AMOLA: Liveness



## EBNF syntax

```
liveness                → { formula }
formula                 → leftHandSide = expression
leftHandSide            → string
expression              → term
                         | parallelExpression
                         | orExpression
                         | sequentialExpression
parallelExpression      → term || term || … || term
orExpression            → term | term | … | term
sequentialExpression    → term . term . … . term
term                    → basicTerm
                         | (expression)
                         | [expression]
                         | term*
                         | term+
                         | term^ω
                         | |term^ω|^number
basicTerm               → string
number                  → digit | digit number
digit                   → 1 | 2 | 3 | …
string                  → letter | letter string
letter                  → a | b | c | …
```

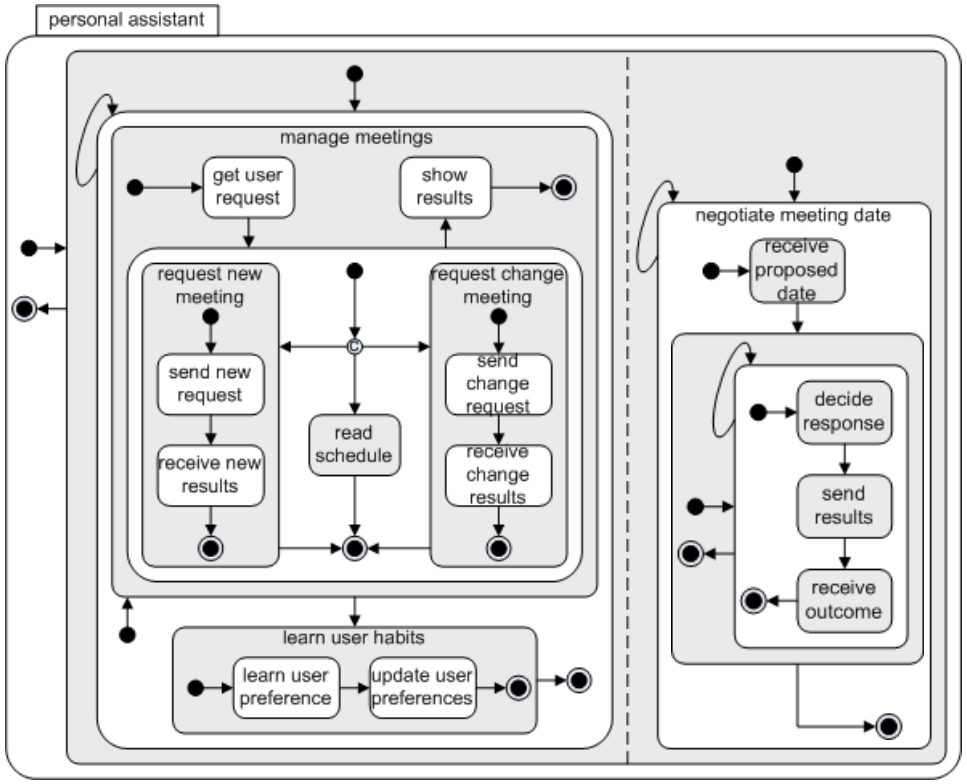
## Semantics

| Operator | Interpretation |
|---|---|
| **x . y** | x followed by y |
| **x \| y** | x or y occurs |
| **x*** | x occurs 0 or more times |
| **x+** | x occurs 1 or more times |
| **$x^{\omega}$** | x occurs infinitely often |
| **[x]** | x is optional |
| **x \|\| y** | x and y interleaved |
| **$\|x^{\omega}\|^{n}$** | x occurs infinitely often n times in parallel |

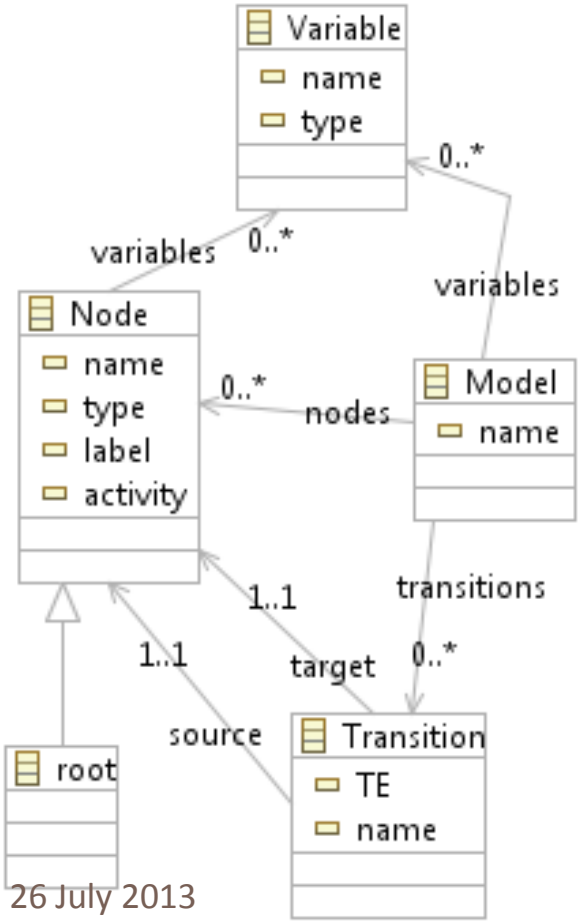
# AMOLA Model: Design



- The **Inter (EAC) and Intra-Agent Control (IAC)** based on statecharts




- And its metamodel d

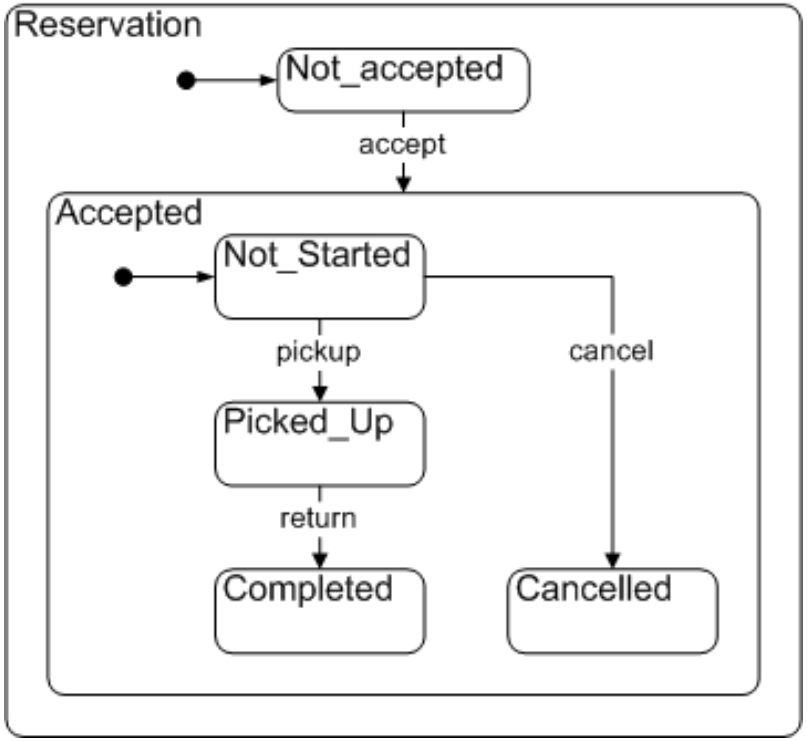


# AMOLA Design: Statecharts

- Syntax (Harel and Naamad, 1996):
  - Start state
  - End state
  - Basic state
  - And state
  - Or state
  - Condition state
  - History state
  - Transition
  - Transition Expression: `e[c]/a`

# Example Statechart for a reservation

- Assume that once a reservation becomes accepted, then it is in a waiting state until the client picks up the vehicle. The client then returns the vehicle and completes the lifecycle of the reservation. Before the client picks up the vehicle he can cancel the reservation.



# ASEME Process

The transformations are fully automated

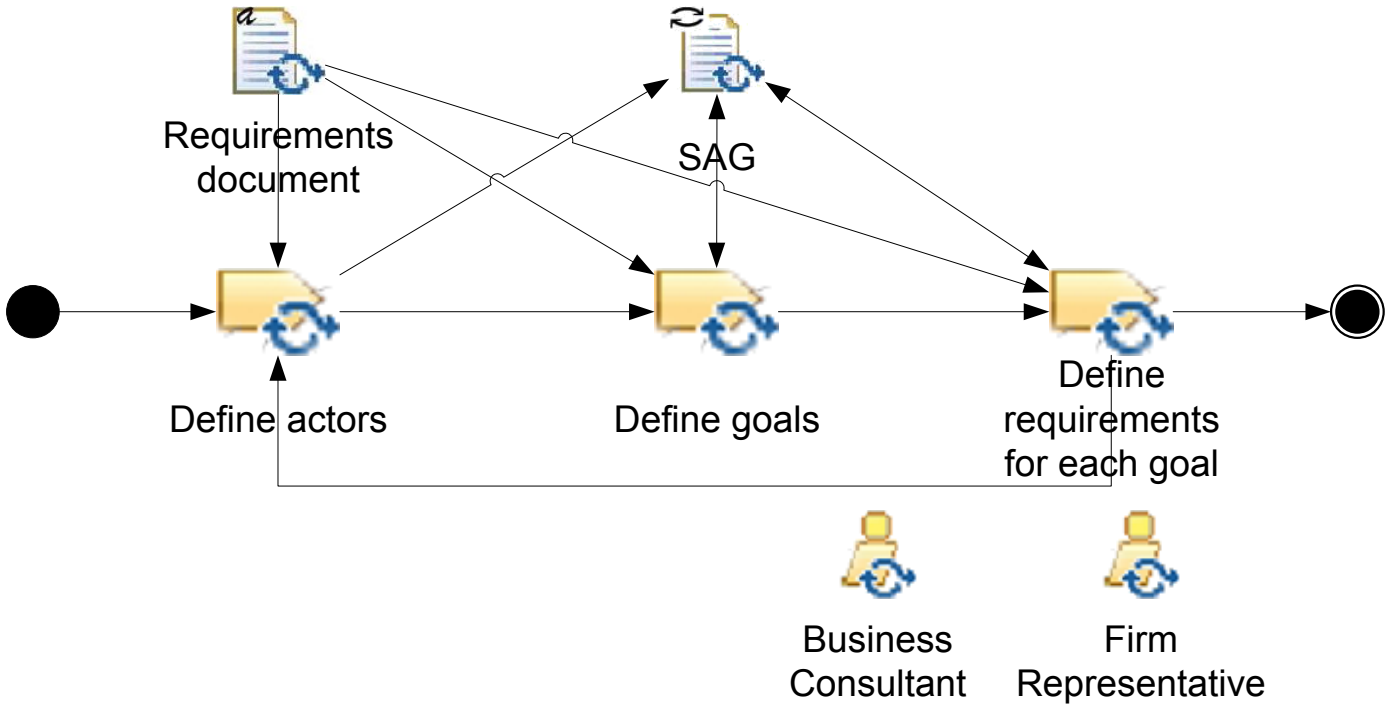
A model of a previous phase is transformed to an instance of the model of the next phase (*initial model*)

Editing and refinement of models is done by the engineer and produces the *refined model*

# ASEME Requirements Analysis Phase

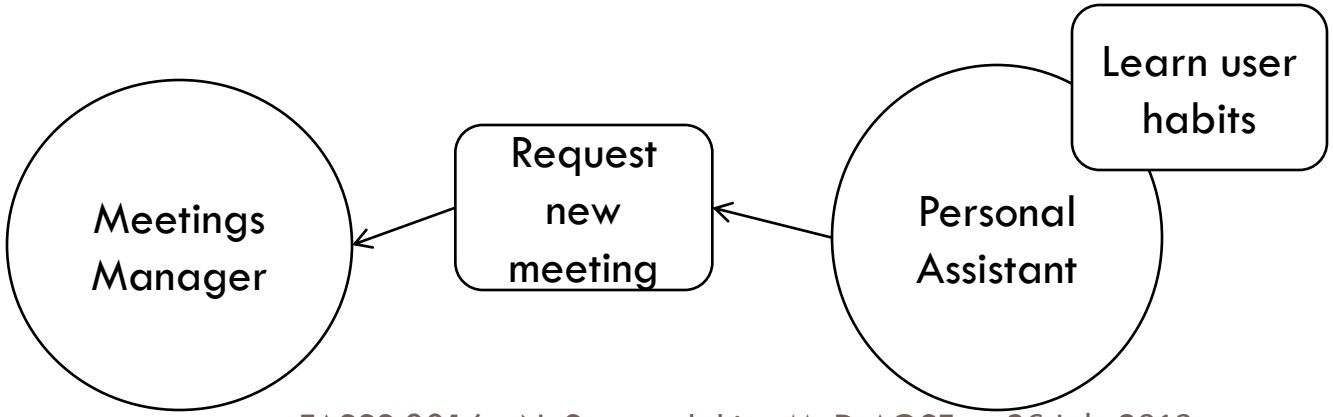


- Using the OMG Software Process Engineering Metamodel (SPEM v2.0)



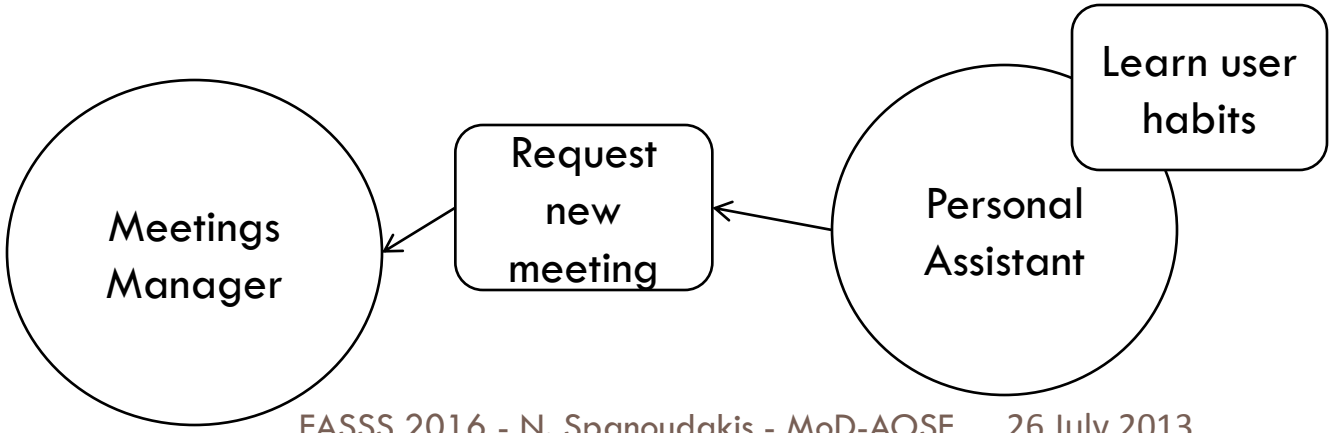
# Create and Edit the SAG Model





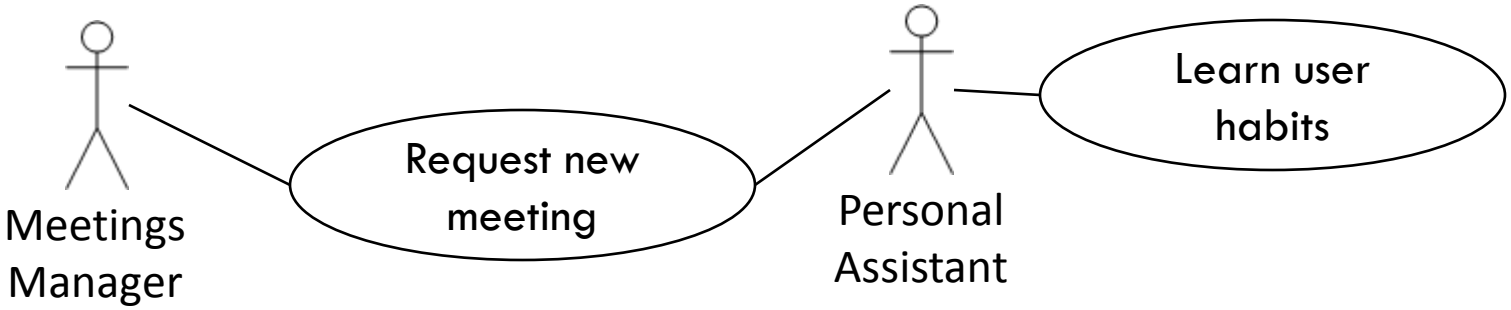
# The SAG 2 SUC Transformation





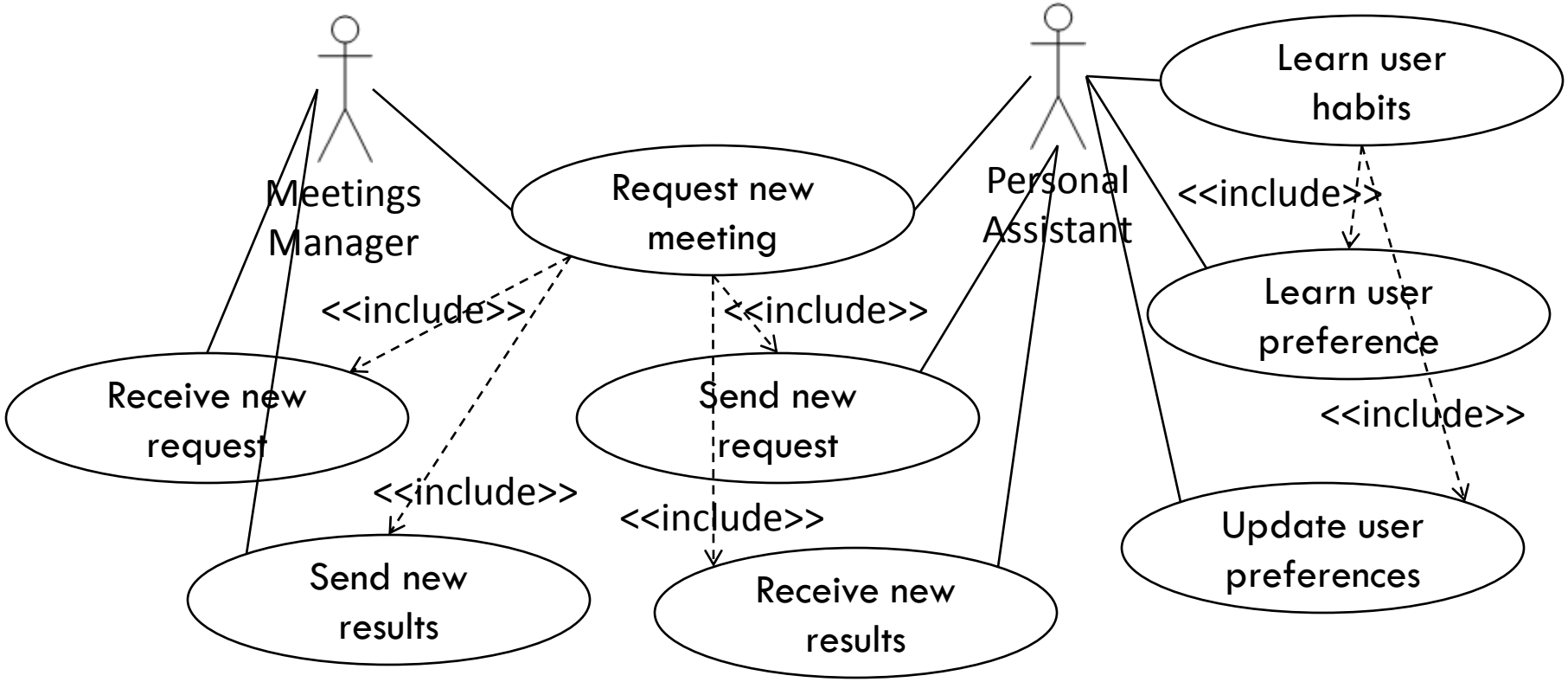
# The SAG 2 SUC Transformation





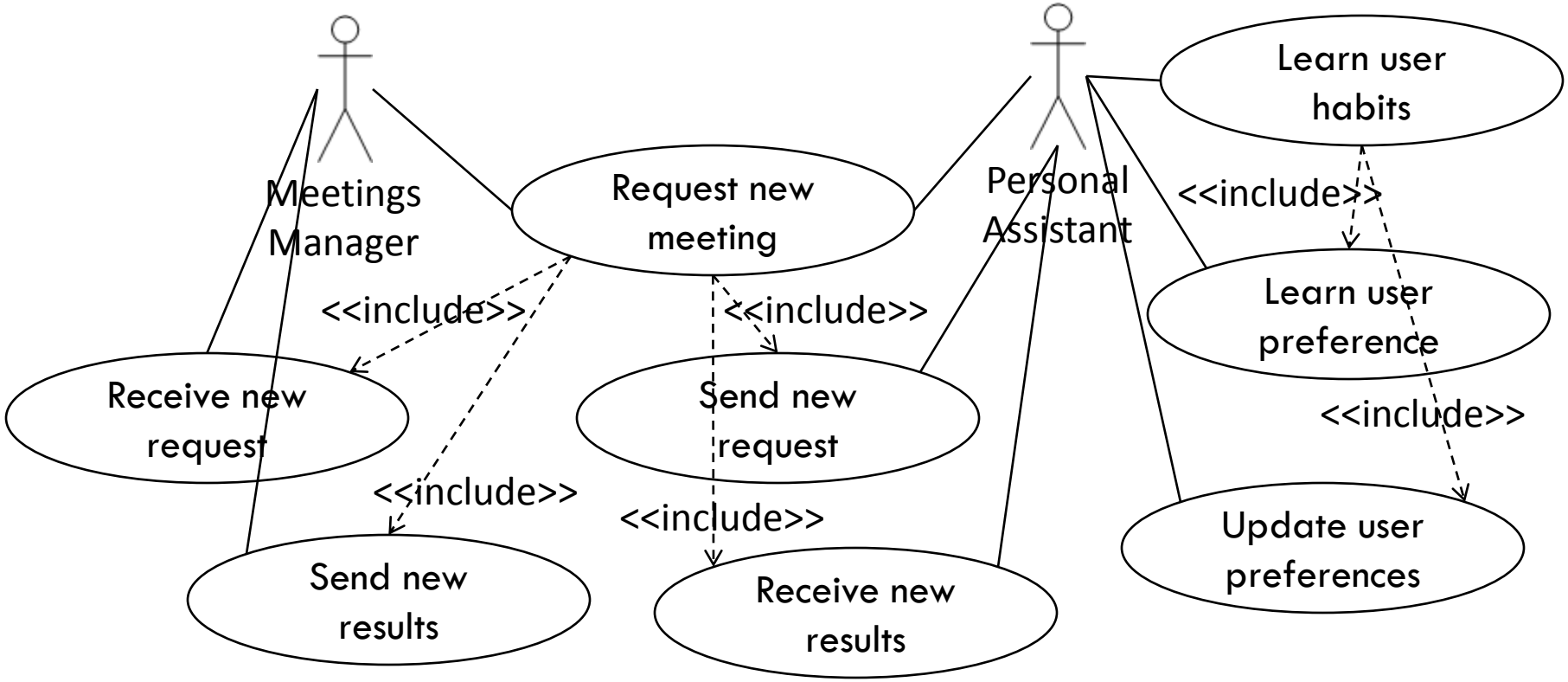
# Refine SUC Model





# SUC 2 SRM Transformation





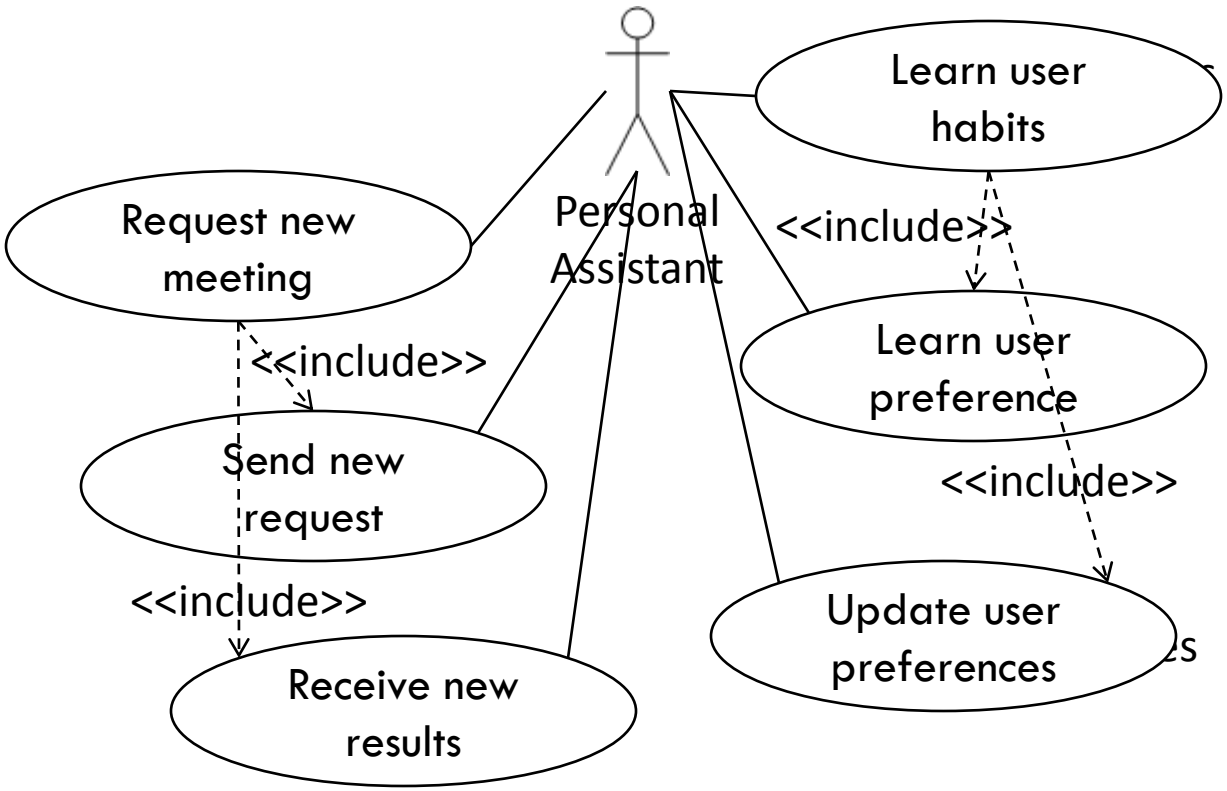
# SUC 2 SRM Transformation



**Role:**

**Capabilities and Protocols:**

**Activities:**



# Refine SRM



**Role:** Personal Assistant
**Capabilities and Protocols:**
manage meetings, learn user habits,
negotiate meeting date, request change meeting, request new meeting
**Activities:**
get user request, read schedule, show results,
learn user preference, update user preferences,
send change request, receive change results, send new request, receive new results,
*receive proposed date, decide response, send results, receive outcome, update schedule*
**Liveness:**

# Refine SRM



**Role:** Personal Assistant
**Capabilities and Protocols:**
manage meetings, learn user habits,
negotiate meeting date, request change meeting, request new meeting
**Activities:**
get user request, read schedule, show results,
learn user preference, update user preferences,
send change request, receive change results, send new request, receive new results,
*receive proposed date, decide response, send results, receive outcome, update schedule*
**Liveness:**
personal assistant = ?
manage meetings = get user request ? read schedule ? request change meeting ? request new meeting? show results
learn user habits = learn user preference ? update user preferences

# Refine SRM

Get information from existing AIP model



**Role:** Personal Assistant

| | Request new meeting | | Negotiate meeting date | |
|---|---|---|---|---|
| ***Participants*** | *Personal Assistant* | *Meetings Manager* | *Personal Assistant* | *Meetings Manager* |
| ***Rules for engaging*** | He needs to create a meeting | - | He needs to create or participate to a meeting | He has a meeting with more than one participants that has no date assigned to it |
| ***Outcomes*** | He has received a confirmation of the meeting creation | - | He has scheduled participation to a meeting | He has arranged a meeting that met all the participants needs |
| ***Process*** | request new meeting = send new request. receive new results | request new meeting = receive new request. send new results | negotiate meeting date = receive proposed date. (decide response. send results. receive outcome)+ | negotiate meeting date = (decide on date. send proposed date. receive results)+. send fixed date. |

learn user habits = learn user preference ? update user preferences

request new meeting = ***send new request. receive new results***

request change meeting = ***send change request. receive change results***

negotiate meeting date = ***receive proposed date. (decide response. send results. receive outcome)+***

? update schedule

# Refine SRM



**Role:** Personal Assistant
**Capabilities and Protocols:**
manage meetings, learn user habits,
negotiate meeting date, request change meeting, request new meeting
**Activities:**
get user request, read schedule, show results,
learn user preference, update user preferences,
send change request, receive change results, send new request, receive new results,
*receive proposed date, decide response, send results, receive outcome, update schedule*
**Liveness:**
personal assistant = ?
manage meetings = get user request ? read schedule ? request change meeting ? request new meeting? show results
learn user habits = learn user preference ? update user preferences
request new meeting = ***send new request. receive new results***
request change meeting = ***send change request. receive change results***
negotiate meeting date = ***receive proposed date. (decide response. send results. receive outcome)+*** ? update schedule

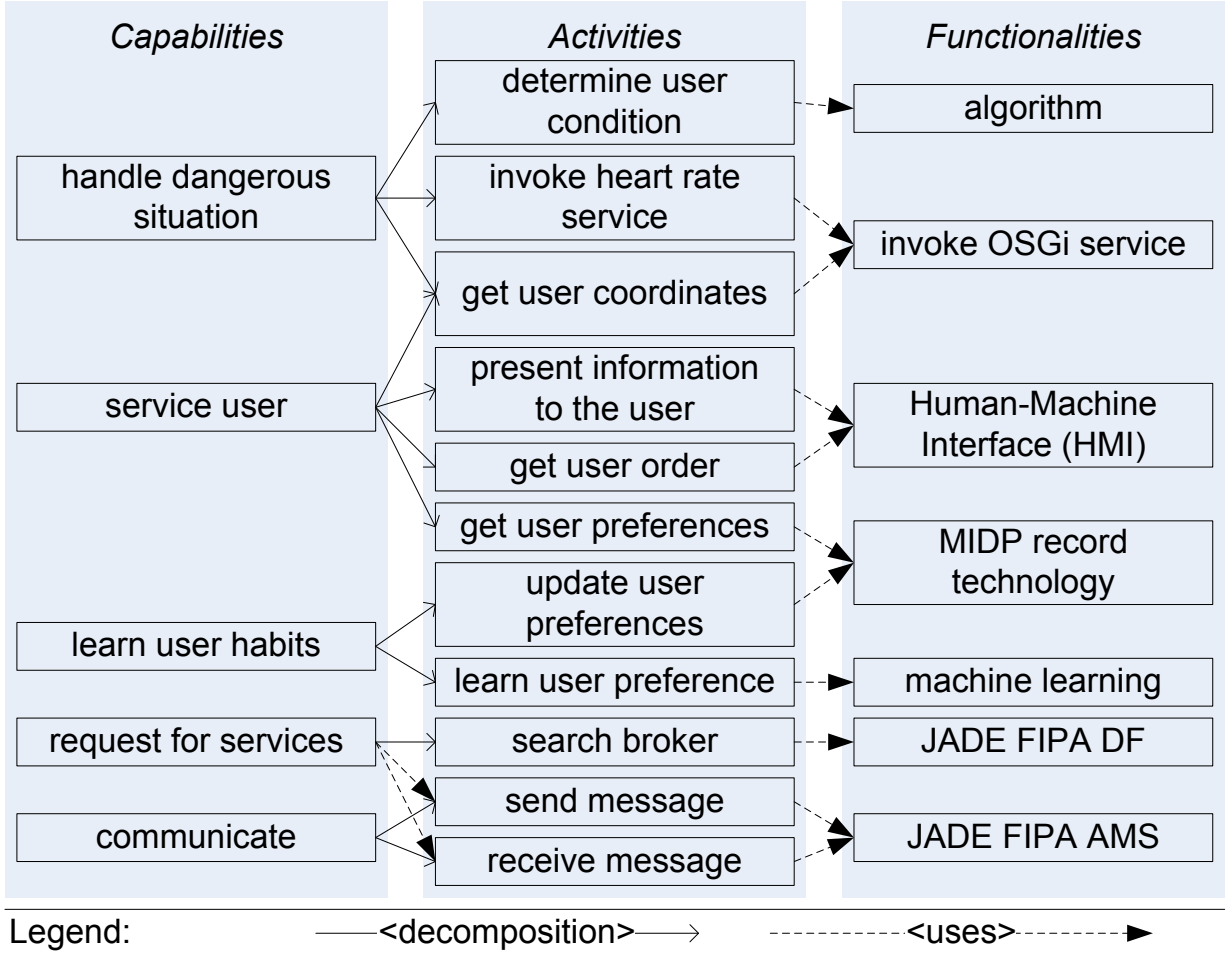
# Refine SRM



**Role:** Personal Assistant
**Capabilities and Protocols:**
manage meetings, learn user habits,
negotiate meeting date, request change meeting, request new meeting
**Activities:**
get user request, read schedule, show results,
learn user preference, update user preferences,
send change request, receive change results, send new request, receive new results,
*receive proposed date, decide response, send results, receive outcome, update schedule*
**Liveness:**
personal assistant = **(manage meetings. learn user habits)$^{\omega}$ || (negotiate meeting date)$^{\omega}$**
manage meetings = **get user request. (read schedule | request change meeting | request new meeting). show results**
learn user habits = **learn user preference. update user preferences**
request new meeting = ***send new request. receive new results***
request change meeting = ***send change request. receive change results***
negotiate meeting date = ***receive proposed date. (decide response. send results. receive outcome)+. update schedule***

# ASEME Analysis phase: Functionality Graph





# Design phase

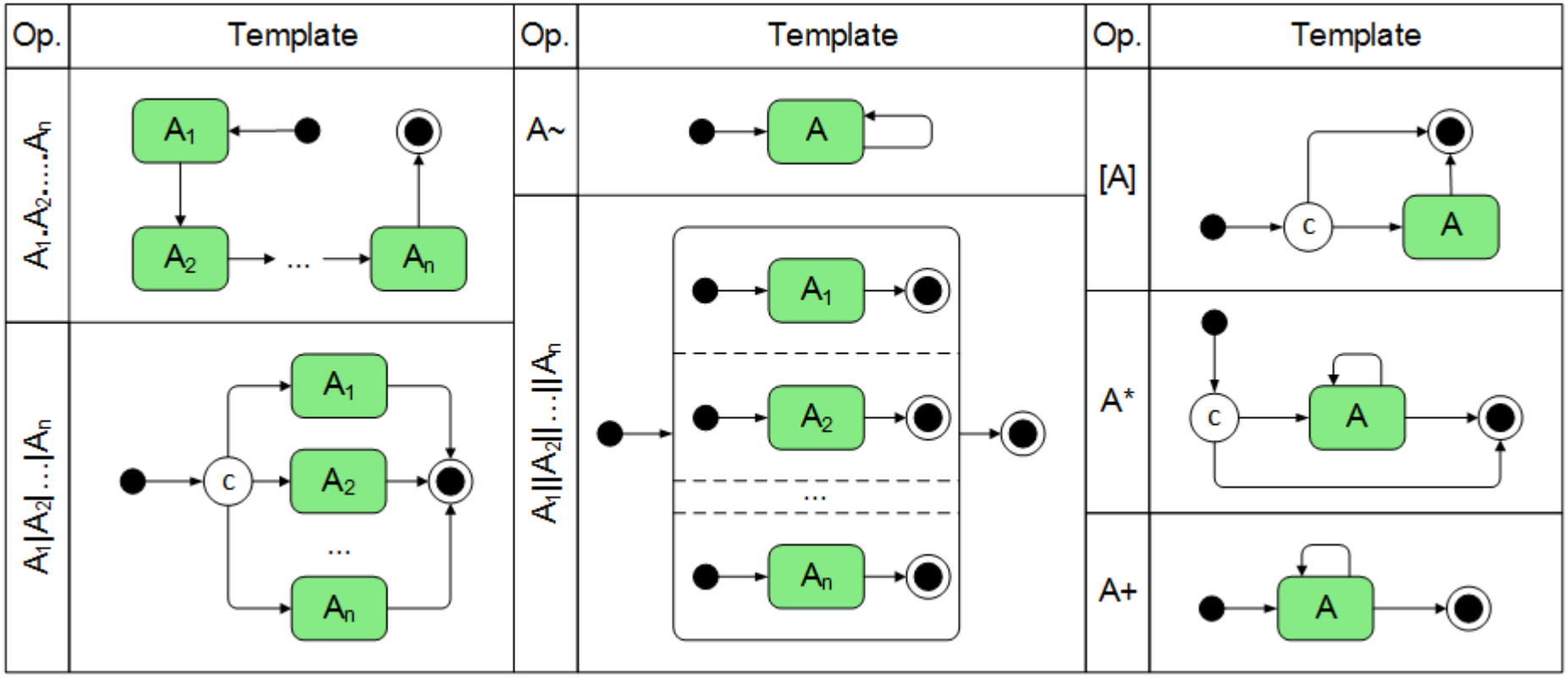


- In the design phase liveness formulas are transformed to statecharts
- Then,
  - The variables (in/out params) of each state activity are defined (corresponding to beliefs)
    - Defined as objects
  - the transition expressions are defined **e[c]/a** (corresponding to plan choosing based on beliefs)
    - Define events and conditions for enabling plan execution
    - Update beliefs

# Transformation templates



- Are applied recursively in formulas



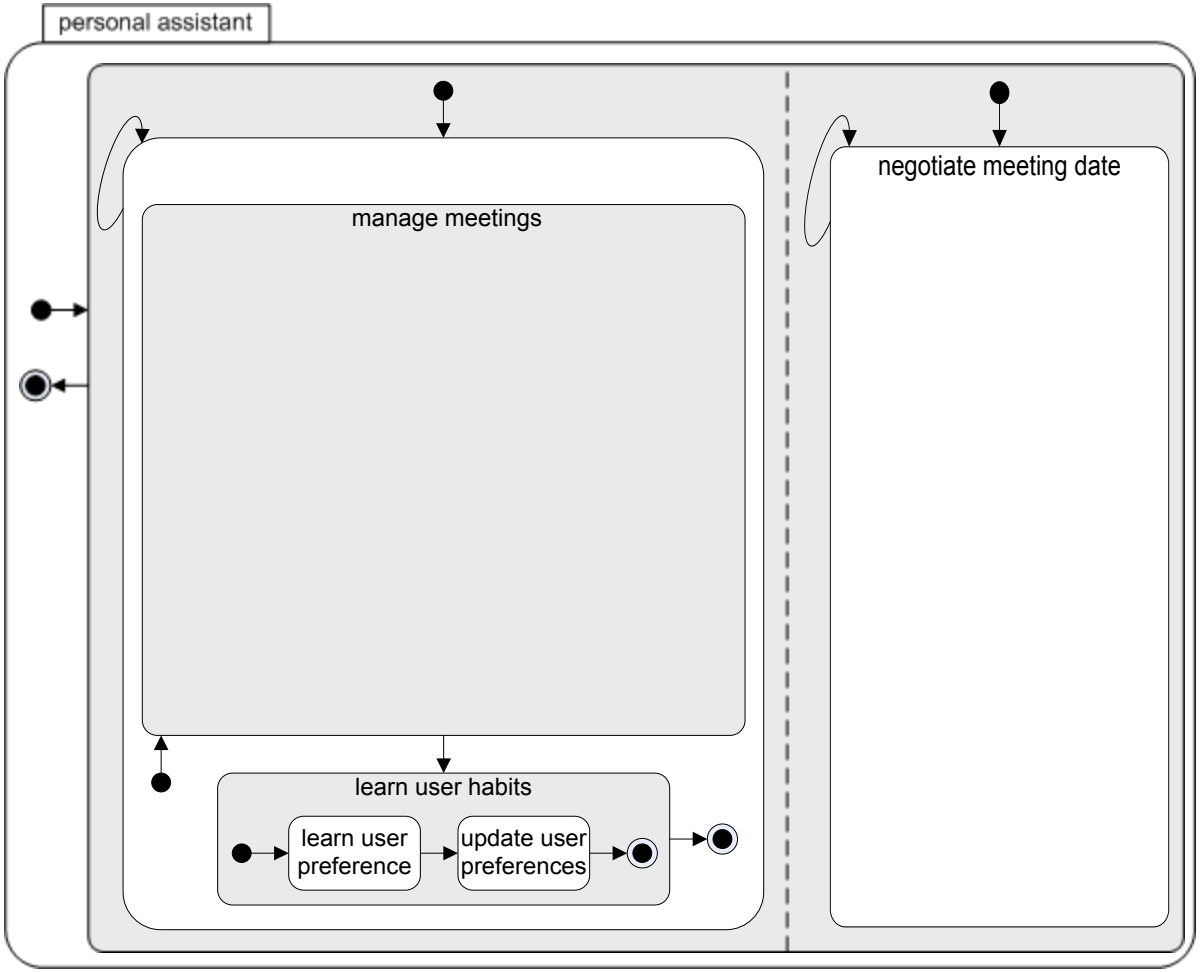
# SRM 2 IAC Transformation

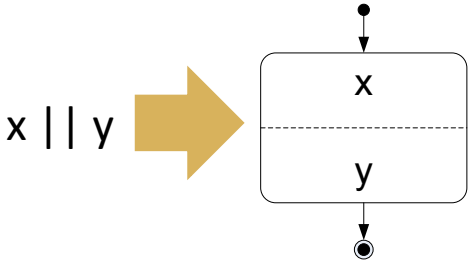


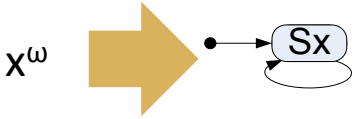
**Liveness:**

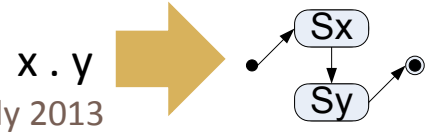
personal assistant = (manage meetings. learn user habits)$^{\omega}$ || (negotiate meeting date)$^{\omega}$

learn user habits = learn user preference. update user preferences



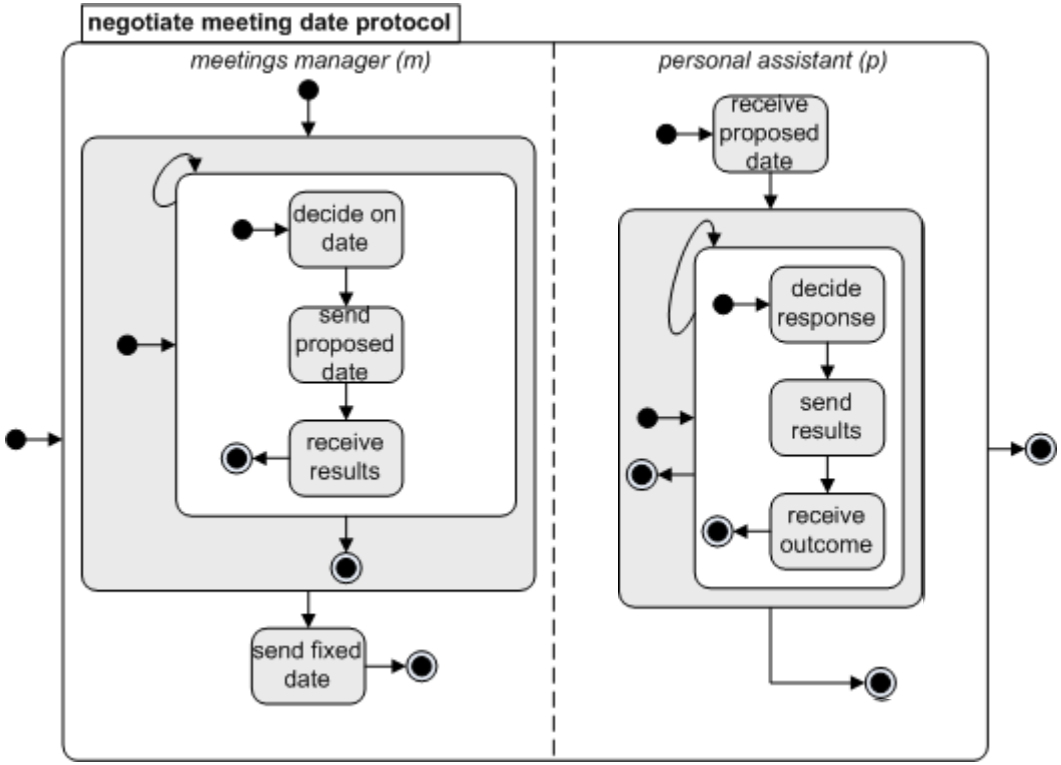
# The Inter- and Intra-Agent Control



- The inter-agent control (EAC) is a statechart defining the parallel behaviour of two or more roles
- The intra-agent control (IAC) coordinates the interactions between the agent's capabilities (or modules)
- Every role in an EAC can be merged in the IAC model as-is and it can be refined:
  - By turning a state to a superstate with substates
- IAC allows the parallel execution of multiple protocols

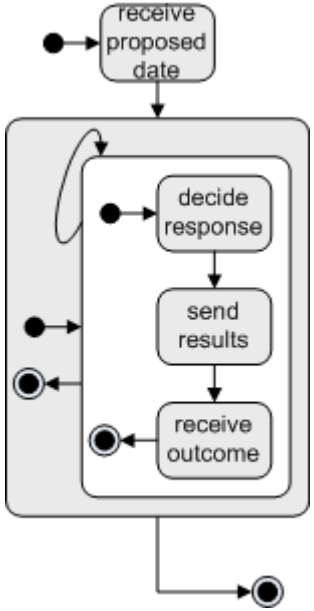
# Integrating an EAC model in an IAC model





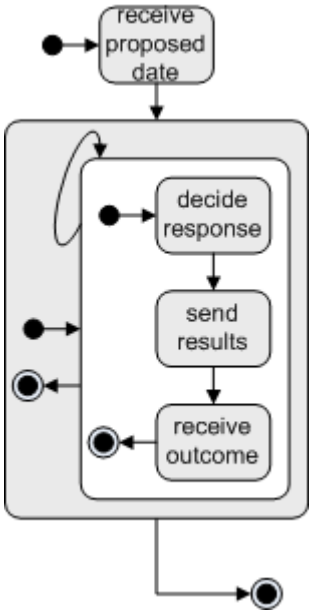
# Integrating an EAC model in an IAC model





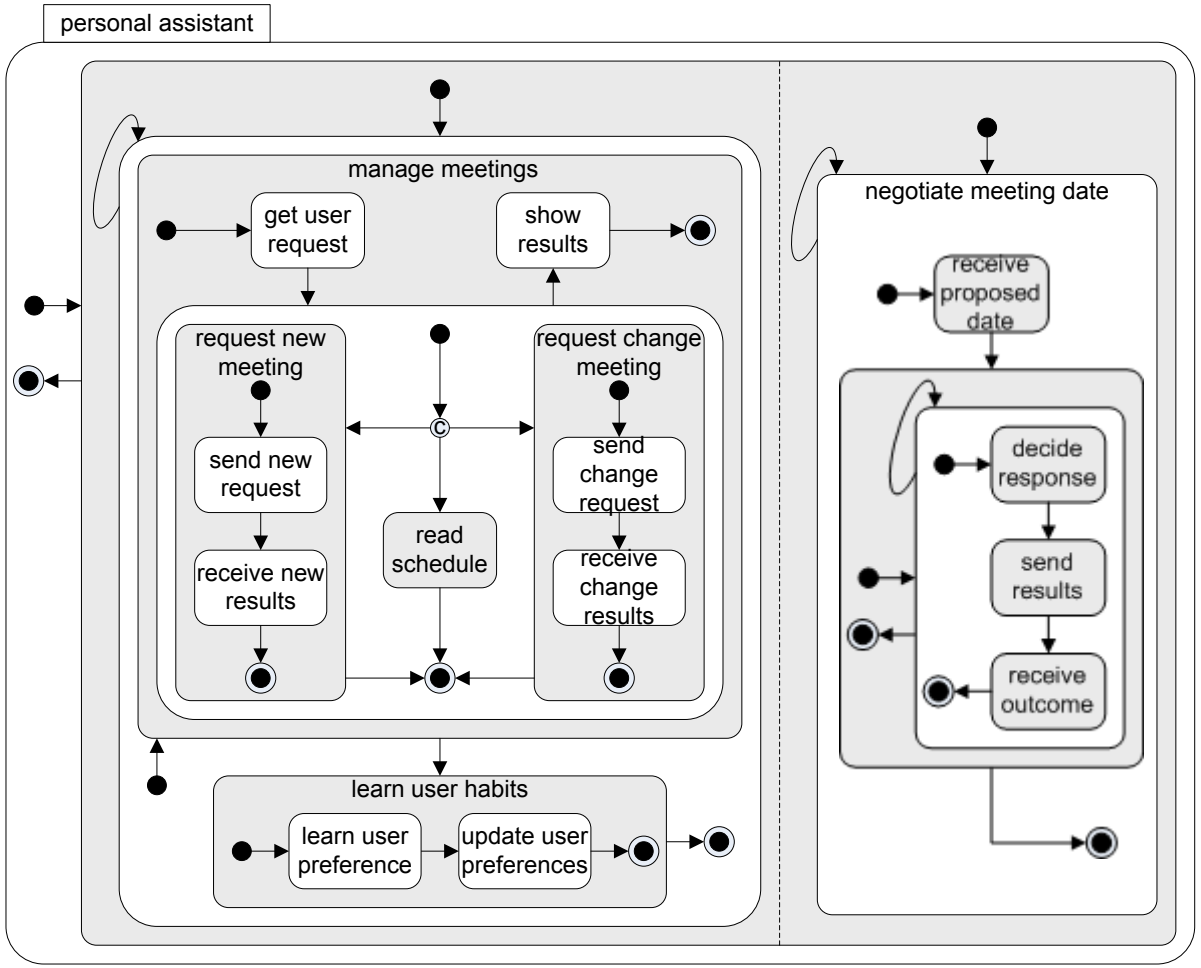
# Integrating an EAC model in an IAC model





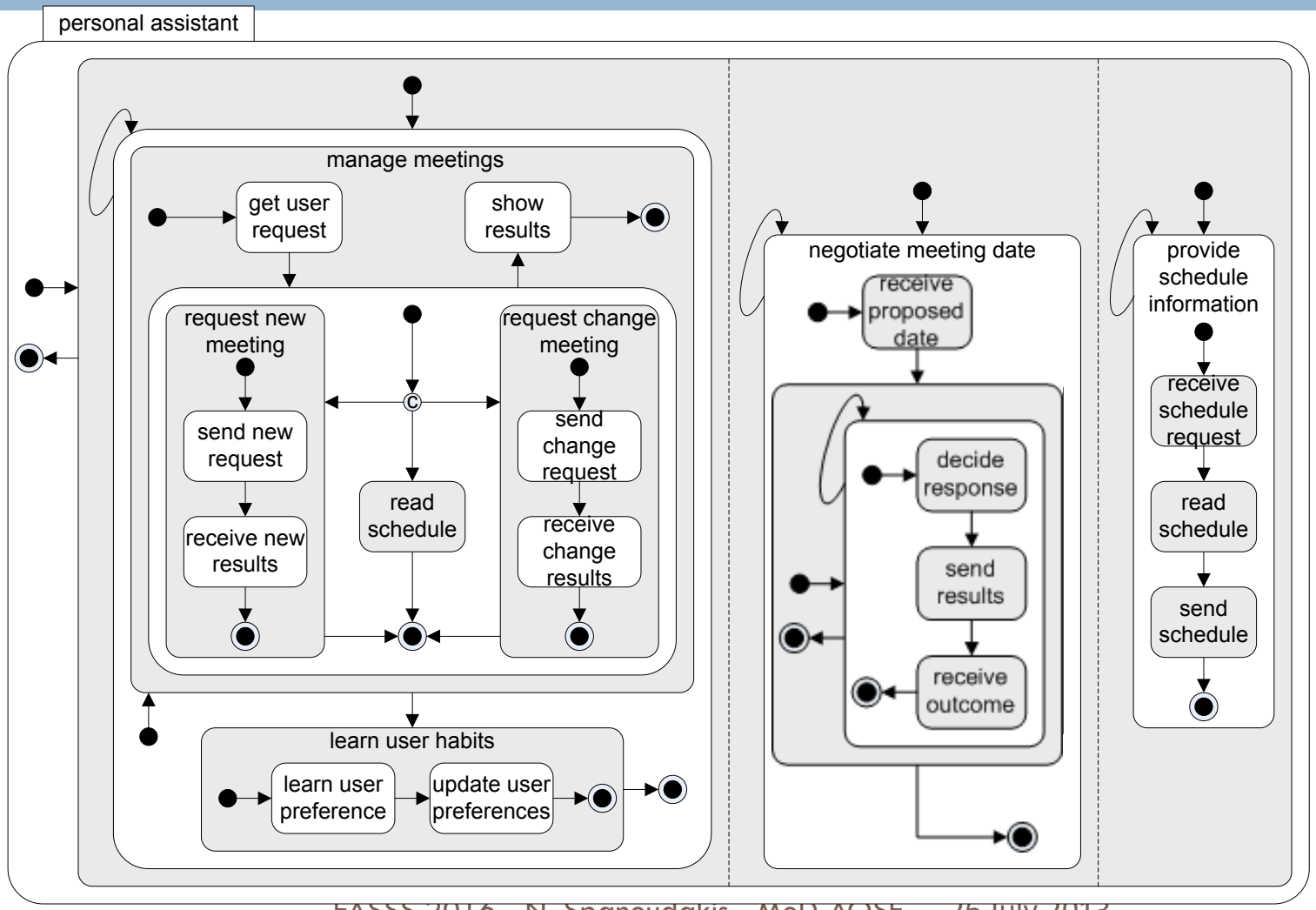
# Integrating an EAC model in an IAC model





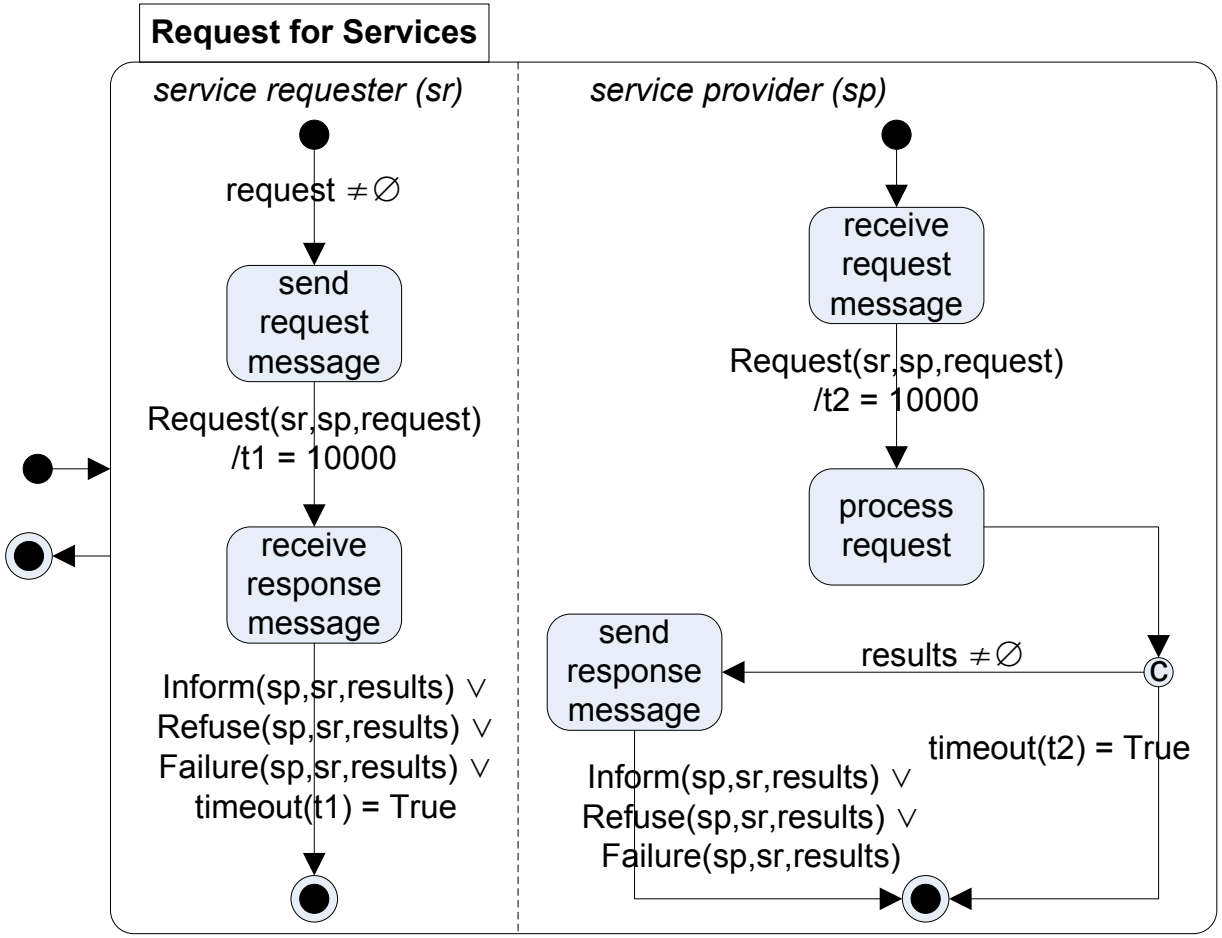
# Support for sub-dialogs





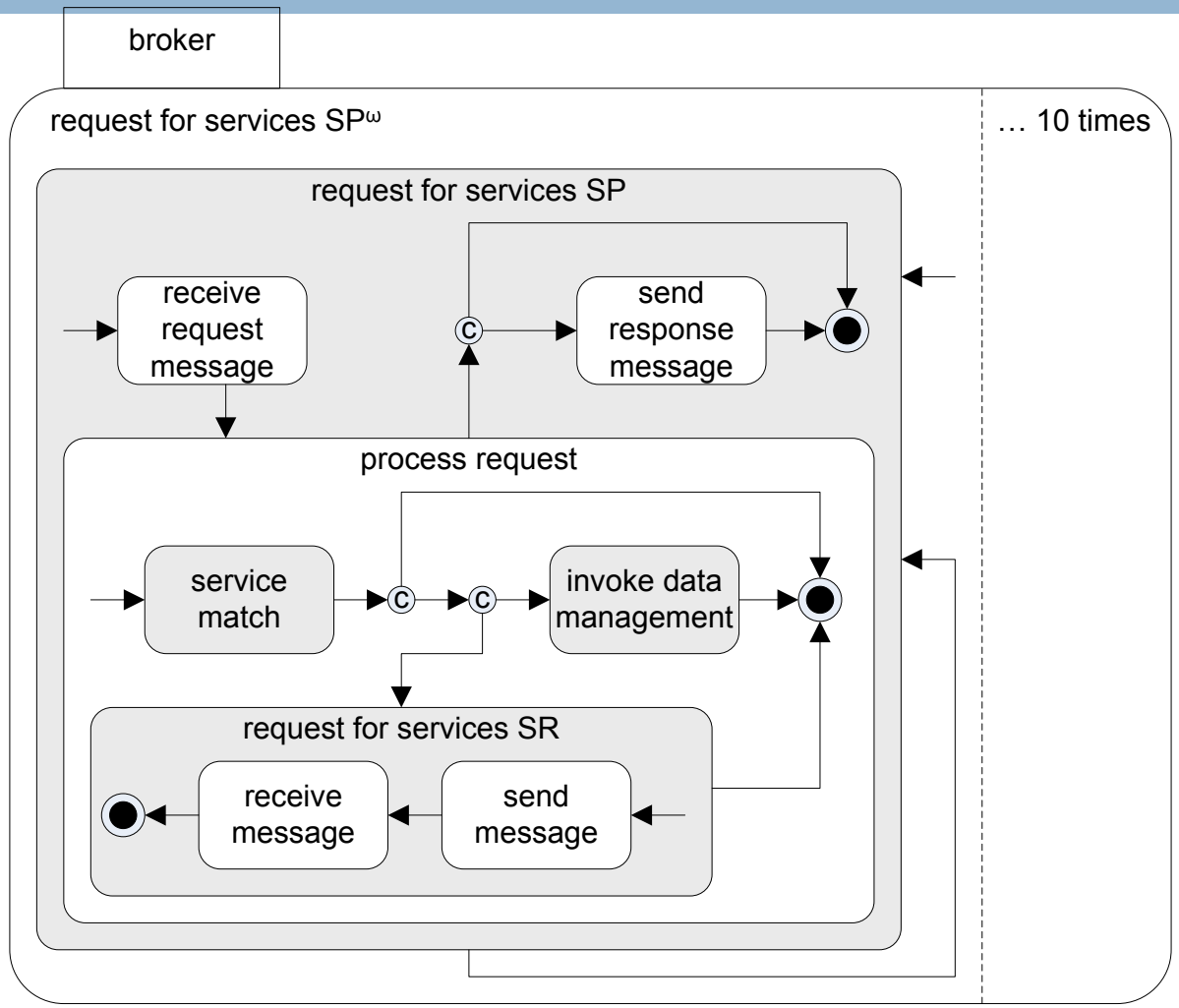
# Support for embedded dialogs





# Support for embedded dialogs





# IAC 2 JADE transformation



For each node in S
  If node is root then  create Agent class
  Else if λ(node)="BASIC" then create a SimpleBehaviour
  Else if λ(node)="AND" then create a ParallelBehaviour (|| gaia op.)
  Else if sons(node).size() = 2 and there exists transitionExpression x | (node.2, x, node.2) belongs to δ then create a CyclicBehaviour (ω gaia op.)
  Else if sons(node).size() = 3 and there exists transitionExpression x | (node.2, x, node.2) belongs to δ then create a SimpleBehaviour (+ gaia op.)
  Else if there exists x belongs to sons(node) | λ(x)=CONDITION then
    If sons(node).size() = 4 then create a SimpleBehaviour (* gaia op.)
    Else create a SequentialBehaviour (| gaia op.)
    End if
  Else create a SequentialBehaviour (. gaia op.)
  End if
End for

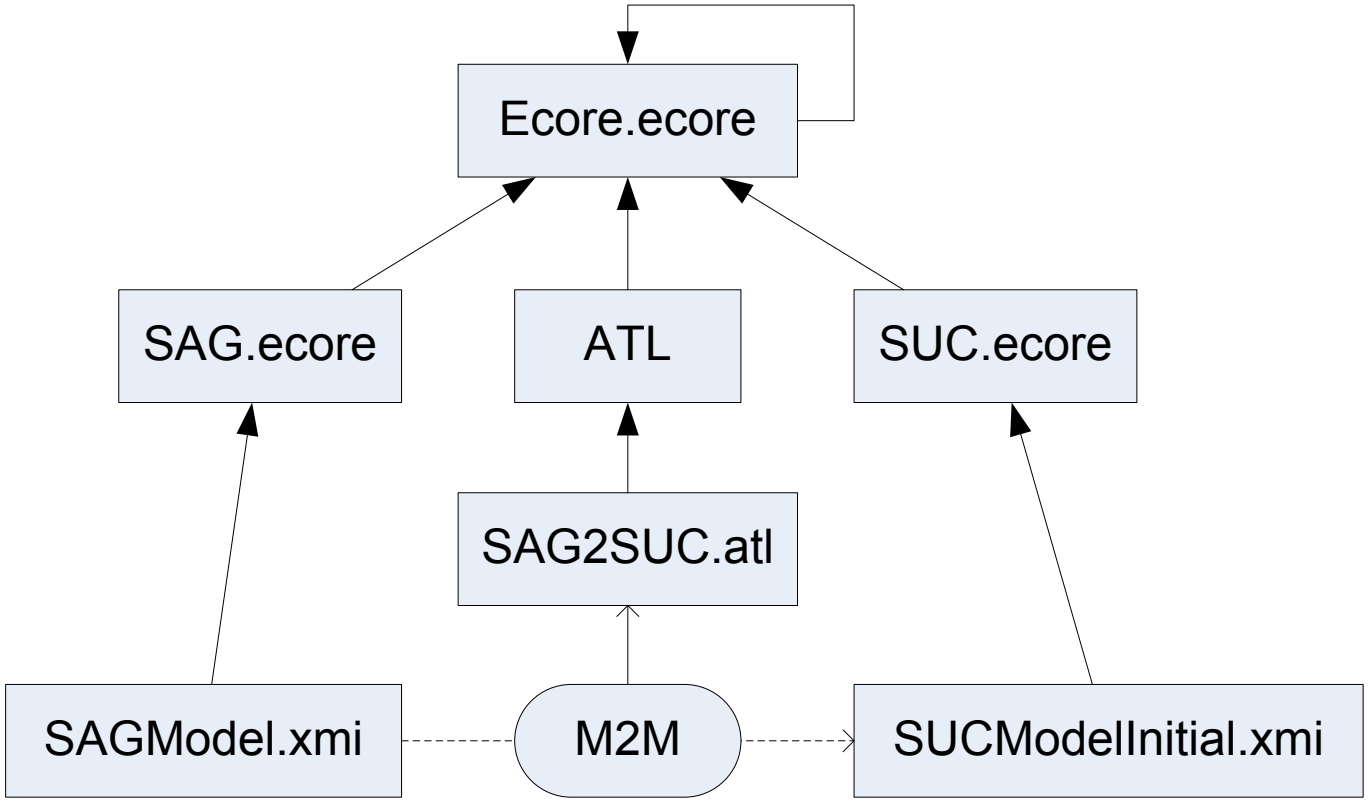
# Tooling

We have three types of tools:

- Graphical editors
- Transformations
- Integrated Development Environment

# Transformations using ATL






XMI: XML Metadata Interchange

# M2M transformation using ATL (Jouault & Kurtev, 2006)

ATL - ASEME_ATL/ATL_files/SAG2SUC.atl - Eclipse Platform

File Edit Navigate Search Project Run Compatibility ATL Editor Window Help

Navigator

- ASEME_ATL
  - .settings
  - ATL_files
    - SAG2SUC.asm
    - SAG2SUC.atl
    - SUC2SRM.asm
    - SUC2SRM.atl
  - metamodels
    - IAC.ecore
    - IAC.ecore_diagram
    - IAC.emf
    - SAG.ecore
    - SAG.ecore_diagram
    - SRM.ecore
    - SUC.ecore
    - SUC.ecore_diagram
  - models
    - SAGModel.xmi
    - SRMModelInitial.xmi
    - SRMModelRefined.xmi
    - SUCModelInitial.xmi
    - SUCModelRefined.xmi
  - .project

SAG2SUC.atl

```
module SAG2SUC; -- Module Template
create OUT : SUC from IN : SAG;

rule Goal2UseCase {
 from
  i : SAG!Goal
 to
  o_1 : SUC!UseCase (
   name <- i.name,
   specified_by <- i.requirements,
   interacter <- i.depender,
   interacter <- i.dependee
  )
}

rule Actor2Role {
 from
  i : SAG!Actor
 to
  o_n : SUC!Role (
   name <- i.name
  )
}
```

Outline

- SAG2SUC : Module
  - OUT : OclModel
  - IN : OclModel
  - Goal2UseCase : MatchedRule
  - Actor2Role : MatchedRule

Problems | Error Log | Console | Properties

| Property | Value |
|---|---|

# Code generation using EMF Xpand



```
«DEFINE javaClass FOR IAC::Model»
  «LET name AS packageName»
    «EXPAND nodeClass(packageName, this) FOREACH nodes»
    «EXPAND variableHolderClass(packageName, this) FOREACH variables»
  «ENDLET»
«ENDDEFINE»

«DEFINE nodeClass(String packageName, Model model) FOR IAC::Node»
  «IF label.compareTo("0")==0»
    «EXPAND agentClass(packageName, model) FOR this»
  «ELSE»
    «EXPAND behaviourClass(packageName, model) FOR this»
  «ENDIF»
«ENDDEFINE»

«DEFINE agentClass(String packageName, Model model) FOR IAC::Node»
  «FILE classFileName()»
    package «packageName»;
    import jade.core.Agent;
    public class «className()» extends Agent{
      public void setup(){
        //add behavior
        addBehaviour(«getAgentBehaviour(this,model)»);
      }
      protected void takeDown() {
        doDelete();
      }
    }
  «ENDFILE»
«ENDDEFINE»
```

# JADE source code generation



openArchitectureWare - IAC_EMF.generator/src-gen/fr/parisdescartes/mi/meetingsmanagement/SendResultsBehaviour.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer | Hierarchy

- IAC_EMF.generator
  - src
  - src-gen
    - fr.parisdescartes.mi.meetingsmanagement
      - _open_group_DecideResponse_sequence_SendResul
      - _open_group_DecideResponse_sequence_SendResul
      - _open_group_ManageMeetings_sequence_LearnUse
      - _open_group_ManageMeetings_sequence_LearnUse
      - _open_group_ManageMeetings_sequence_LearnUse
      - _open_group_ManageMeetings_sequence_LearnUse
      - _open_group_ReadSchedule_or_RequestChangeMee
      - ACLMessageHolder.java
      - DecideResponseBehaviour.java
      - GetUserRequestBehaviour.java
      - LearnUserHabitsBehaviour.java
      - LearnUserPreferenceBehaviour.java
      - ManageMeetingsBehaviour.java
      - MeetingHolder.java
      - NegotiateMeetingDate_forever__parallel_Behaviour.j
      - NegotiateMeetingDate_forever_Behaviour.java
      - NegotiateMeetingDateBehaviour.java
      - PersonalAssistantAgent.java
      - ReadScheduleBehaviour.java
      - ReceiveChangeResultsBehaviour.java
      - ReceiveNewResultsBehaviour.java
      - ReceiveOutcomeBehaviour.java
      - ReceiveProposedDateBehaviour.java
      - RequestChangeMeetingBehaviour.java
      - RequestNewMeetingBehaviour.java
      - SendChangeRequestBehaviour.java
      - SendNewRequestBehaviour.java
      - SendResultsBehaviour.java
      - ShowResultsBehaviour.java
      - UpdateScheduleBehaviour.java
      - UpdateUserPreferencesBehaviour.java
    - path.properties
  - JRE System Library [J2SE-1.5]

SendResultsBehaviour.java

```
package fr.parisdescartes.mi.meetingsmanagement;

import jade.core.Agent;

public class SendResultsBehaviour extends SimpleBehaviour {

    MeetingHolder e = null;
    ACLMessageHolder accept = null;
    ACLMessageHolder reject = null;

    boolean finished = false;

    public SendResultsBehaviour(Agent a, MeetingHolder e,
            ACLMessageHolder accept, ACLMessageHolder reject) {
        super(a);
        this.e = e;
        this.accept = accept;
        this.reject = reject;
    }

    public void action(){
            ACLMessage msg = null;
        if (/*insert condition*/) {
            msg = new ACLMessage(ACLMessage.ACCEPT);
        }
        else if (/*insert condition*/) {
            msg = new ACLMessage(ACLMessage.REJECT);
        }
        //insert message initialization code
        myAgent.send(msg);
        finished = true;
            }
    public boolean done() {
        return finished;
    }

}
```

Problems | Javadoc | Declaration | Console | Error Log

# Automatic code generation

- Targeting three platforms (Spanoudakis and Moraitis, 2010; Paraschos et al., 2012)

| **Project – Case study** | **ASK-IT** | **MARKET-MINER** | **Robocup player** |
|---|---|---|---|
| *Implementation Platform* | *JADE* | *Rhapsody* | *Monas* |
| Total project code | 160,241 (bytes) | 179,750 (bytes) | 826 (lines) |
| Automatically generated (bytes) | 41,759 (bytes) | 97,631 (bytes) | 805 (lines) |
| Manually written Java code (bytes) | 118,482 (bytes) | 60,365 (bytes) | 21 (lines) |
| Manually written non-java code (bytes) | - | 21,754 | - |
| **Percentage of automatically generated code** | **26.06%** | **54.31%** | **97.45%** |

# More aspects of the truth…

**Table 2: Comparison of development methods.**

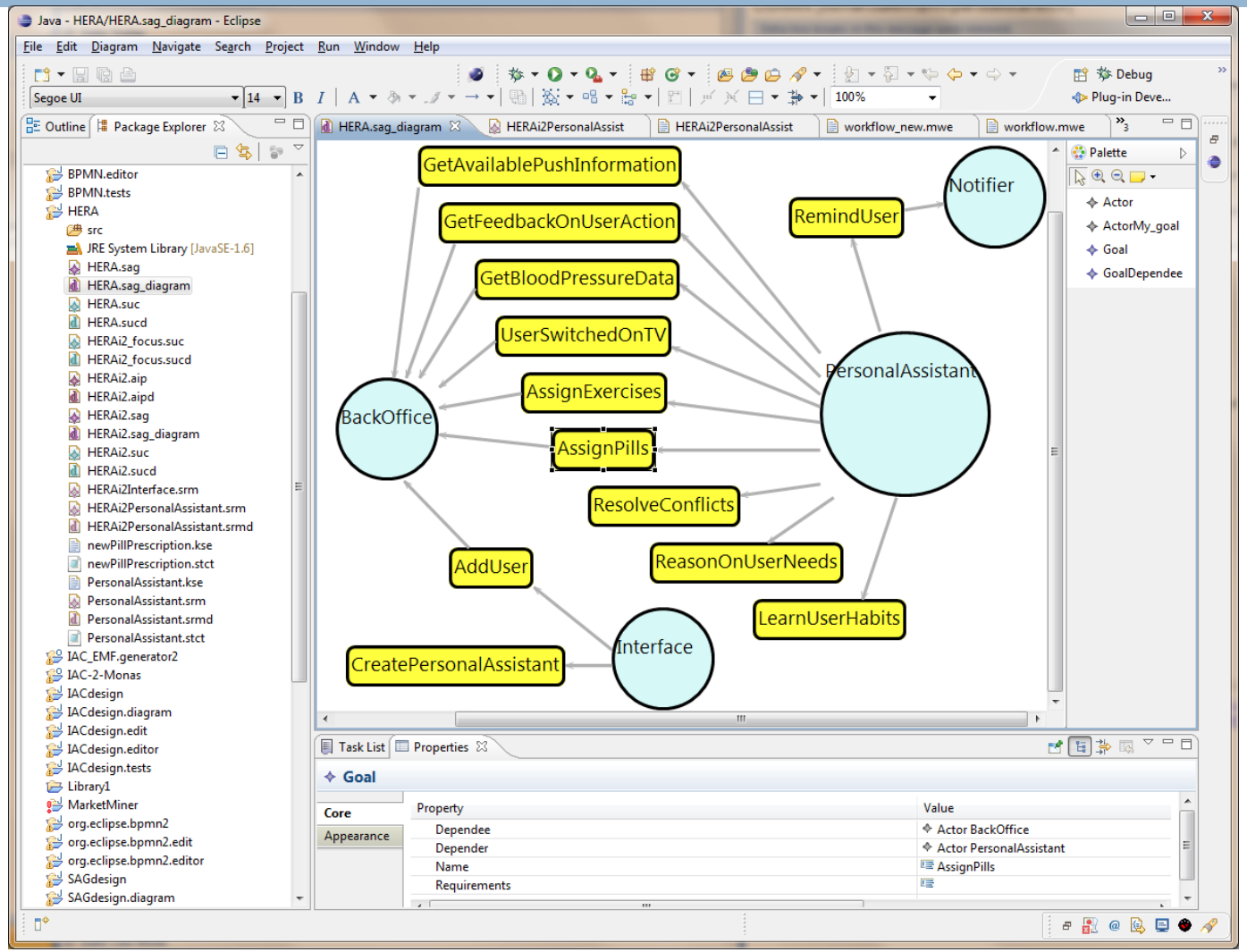
| Metric / Concept | NaoQi | ASEME |
|---|---|---|
| Run-Time Performance (load) | 1.3 | 0.8 |
| Development Phase | | |
| Total Development Time | 8 hours | 5 hours |
| Lines of Source Code | 490 | 826 |
| Auto-Generated Lines | N/A | 805 |
| Number of State Variables | 18 | N/A |
| Debugging Phase | | |
| Total Debugging Time | 12 hours | 5 hours |
| Number of Bugs | 16 | 4 |

(Paraschos et al., 2012)

# SAG model graphical editor



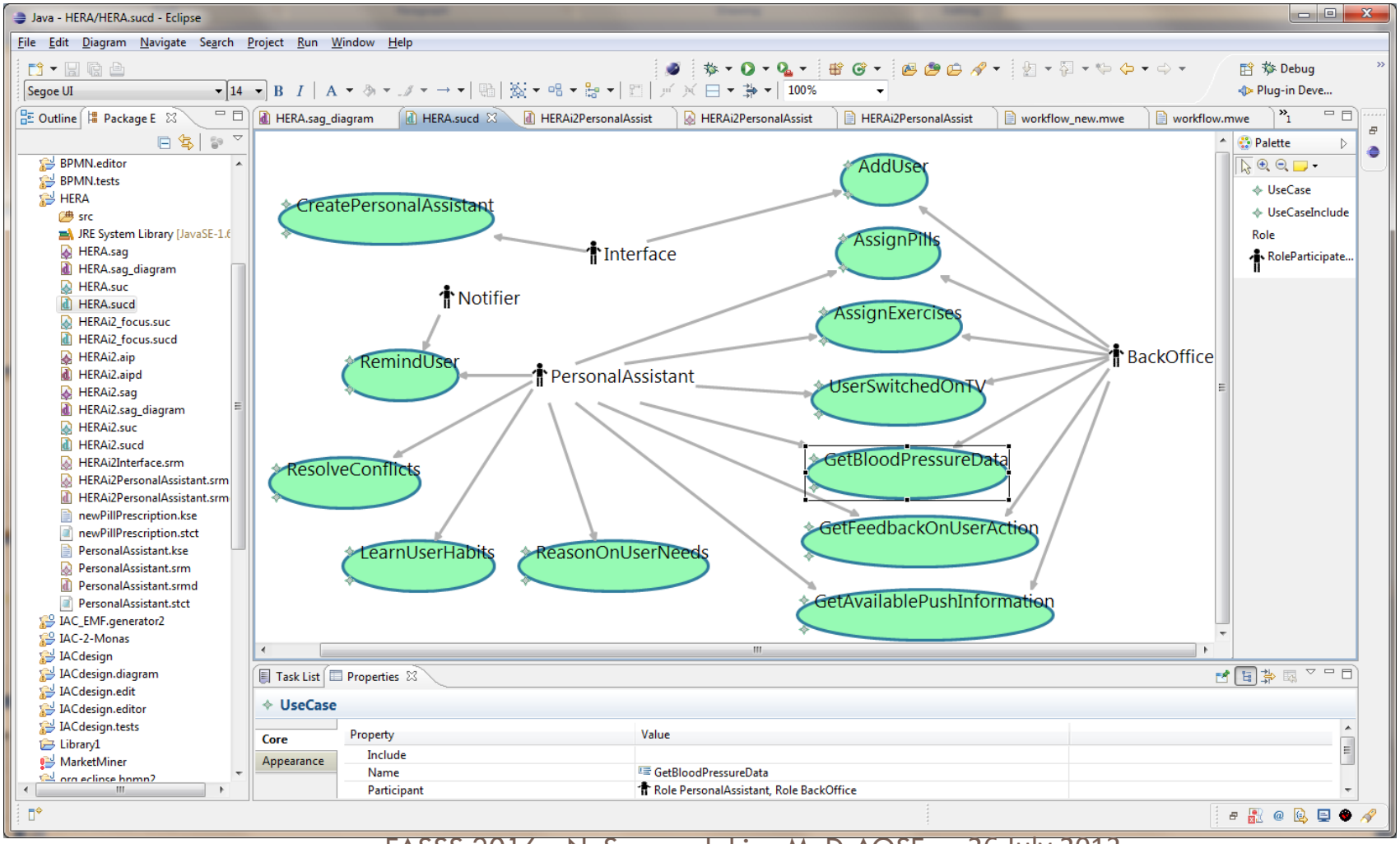
- Using EMF Graphical Modeling Framework (GMF)




(Argyriou, 2010)

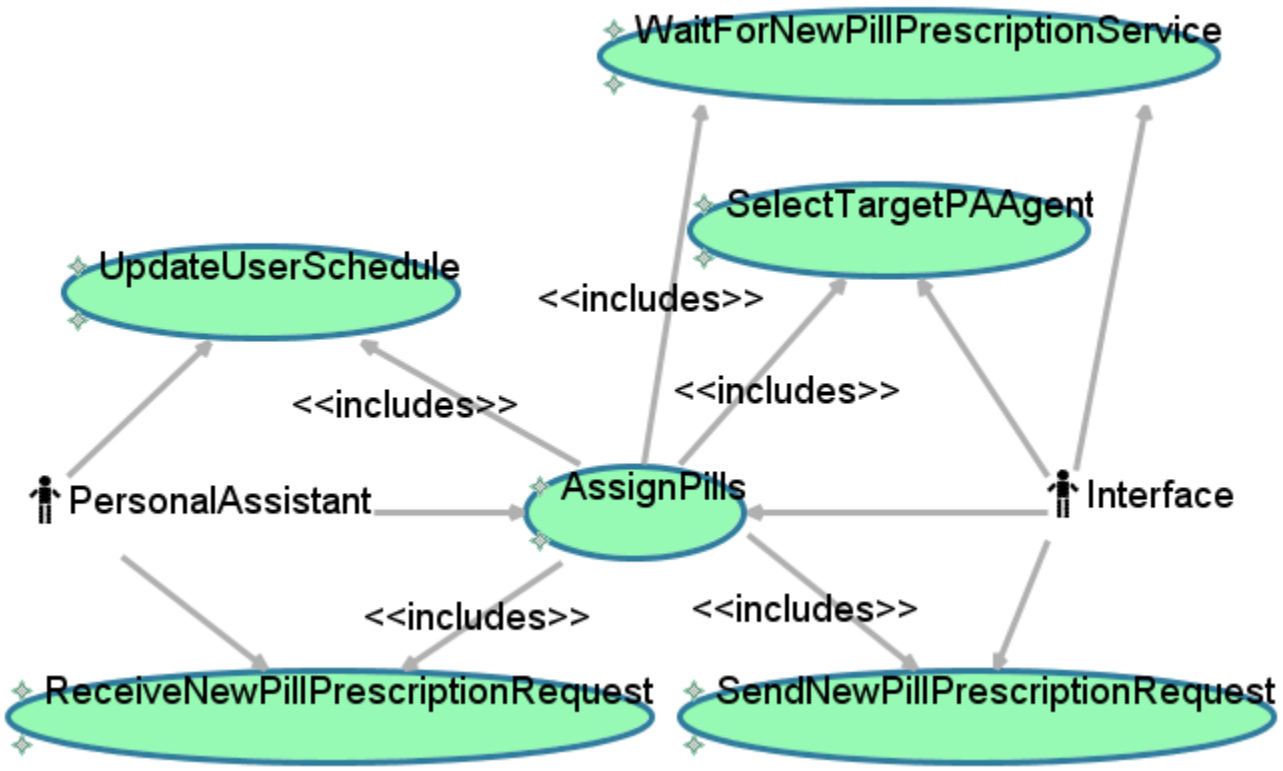
# SUC model graphical editor init





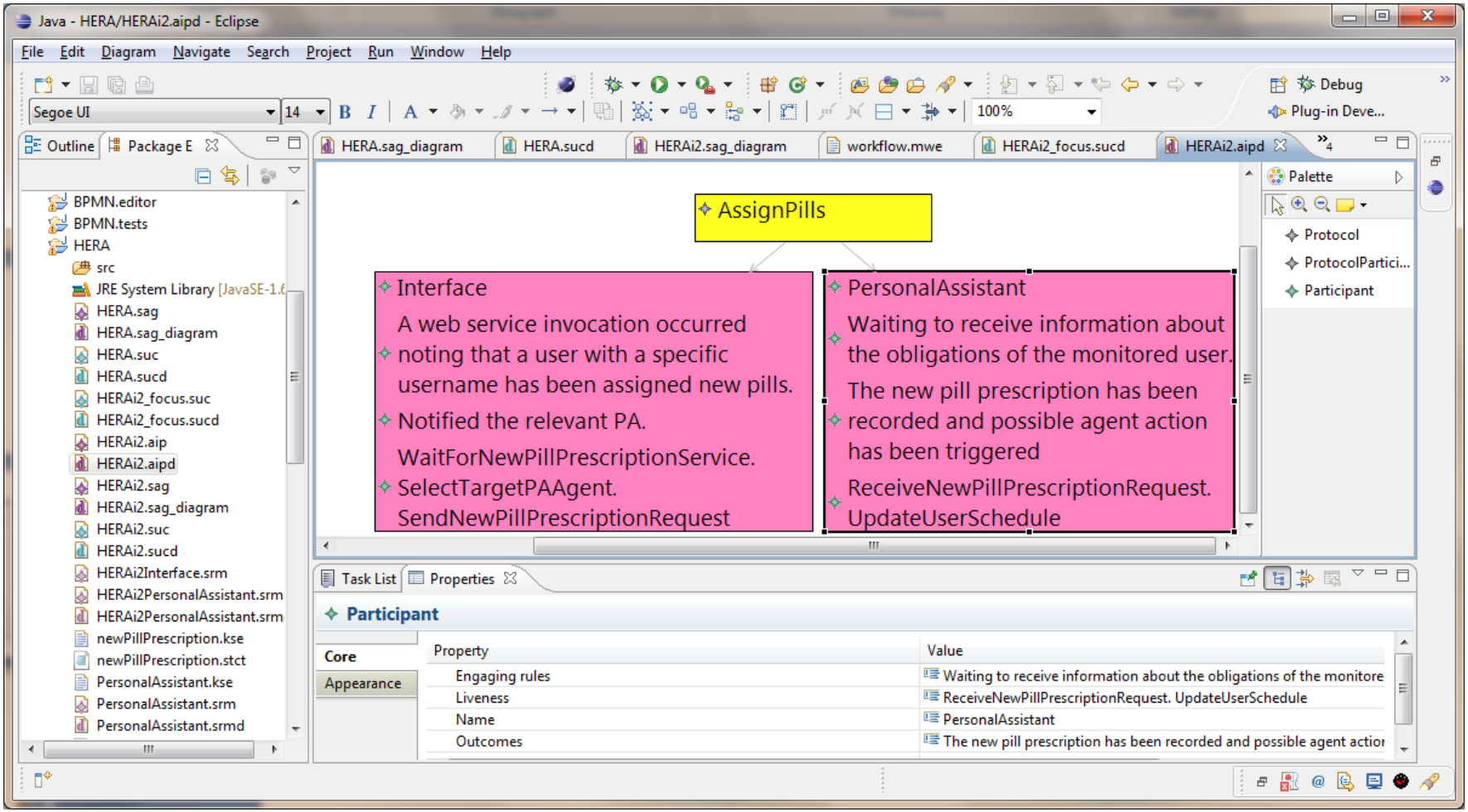
# SUC: Refining Use Cases





# AIP model

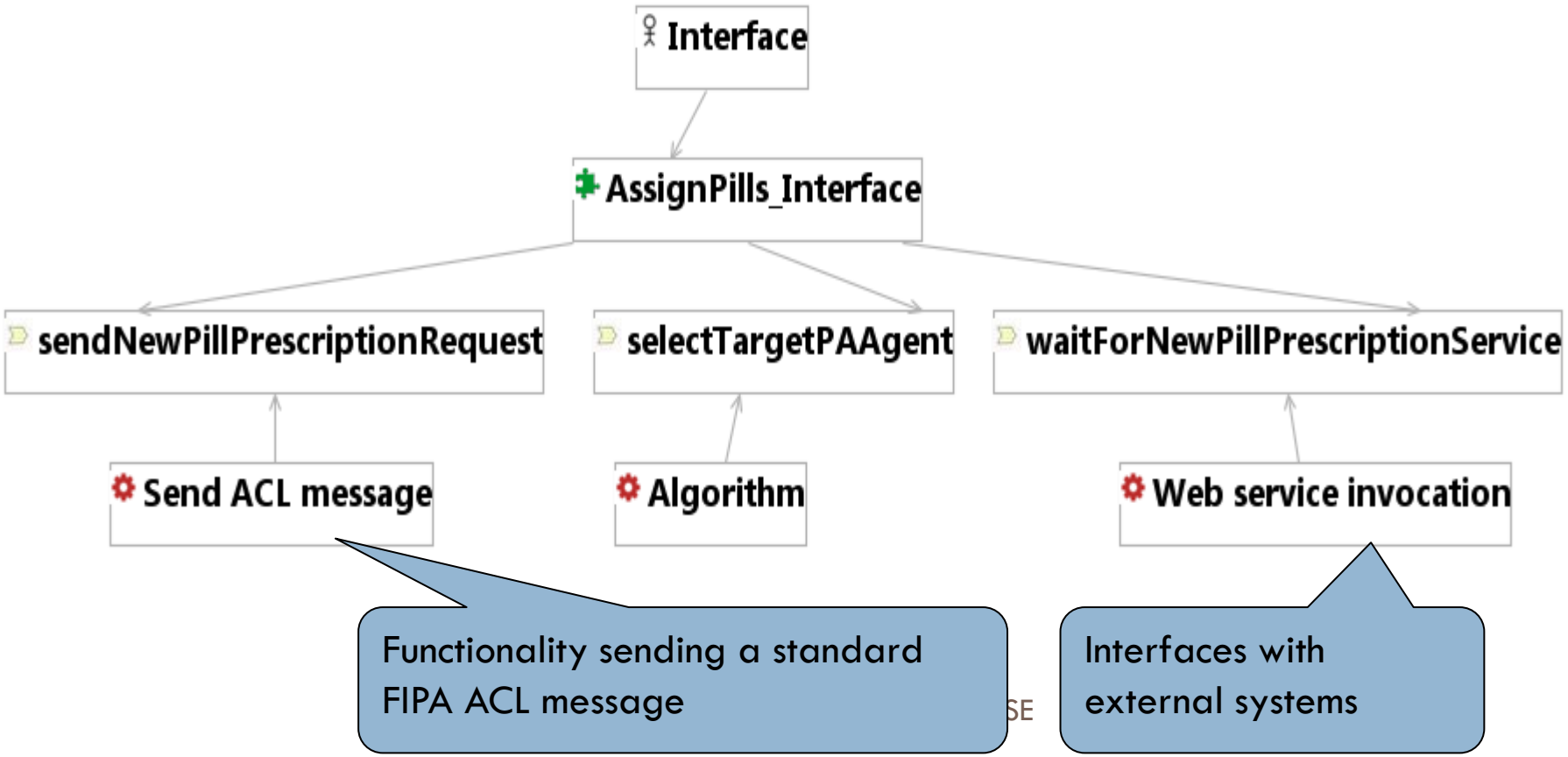



Java - HERA/HERAi2.aipd - Eclipse
File Edit Diagram Navigate Search Project Run Window Help
Outline
Package E
BPMN.editor
BPMN.tests
HERA
src
JRE System Library [JavaSE-1.6
HERA.sag
HERA.sag_diagram
HERA.suc
HERA.sucd
HERAi2_focus.suc
HERAi2_focus.sucd
HERAi2.aip
HERAi2.aipd
HERAi2.sag
HERAi2.sag_diagram
HERAi2.suc
HERAi2.sucd
HERAi2Interface.srm
HERAi2PersonalAssistant.srm
HERAi2PersonalAssistant.srm
newPillPrescription.kse
newPillPrescription.stct
PersonalAssistant.kse
PersonalAssistant.srm
PersonalAssistant.srmd
HERA.sag_diagram
HERA.sucd
HERAi2.sag_diagram
workflow.mwe
HERAi2_focus.sucd
HERAi2.aipd
AssignPills
Interface
A web service invocation occurred noting that a user with a specific username has been assigned new pills.
Notified the relevant PA.
WaitForNewPillPrescriptionService.
SelectTargetPAAgent.
SendNewPillPrescriptionRequest
PersonalAssistant
Waiting to receive information about the obligations of the monitored user.
The new pill prescription has been recorded and possible agent action has been triggered
ReceiveNewPillPrescriptionRequest.
UpdateUserSchedule
Palette
Protocol
ProtocolPartici...
Participant
Task List
Properties
Participant
Core
Appearance
Property
Value
Engaging rules
Waiting to receive information about the obligations of the monitore
Liveness
ReceiveNewPillPrescriptionRequest. UpdateUserSchedule
Name
PersonalAssistant
Outcomes
The new pill prescription has been recorded and possible agent actior

# A graphical view of SRM



- The functionality graph



# How to evaluate a methodology?

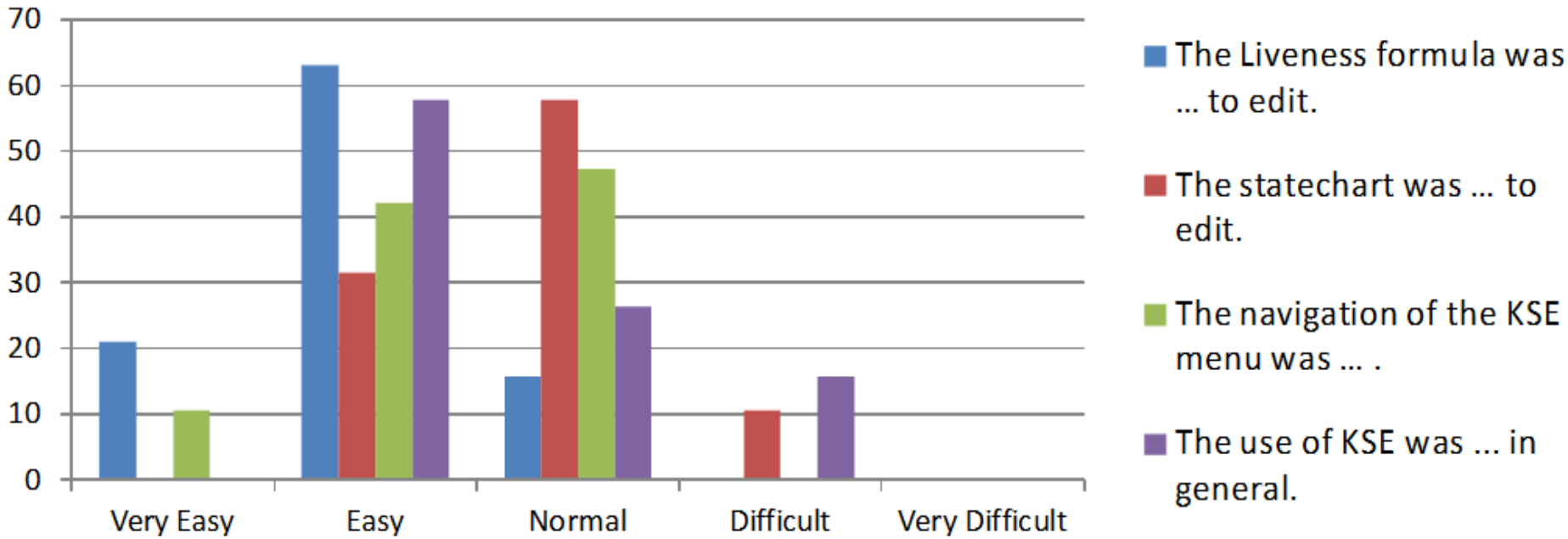


- It supports new features
- It is evaluated against other successful ones over a set of criteria (danger of bias)
- Through a successful case study for a real world system
- Compare it against others through a benchmark or common application (conference management system)
- De facto - Through acceptance by practitioners

# KSE Empirical Evaluation



- During a lab session 28 undergraduate students used KSE to design an attacker behavior for RoboCup SPL.
- All of them completed the task and evaluated KSE positively.



# ASEME usage



- Built systems
  - The MarketMiner Product Pricing Agent (Spanoudakis and Moraitis, 2009)
  - A Hybrid Wind Turbine Monitoring System (Smarsly and Hartmann, 2010)
  - The HERA Ambient Assisted Living System (Marcais et al., 2011)
  - Robotic Behaviors for the RoboCup Competition (Kouretes TDP 2011; 2012; 2013)
  - A Structural Health Monitoring System (Smarsly and Law, 2012)
- Has been extended for Application Domains
  - Agent-Oriented Monitoring Systems Engineering (AGEME) (Smarsly and Hartmann, 2010)
  - Kouretes Statechart Editor (KSE) (Topalidou-Kyniazopoulou et al., 2012)

ASEME
AGENT SYSTEMS ENGINEERING METHODOLOGY

HERA

KSE

# Method Engineering

Where a team customizes the software process

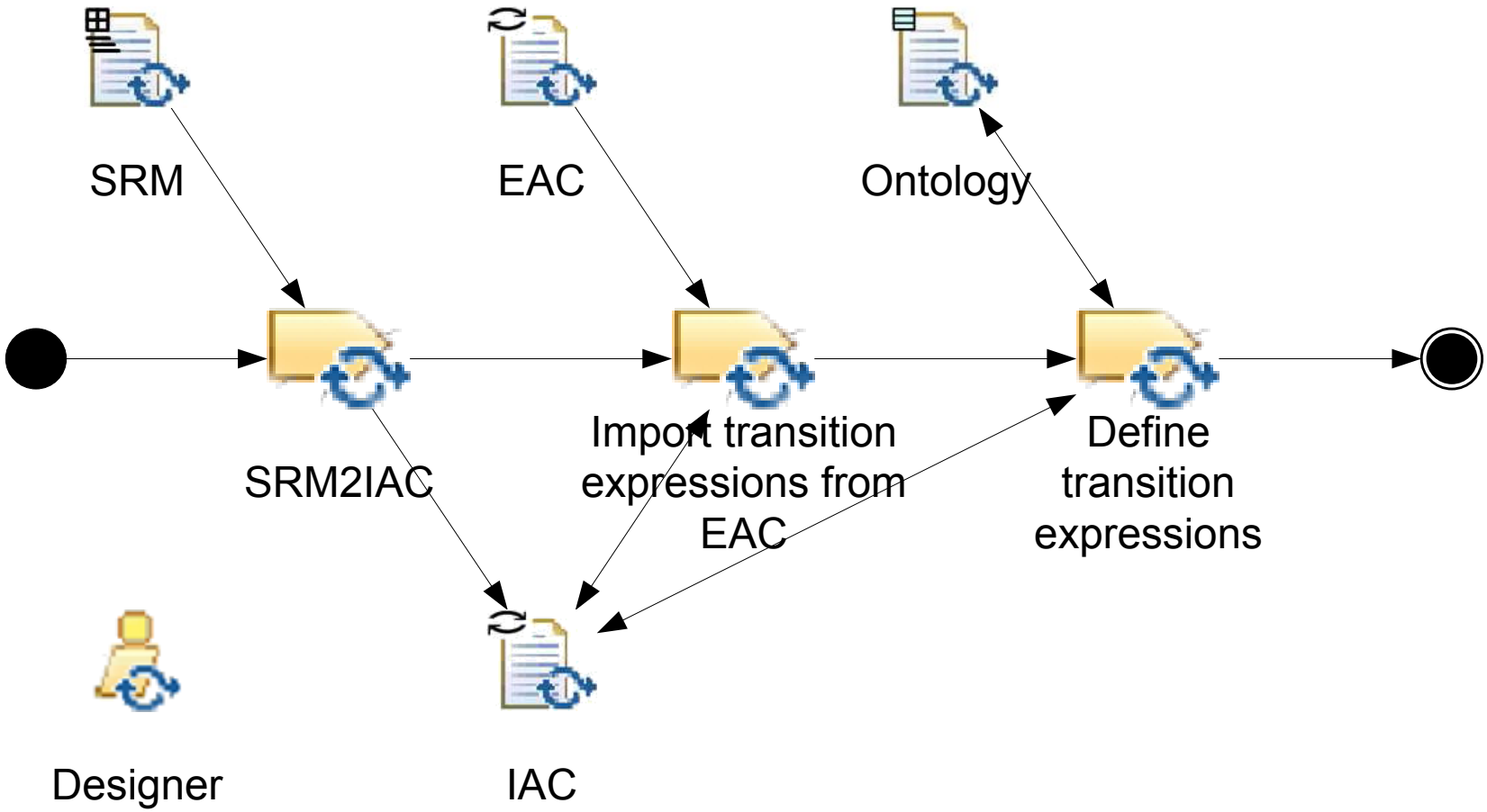
# Method Engineering



- Different methodologies have their strengths and weaknesses
- An engineer can need to combine method fragments from different methodologies
- Method fragments are reusable methodological parts that can be used by engineers in order to produce a new design process for a specific situation (Cossentino et al., 2007)
- This allows a development team to come up with a hybrid methodology that will support the needs of specific programming and modeling competencies

# Method Fragment

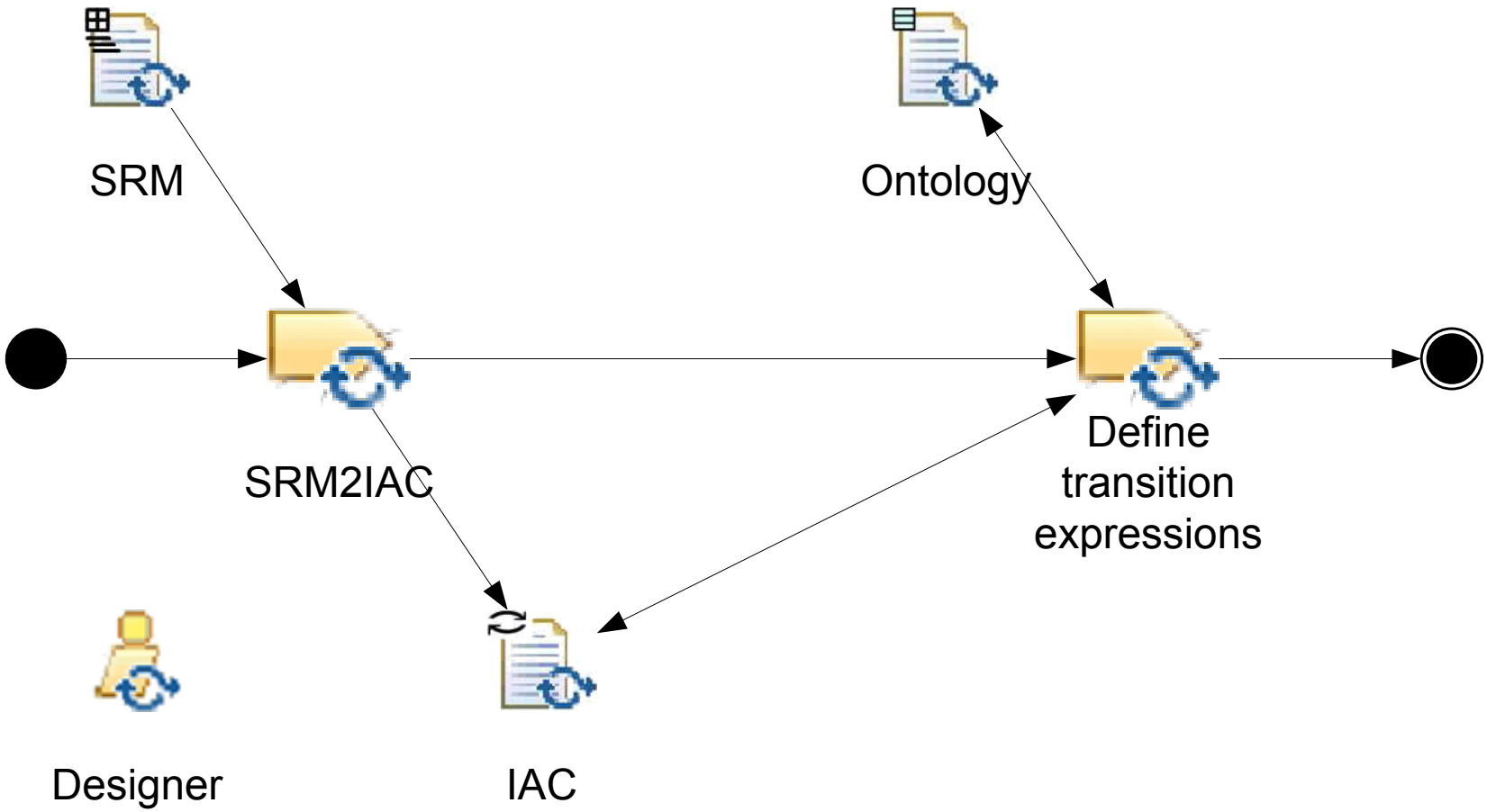





- Transform the SRM analysis model to the IAC design model

# Method Fragment - single agent

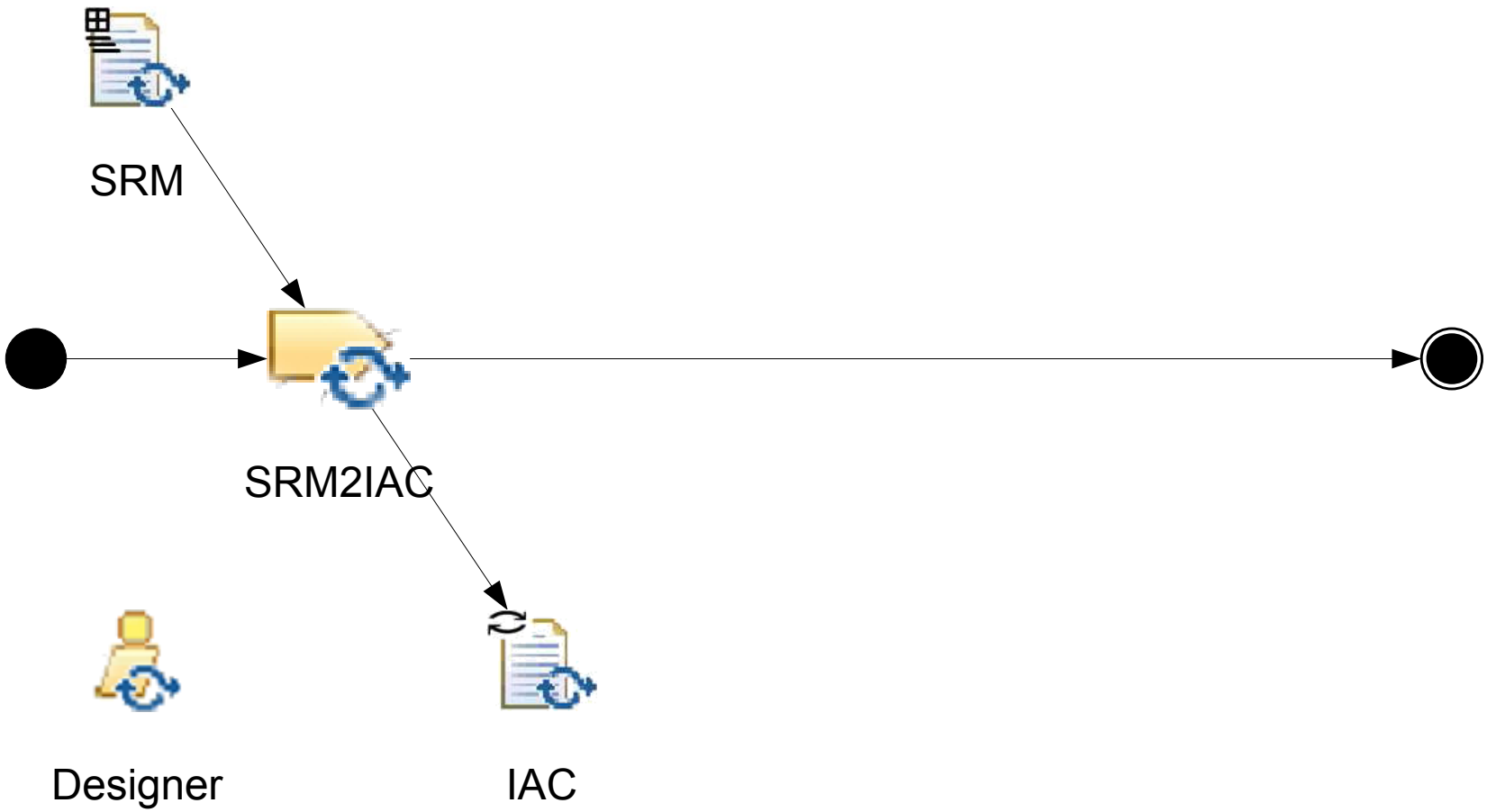





- Transform the SRM analysis model to the IAC design model

# Method Fragment - only process






- Transform the SRM analysis model to the IAC design model

# How to combine fragments?



- Define the scenario
- Define input/output models
- Apply transformations

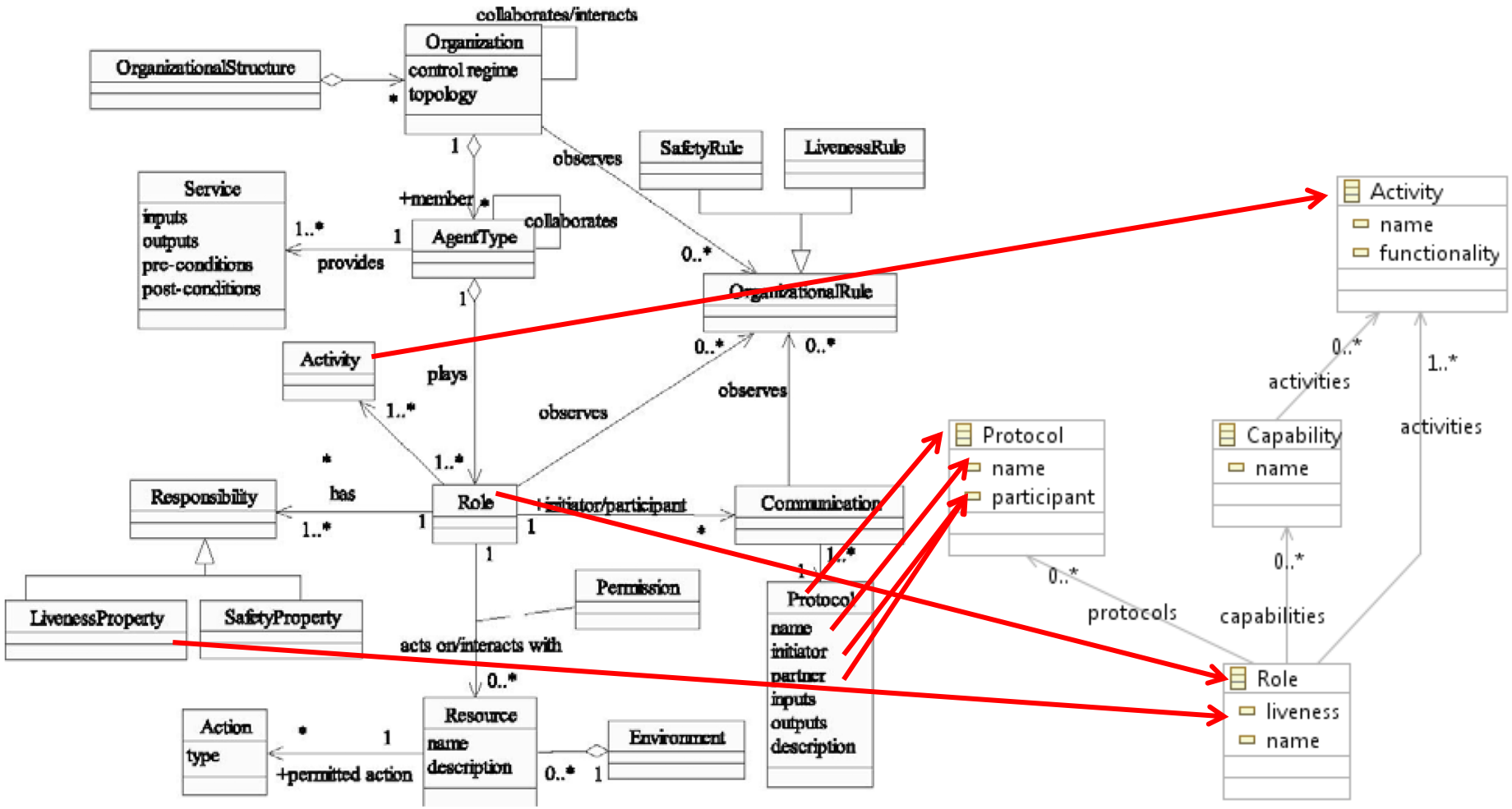
- E.g.
  - Use the **Gaia roles model** outputted by my Gaia analysis phase and then use the ASEME design phase (starting with **the previous fragment**) as I want to develop JADE agents

# Mapping Metamodels



- Gaia metamodel

ASEME SRM



# Transformation



- There are procedural or declarative languages to write transformation models Using the Eclipse Modeling Framework
  - Atlas Transformation Language (ATL)

```
rule GaiaRole2SRMRole{
 from gaiaRole : Gaia!Role
 to srmRole : SRM!Role
        (srmRole.name <- gaiaRole.name)
}
```

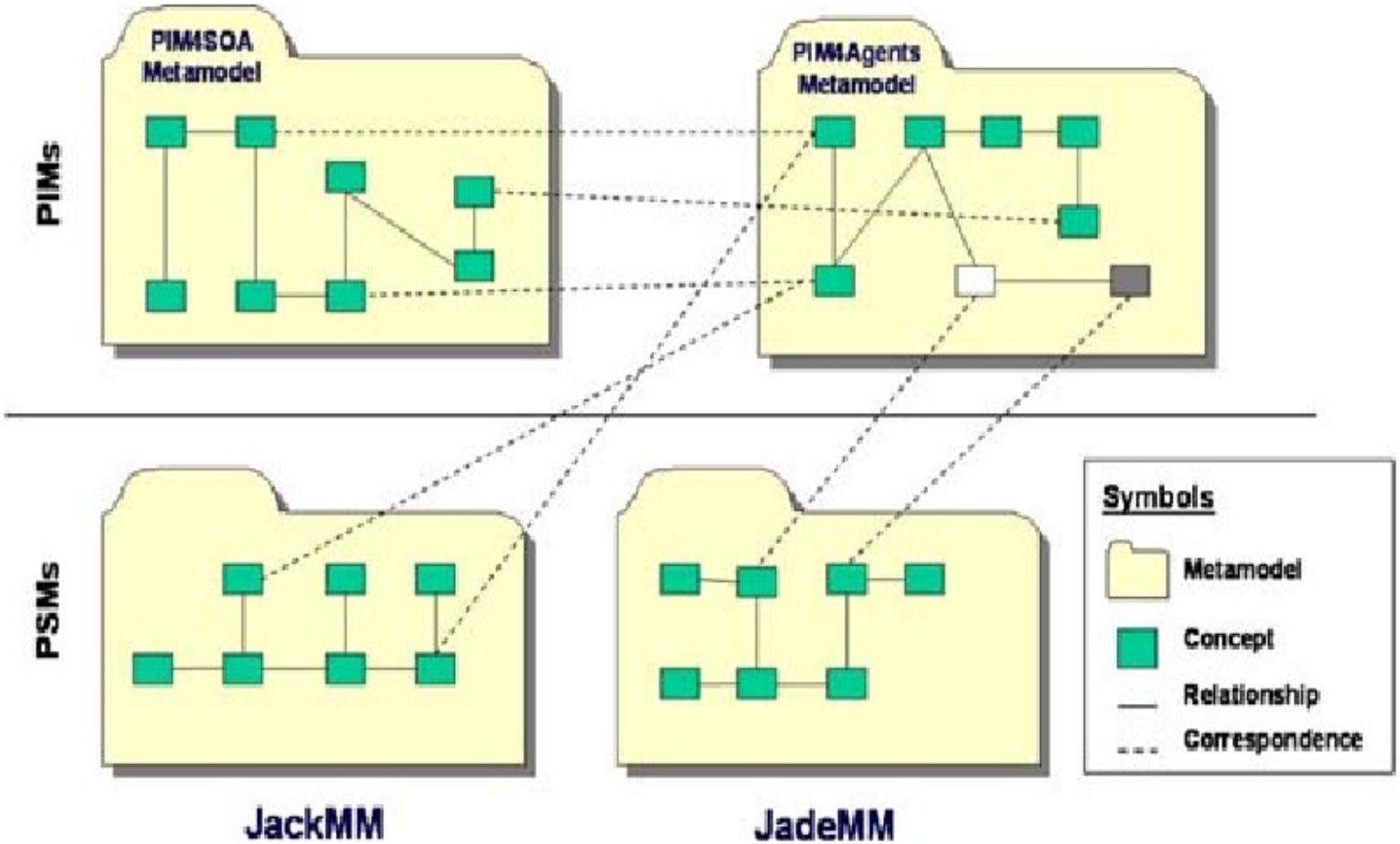
  - Java

```
SRM.Role newRole = new SRM.Role();
newRole.name = gaiaRole.name;
```

# The PIM4Agents metamodel



- Hahn, Madrigal-Mora and Fischer proposed in 2009 the PIM4Agents metamodel
- PIM4Agents abstracts from existing agent-oriented methodologies, programming languages, and platforms and could thus be considered as platform-independent
- A method engineer can transform his/her design model to PIM4Agents and then have an automatic implementation in JACK or JADE platforms






# The PIM4Agents concept

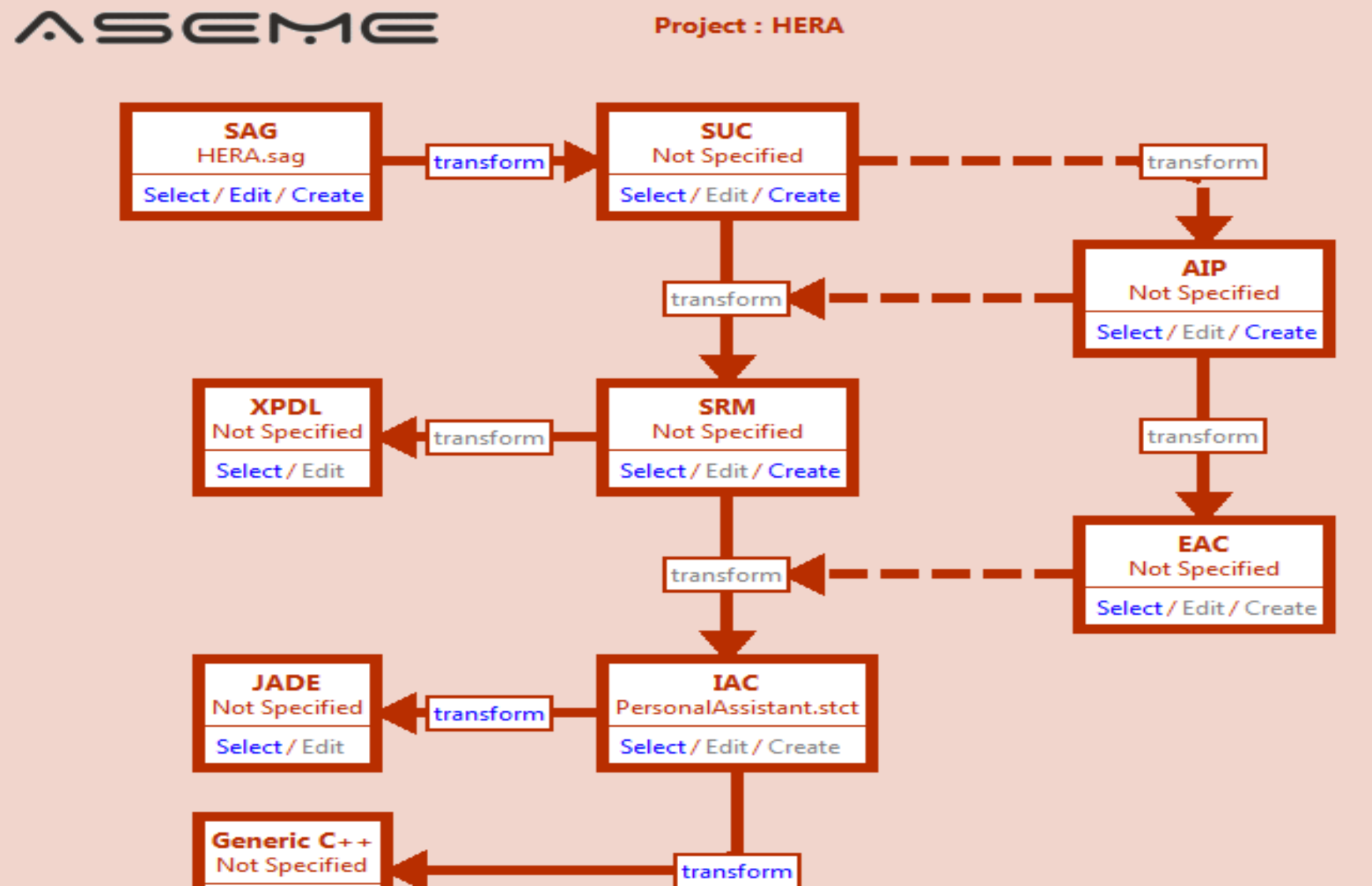
(Hahn, Madrigal-Mora and Fischer, 2009)

# IDEs

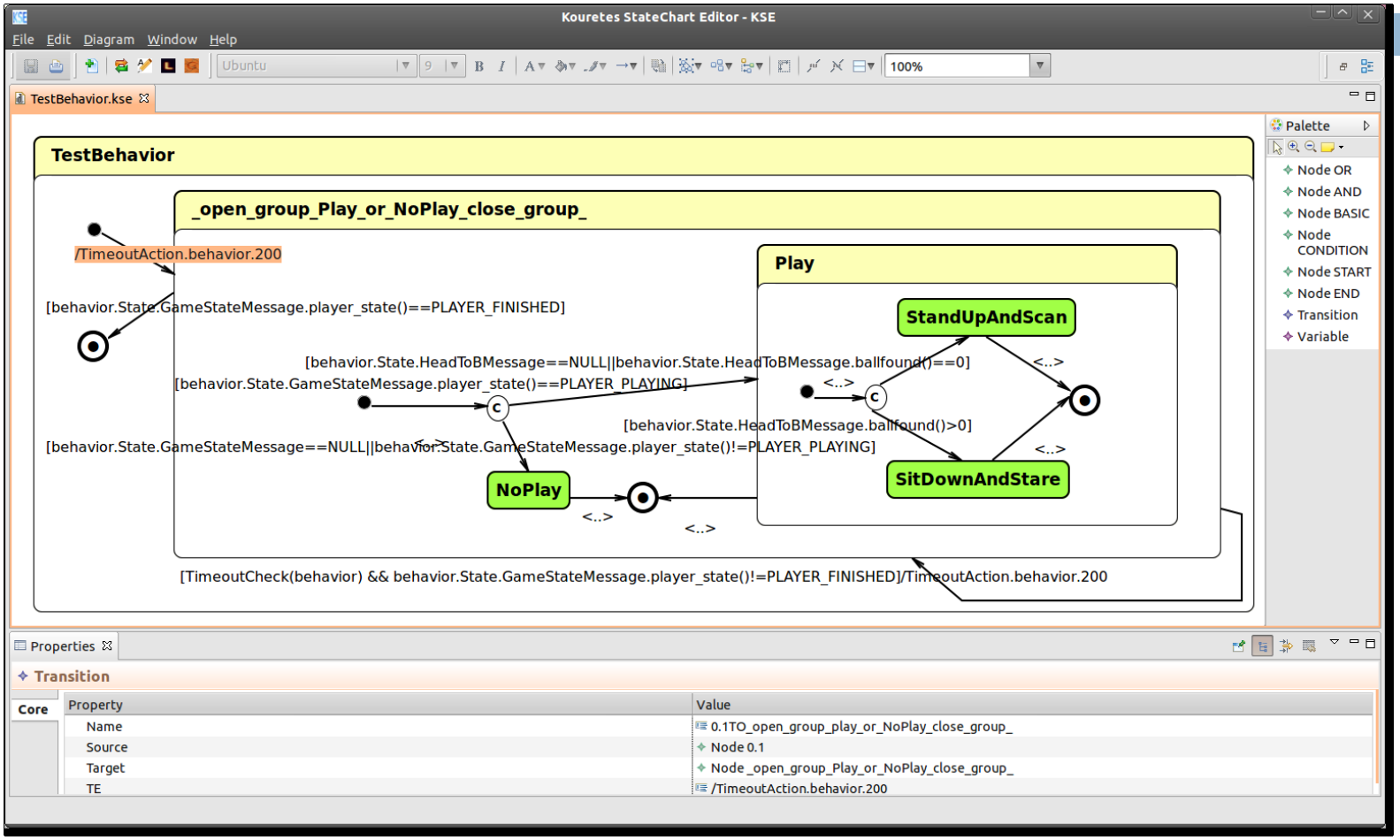
- ASEME IDE
- KSE

# The ASEME IDE Dashboard





# The Kouretes Statechart Editor (KSE)

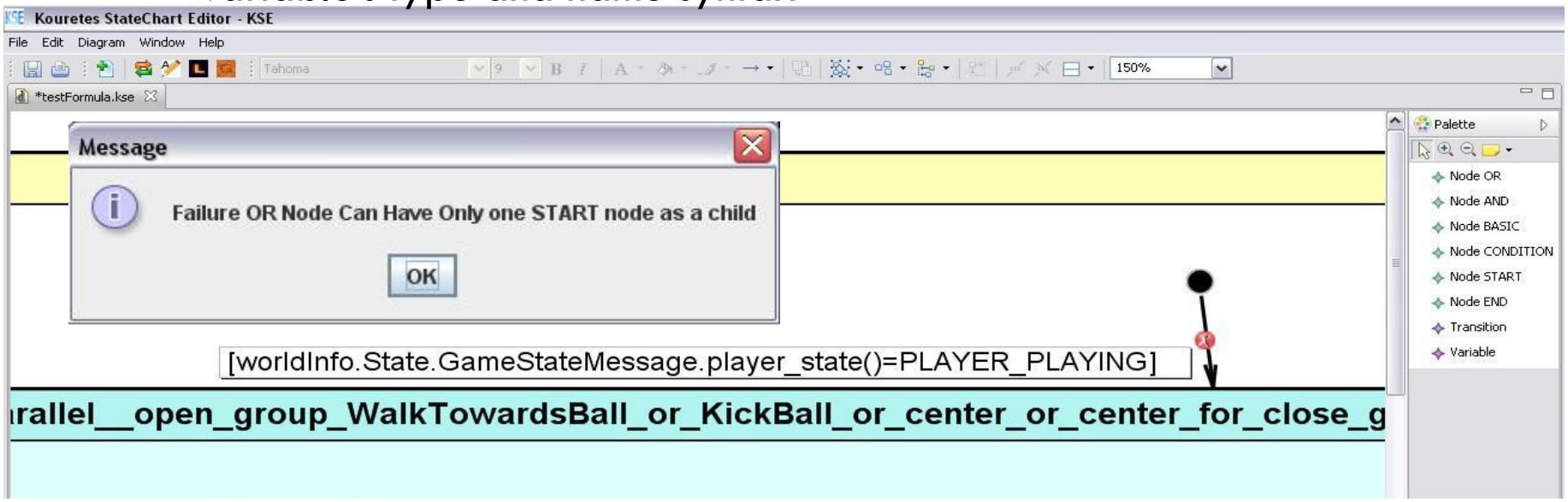



(Topalidou-Kyniazopoulou et al., 2012)

# IAC Model Validation Tool



- Live and on-demand model's validation
- Validation of the model according to
  - Harel's statechart rules
  - transition expression's syntax
  - variable's type and name syntax

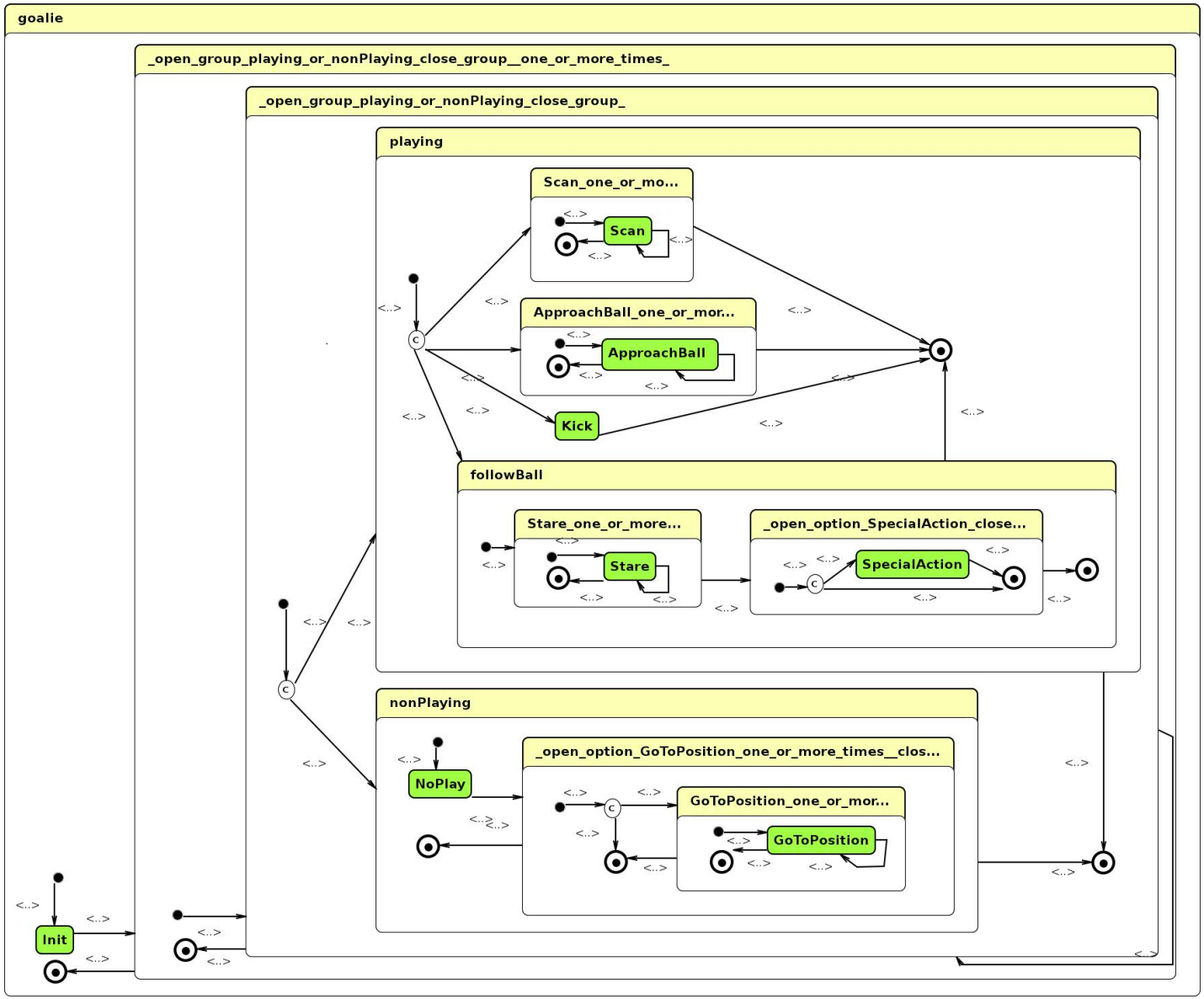


# A KSE Agent



- Robocup 2012 goalie

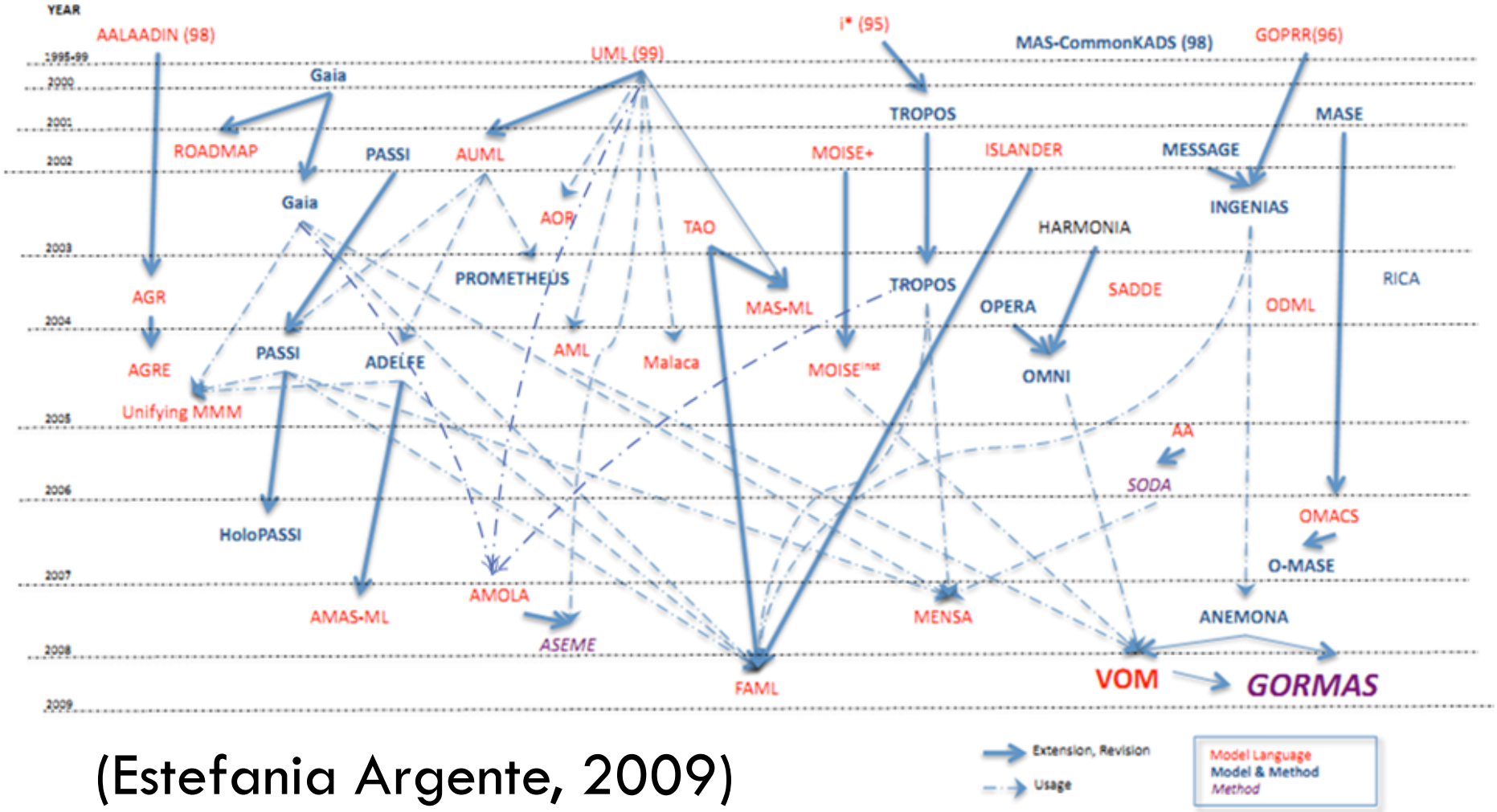
(Topalidou-Kyniazopoulou et al., 2012)



# RoboCup 2012 Goalie



# Evolution of MAS Modeling Languages and Methodologies




(Estefania Argente, 2009)

# Research Directions



- Automation in models refinement
  - using knowledge and repositories of models
- Self -Assessment and -Healing Agent Capability
  - by monitoring the intra-agent control model execution
- Adaptation automation
  - Using evolutionary techniques and fine tuning
- Prove properties of ASEME

# THANK YOU!

Questions????

Find out more about my work and CV at:
http://users.isc.tuc.gr/~nispanoudakis

Get in touch for ideas related to the above topics.

# References


Austin, John L.. **How To Do Things With Words.** Harvard University Press, 2nd edition, 1975

Baik, J., Boehm, B.: **Empirical analysis of case tool effects on software development effort**. **ACIS International Journal of Computer & Information Science** 1 (November 2000), pp. 1-10

Beck, K. 2000. **Extreme Programming Explained: Embrace Change**. Addison-Wesley Longman, Inc.

Bernon, C., Gleizes, M. P., Peyruqueou, S., & Picard, G. (2003). **Adelfe: A methodology for adaptive multi-agent systems engineering**. In Engineering Societies in the Agents World III (pp. 156-169). Springer Berlin Heidelberg.

Beydeda, S., Book, M., Gruhn, V. (2005), **Model-Driven Software Development. Springer**

Bittencourt, I. I., Bispo, P., Costa, E., Pedro, J., Véras, D., Dermeval, D., & Pacca, H. (2009). **Modeling jade agents from gaia methodology under the perspective of semantic web**. In Enterprise Information Systems (pp. 780-789). Springer Berlin Heidelberg.

Boehn B, **A Spiral Model of Software Development and Enhancement**, *Computer, IEEE*, 21(5):61-72, May 1988

Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J. and Perini, A. 2004. **TROPOS: An Agent-Oriented Software Development Methodology**. *J. Auton. Agent. Multi-Ag. 8, 3* (October 2004), 203-236

Burrafato, P., Cossentino, M.: **Designing a multi-agent solution for a bookstore with the PASSI methodology**. *Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002)*. May 2002, Toronto, Ontario, Canada at CAiSE'02

Chatzilaris E., Kyranou I., Ma J., Orfanoudakis E., Panakos A., Paraschos A., Pierris G., Spanoudakis N., Threlfall J., Topalidou A., Tzanetatou D., Vazaios E., Cameron S., Dahl, T., Lagoudakis M.G.: Kouretes 2011 SPL Team Description Paper. In: RoboCup 2011 Team Description Papers, Istanbul, 5-11 July, 2011

Cossentino, M., Gaglio, S., Garro, A. and Seidita, V. 2007. Method fragments for agent design methodologies: from standardisation to research. Int. J. of Agent-Oriented Software Engineering, 1, 1 (April 2007), 91-121

# References (cont.)


Davis N. and Mullaney J. The Team Software Process (TSP) in Practice: A Summary of Recent Results. TECHNICAL REPORT CMU/SEI-2003-TR-014 ESC-TR-2003-014, September 2003

Deloach, S.A.: Multiagent systems engineering of organization-based multiagent systems. In: Software Engineering for Multi-Agent Systems IV, LNCS 3914/2006, pages 109-125, Springer Berlin / Heidelberg, 2006

Deloach, S.A., Wood, M.F. and Sparkman, C.H. 2001. Multiagent Systems Engineering. Int. J. Softw. Eng. Know. 11, 3 (June 2001), 231-258

Fornara, N. and Colombetti, M. 2003. Defining interaction protocols using a commitment-based agent communication language. In Proceedings of the Second international Joint Conference on Autonomous Agents and Multiagent Systems (Melbourne, Australia, July 14 - 18, 2003). AAMAS '03. ACM, New York, NY, 520-527

Fowler, M. and J. Highsmith, "The Agile Manifesto," in Software Development, August 2001, pp. 28-32.

Hahn C., Madrigal-Mora C., Fischer K. (2009), **A platform-independent metamodel for multiagent systems**, *Autonomous Agents and Multi-Agent Systems*,18(2), pp 239–266

Harel, D. and Naamad, A. (1996), **The STATEMATE Semantics of Statecharts.** ***ACM T. Softw. Eng. Meth. 5, 4 (October 1996), 293-333.***

Hexmoor H. and Brainov S. Multi-Agent Systems. SAC 2002 Tutorial.

Hindriks K. Agent-Oriented Programming Tutorial. Leuven October 2011. http://people.cs.kuleuven.be/~mariohenrique.cruztorres/awpdf/KoenHindriks.pdf

IEEE Standard Glossary of Software Engineering Terminology. IEEE std 610.12-1990, 1990

Jouault F., Bezivin J., **KM3: A DSL for Metamodel Specification. In** ***Formal Methods for Open Object-Based Distributed Systems, Proceedings of the 8th IFIP WG 6.1 International Conference, (FMOODS 2006), Bologna, Italy, June 14-16, 2006. pp. 171-185.***

Jouault F., Kurtev, I. (2006), **Transforming models with ATL. Satellite Events at the MoDELS 2005 Conference, LNCS 3844, Springer-Verlag, 128–138**

# References (cont.)


Karamitrou M., Kofinas N., Pavlakis N., Topalidou-Kyniazopoulou A., Tzanetatou A.D., Spanoudakis N.I., Lagoudakis M.G.: Kouretes 2012 SPL Team Description Paper. In: RoboCup 2012 Team Description Papers, Mexico-City, 18-24 June, 2012

Kargas N., Kofinas N., Michelioudakis E., Pavlakis N., Piperakis S., Spanoudakis N.I., Lagoudakis M.G.: Kouretes 2013 SPL Team Description Paper. In: RoboCup 2013 Team Description Papers, Eindhoven, 24-30 June, 2013

Klatt, B. (2008), **Xpand: a closer look at the model2Text transformation language.** In: *Proceedings of the 12th European conference on software maintenance and reengineering (CSMR'08)*, Athens, Greece, http://www.bar54.de/benjamin.klatt-Xpand.pdf

Kleppe A., (2008), **Software Language Engineering: Creating Domain-Specific Languages Using Metamodels.** Addison-Wesley Professional

Langlois, B., Jitia, C.E., Jouenne, E.. DSL Classification. In 7th OOPSLA Workshop on Domain-Specific Modeling, 2007. http://www.dsmforum.org/events/DSM07/papers/langlois_jitia.pdf

Marcais J., Spanoudakis N., Moraitis P.: Using Argumentation for Ambient Assisted Living. In: Workshop on Artificial Intelligence Applications in Biomedicine (AIAB 2011) in conjunction with the 12th Engineering Applications of Neural Networks (EANN 2011) / 7th Artificial Intelligence Applications and Innovations (AIAI 2011) Joint Conferences, September 15-18, Corfu, Greece, 2011

Mazouzi, H., Seghrouchni, A. E., and Haddad, S. 2002. Open protocol design for complex interactions in multi-agent systems. In Proceedings of the First international Joint Conference on Autonomous Agents and Multiagent Systems: Part 2 (Bologna, Italy, July 15 - 19, 2002). AAMAS '02. ACM, New York, NY, 517-526

Moore, S.A. “On conversation policies and the need for exceptions”, Issues in Agent Communication, Lecture Notes in Artificial Intelligence 1916, Springer, 2000, pp. 144–159

Moore, Scott A.. A foundation for flexible automated electronic commerce. Working paper at the University of Michigan Business School no. 99-011, May 1999, http://deepblue.lib.umich.edu/bitstream/2027.42/35926/2/b2014129.0001.001.pdf

# References (cont.)


Odell J., Parunak H. V. D. and Bauer B. Extending UML for Agents. In Proc. of the Agent-Oriented Information Systems (AOIS) Workshop at the 17th National conference on Artificial Intelligence (AAAI), 2000

Odell, J., Parunak, H. V. D. and Bauer, B. Representing Agent Interaction Protocols in UML. In Agent-Oriented Software Engineering, LNCS 1957/2001. Springer, 201-218

OMG (2002), **Software Process Engineering Metamodel Specification. OMG, 2002, http://www.omg.org**

OMG (2001), **Model Driven Architecture (MDA), Document number ormsc/2001-07-01 , http://www.omg.org**

Padgham, L. and Winikoff, M. Prometheus: A Methodology for Developing Intelligent Agents. In G. Goos, J. Hartmanis, and J. van Leeuwen (eds) Agent-Oriented Software Engineering III, Lecture Notes in Computer Science, Volume 2585/2003, Springer Berlin/Heidelberg, pp. 174-185

Paraschos A., Spanoudakis N., Lagoudakis M.G.: **Model-Driven Behavior Specification for Robotic Teams. In the *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012), Valencia, Spain, June 4–8, 2012***

Paurobally, S., Cunningham, R. and Jennings, N.R. Developing agent interaction protocols using graphical and logical methodologies. In 2004 Programming Multi-Agent Systems, LNCS 3067/2004. Springer, 149-168

Pavón, J., Gómez-Sanz, J.J. & Fuentes, R.: The INGENIAS Methodology and Tools. In: Henderson-Sellers, B. and Giorgini, P., editors: Agent-Oriented Methodologies. Idea Group Publishing, pp. 236-276 (2005)

Ricordel, P. and Demazeau, Y. (2002). Volcano, a vowels-oriented multi-agent platform. In From Theory to Practice in Multi-Agent Systems, Second International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS 2001), volume 2296 of Lecture Notes in Computer Science, pages 253-262. Springer

Schwaber, K. & Beedle, M. 2002. Agile Software Development with Scrum. Prentice-Hall. Upper Saddle River, NJ. 0-13-067634-9 158

# References (cont.)


Smarsly, K. and Hartmann, D.: Agent-oriented development of hybrid wind turbine monitoring systems. In: W. Tizani (Editor), Proceedings of the International Conference of Computing in Civil and Building Engineering (ICCCBE 2010), 30 June-2 July, Nottingham, UK, Nottingham University Press, Paper 214, p. 427, ISBN 978-1-907284-60-1, 2010

Smarsly, K., & Law, K. H. (2012). Advanced structural health monitoring based on multi-agent technology. Computation for Humanity: Information Technology to Advance Society.

Spanoudakis N. (2009), **The Agent Systems Engineering Methodology (ASEME).** *PhD Thesis*, Paris Descartes University, Paris, France

Spanoudakis N., Moraitis P.: Using ASEME Methodology for Model-Driven Agent Systems Development. In D. Weyns and M.-P. Gleizes (Eds.): Agent Oriented Software Engineering XI, Lecture Notes in Computer Science (LNCS), vol. 6788, Springer-Verlag Berlin Heidelberg, 2011, pp. 106-127

Spanoudakis N., Moraitis P.: Modular JADE Agents Design and Implementation using ASEME. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'10), Toronto, Canada, 2010

Spanoudakis N., Moraitis, P.. Engineering an Agent-based System for Product Pricing Automation. International Journal of Engineering Intelligent Systems, CRL Publishing, Vol. 17, No. 2-3, June/September 2009

Sycara K., Zeng D. (1996) Coordination of Multiple Intelligent Software Agents. International Journal of Cooperative Information Systems 05 (2-3) p. 181-211

Tolvanen, J. P. 1998. Incremental Method Engineering with Modeling tools. Doctoral Thesis. University of Jyväskylä. 301 p.

Topalidou-Kyniazopoulou A., Spanoudakis N.I., Lagoudakis, M.G.: A CASE Tool for Robot Behavior Development. In the 16th RoboCup International Symposium (RCS 2012), Mexico City, June 24, 2012

Weiss, M. Patterns for Motivating an Agent-Based Approach. In G. Goos, J. Hartmanis, and J. van Leeuwen (eds) Conceptual Modeling for Novel Application Domains, Lecture Notes in Computer Science, Volume 2814/2003, Springer Berlin / Heidelberg, 2003

# References (cont.)


Weiss G. (2013), **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence**. 2nd edition. MIT Press

Winikoff, M. (2005). JACK™ intelligent agents: An industrial strength platform. In Multi-Agent Programming (pp. 175-193). Springer US.

Wirth, Niklaus (1996), **Extended Backus-Naur Form (EBNF), ISO/IEC 14977:1996(E)**

Wooldridge M. & Jennings N. (1995), **Intelligent Agents: Theory and Practice, The Knowledge Engineering Review 10(2), 115-152**

Wooldridge, M., Jennings, N.R., Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems Vol. 3. No. 3 (2000) 285-312

Zambonelli, F., Jennings, N.R. and Wooldridge, M. 2003. Developing multiagent systems: the Gaia Methodology. ACM T. Softw. Eng. Meth. 12, 3 (July 2003), 317-370.