

---

# Model-Driven Agents Development with ASEME

Nikolaos Spanoudakis      nikos@science.tuc.gr  
Technical University of Crete, Dept of Sciences

Pavlos Moraitis      pavlos@mi.parisdescartes.fr  
Paris Descartes University, Laboratoire d'Informatique Paris Descartes (LIPADE)

---

# Overview

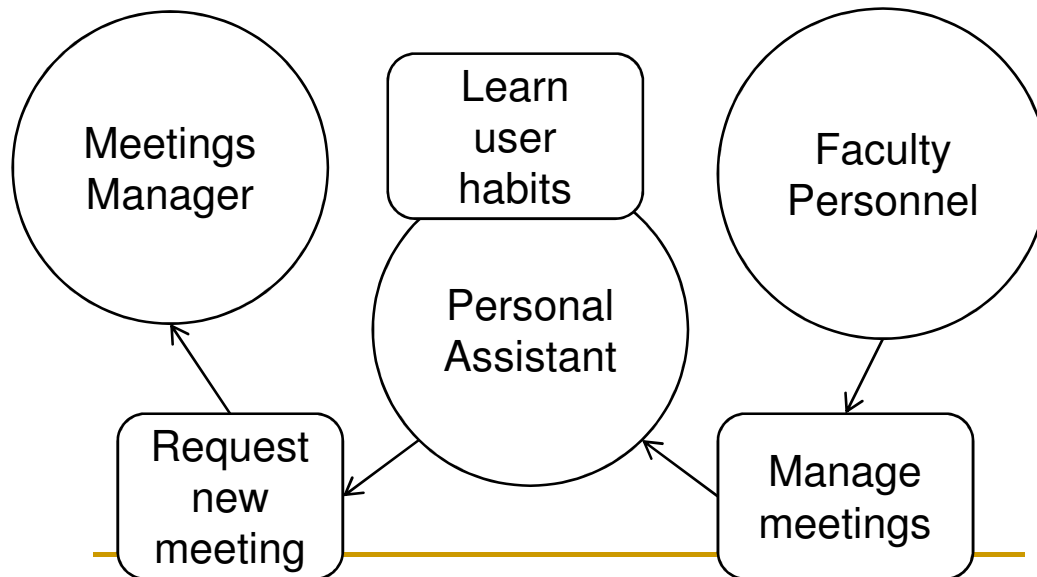
- Present the Agent MOdeling LAnguage (AMOLA) models
- Present the Model-Driven Agent Systems Engineering Methodology (ASEME)
- Conclude

# ASEME overview

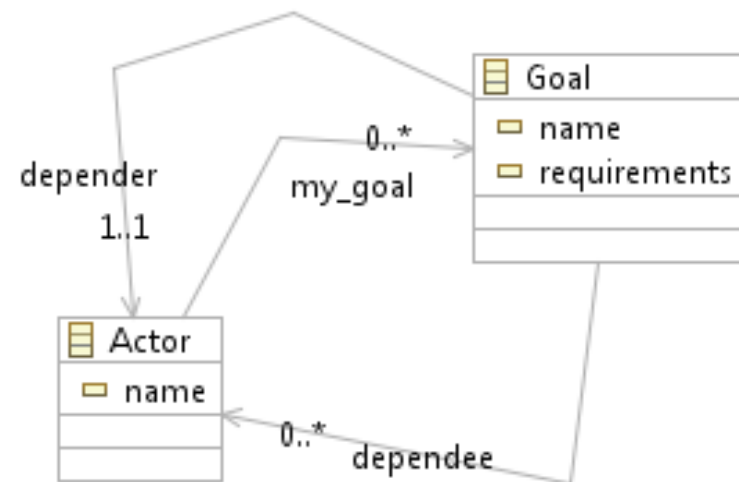
<i>Development Phase</i>	<i>Levels of Abstraction</i>		
	<i>Society Level</i>	<i>Agent Level</i>	<i>Capability Level</i>
<b>Requirements Analysis</b> <i>AMOLA Models</i>	<b>Actors</b> Actor Diagram	<b>Goals</b> Actor Diagram	<b>Requirements</b> Requirements per goal
<b>Analysis</b> <i>AMOLA Models</i>	<b>Roles and Protocols</b> Use case Diagram, Agent Interaction Protocols	<b>Capabilities</b> Use case Diagram, Roles Model	<b>Functionalities</b> Functionality Table
<b>Design</b> <i>AMOLA Models</i>	<b>Society Control</b> Inter-agent control model, Ontology, Message Types	<b>Agent Control</b> Intra-agent control model	<b>Components</b>
<b>Implementation</b>	<b>Platform management code</b>	<b>Agent code</b>	<b>Capabilities code</b>
<b>Verification</b>	<b>Protocols testing</b>	<b>Agent testing</b>	<b>Component testing</b>
<b>Optimization</b>	<b>Number of instantiated agents</b>	<b>Agent resources</b>	<b>Code optimization</b>

# AMOLA Model: Requirements Analysis

- The **System Actors-Goals model (SAG)** inspired by the Tropos actor diagram

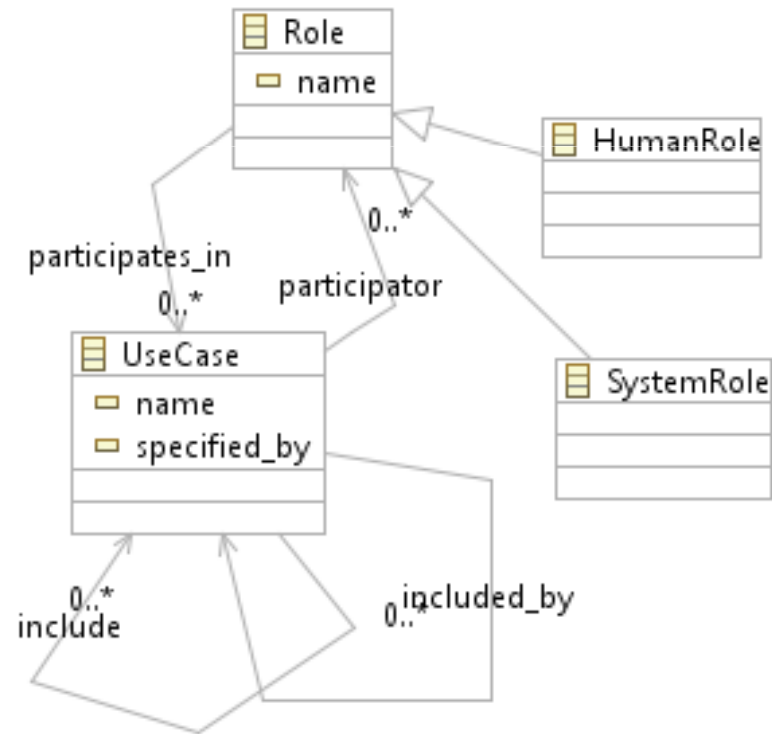
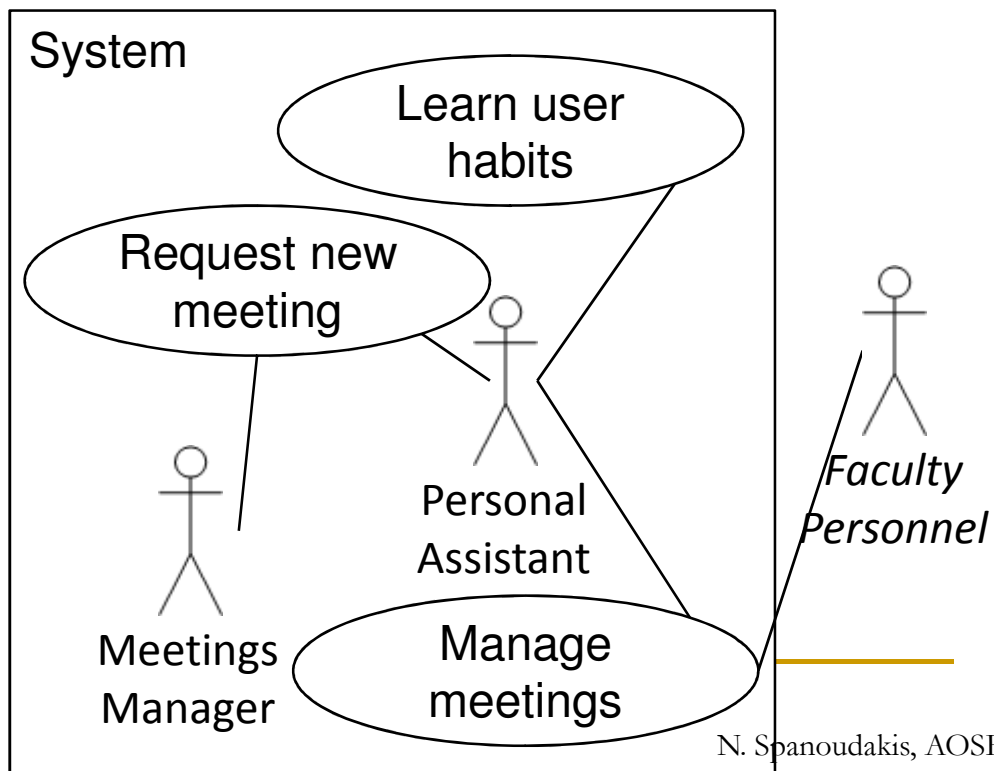


- And its metamodel definition using Ecore, the Eclipse Modeling Framework 's (EMF) model of a model



# AMOLA Model: Analysis

- The **System Use Cases model (SAG)** inspired by UML use cases
- And its metamodel definition



# AMOLA Model: Analysis (cont)

- The **System Roles model (SRM)** inspired by the Gaia roles model

**Role:** Personal Assistant

**Capabilities and Protocols:**

learn user habits, request new meeting:  
personal assistant, ...

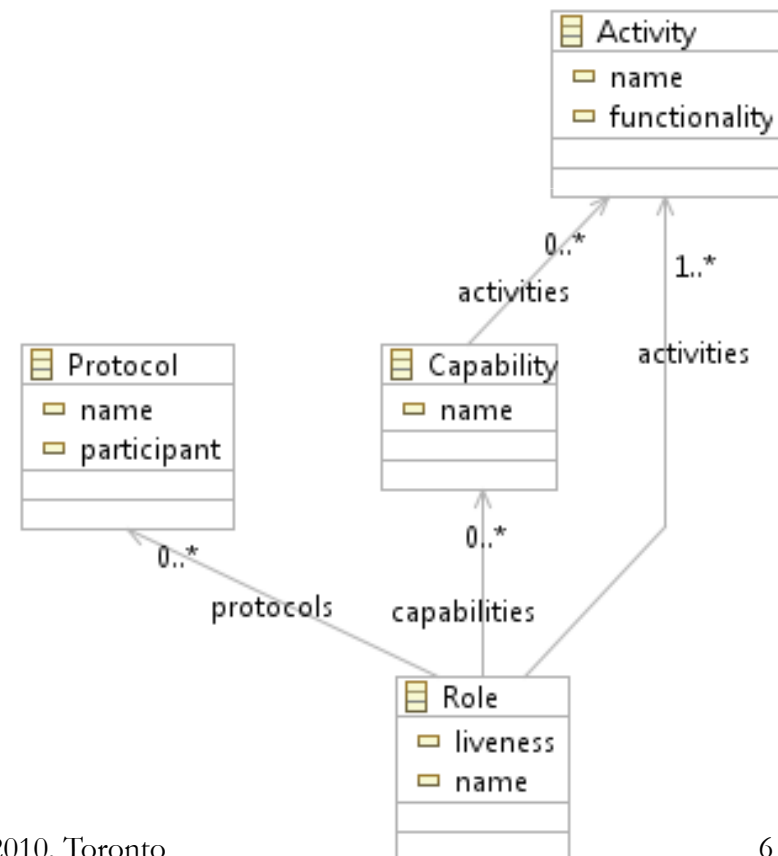
**Activities:**

learn user preference, update user preferences, ...

**Liveness:**

personal assistant = request new meeting ||  
learn user habits  
learn user habits = learn user preference.  
update user preferences

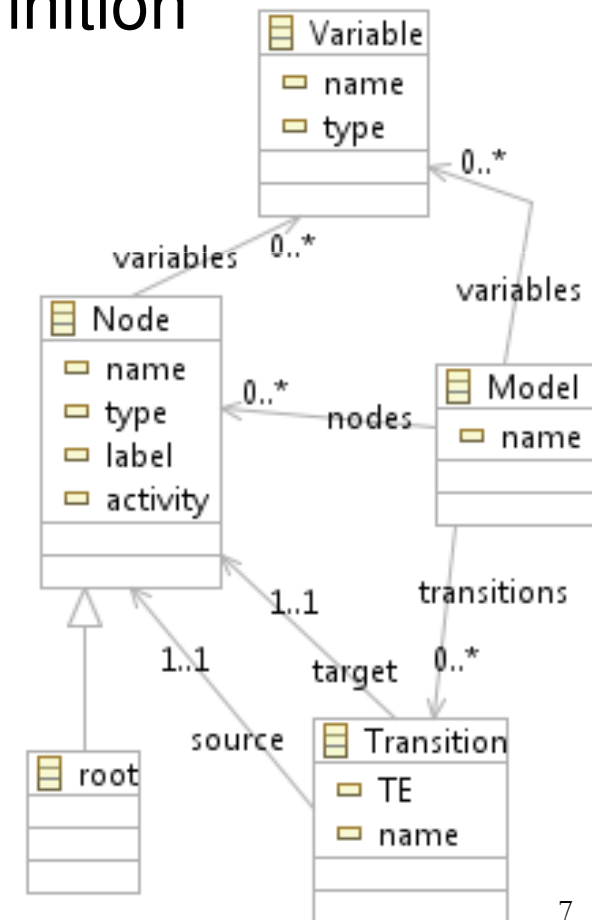
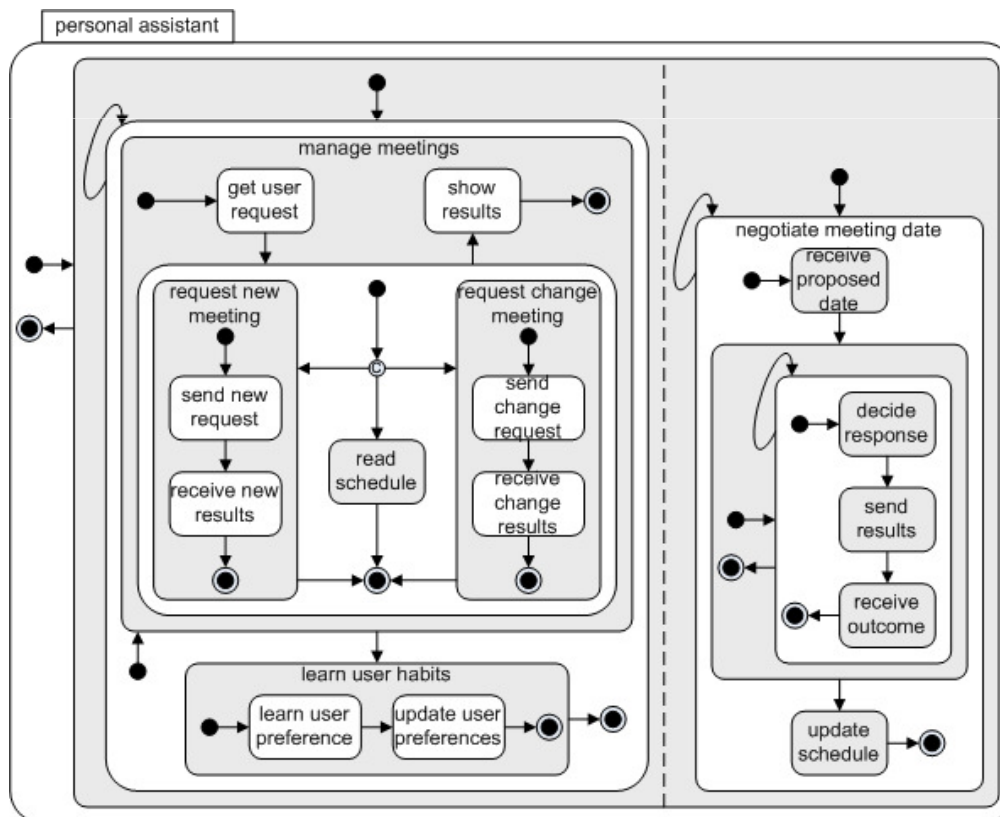
- And its metamodel definition



# AMOLA Model: Design

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

- And its metamodel definition

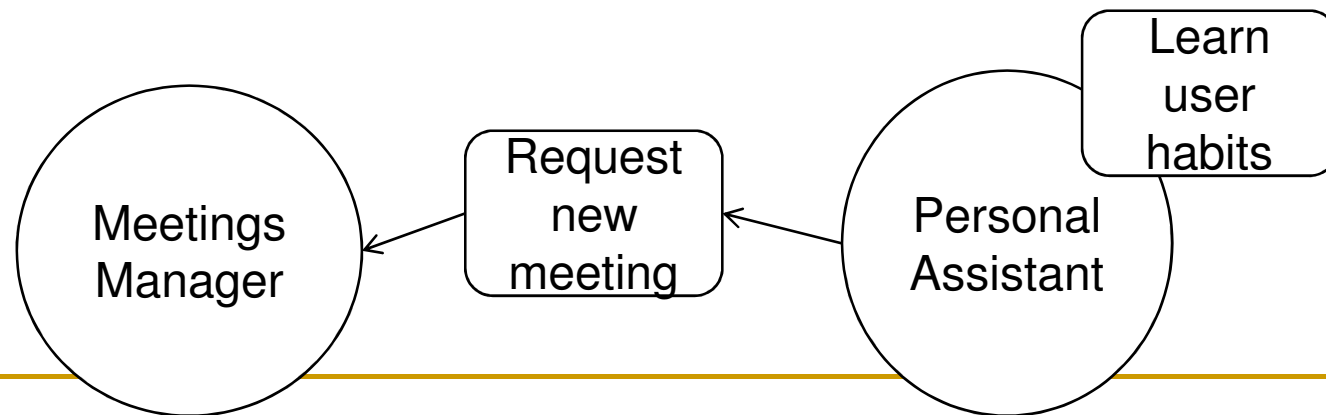


---

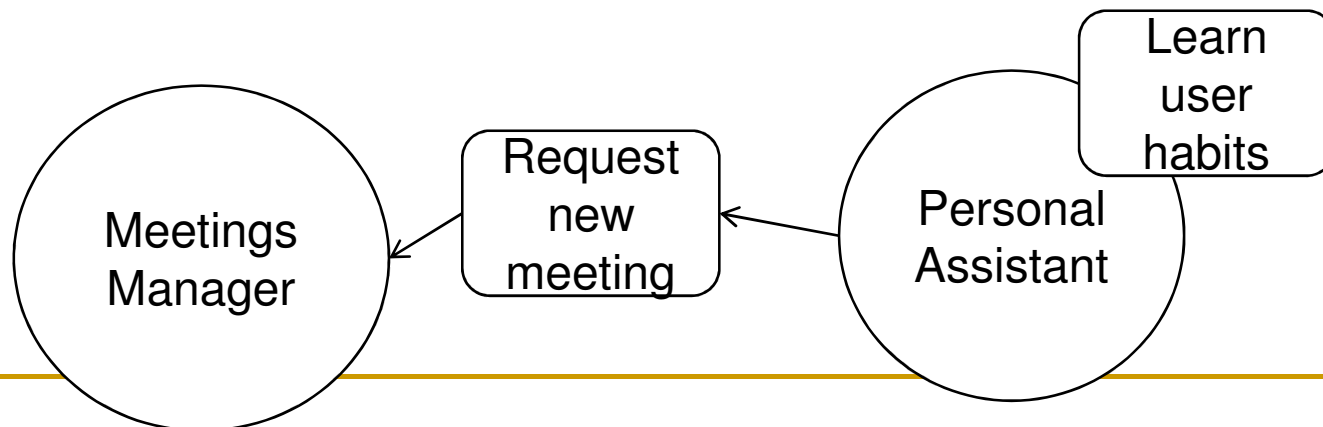
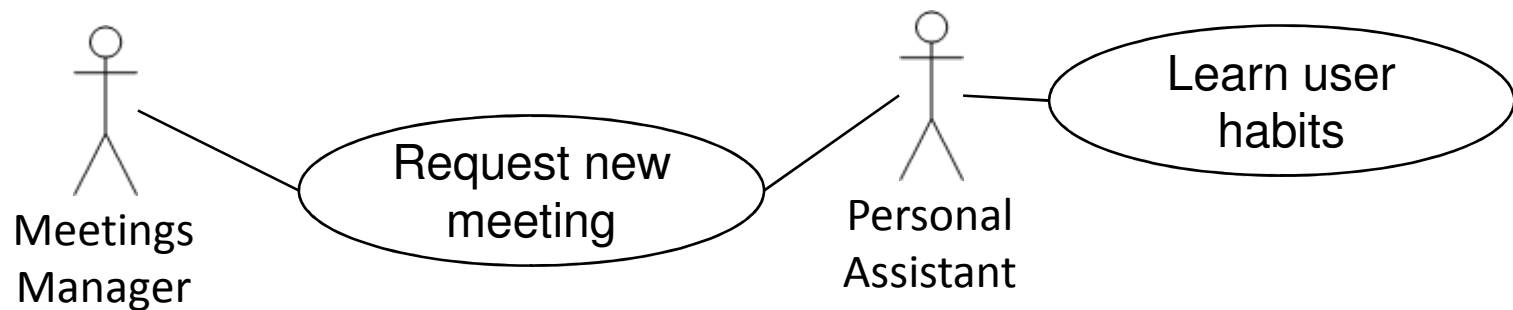
# ASEME Process

- 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*

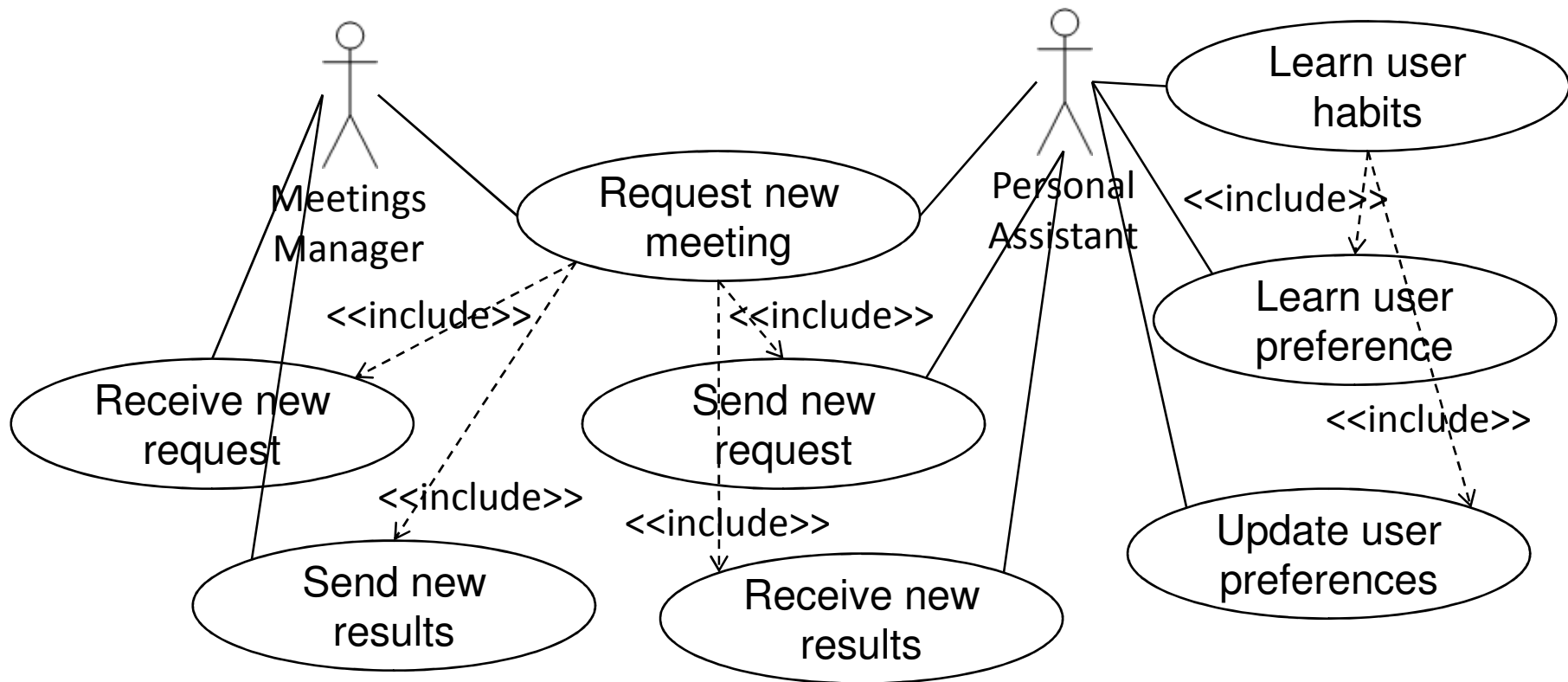
# Create and Edit the SAG Model



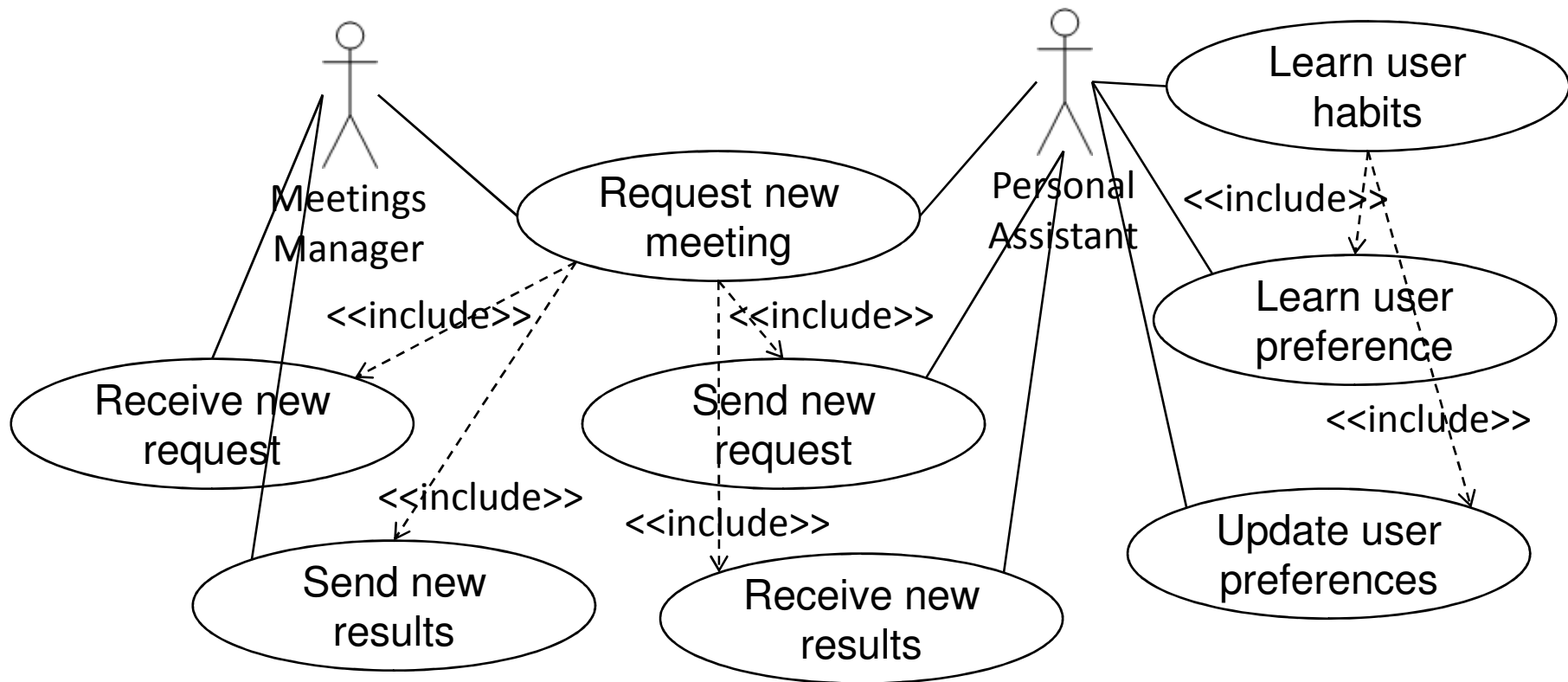
# The SAG 2 SUC Transformation



# Refine SUC Model



# SUC 2 SRM Transformation



# SUC 2 SRM Transformation

**Role:** Personal Assistant

**Capabilities and Protocols:**

Learn user habits

Request new meeting

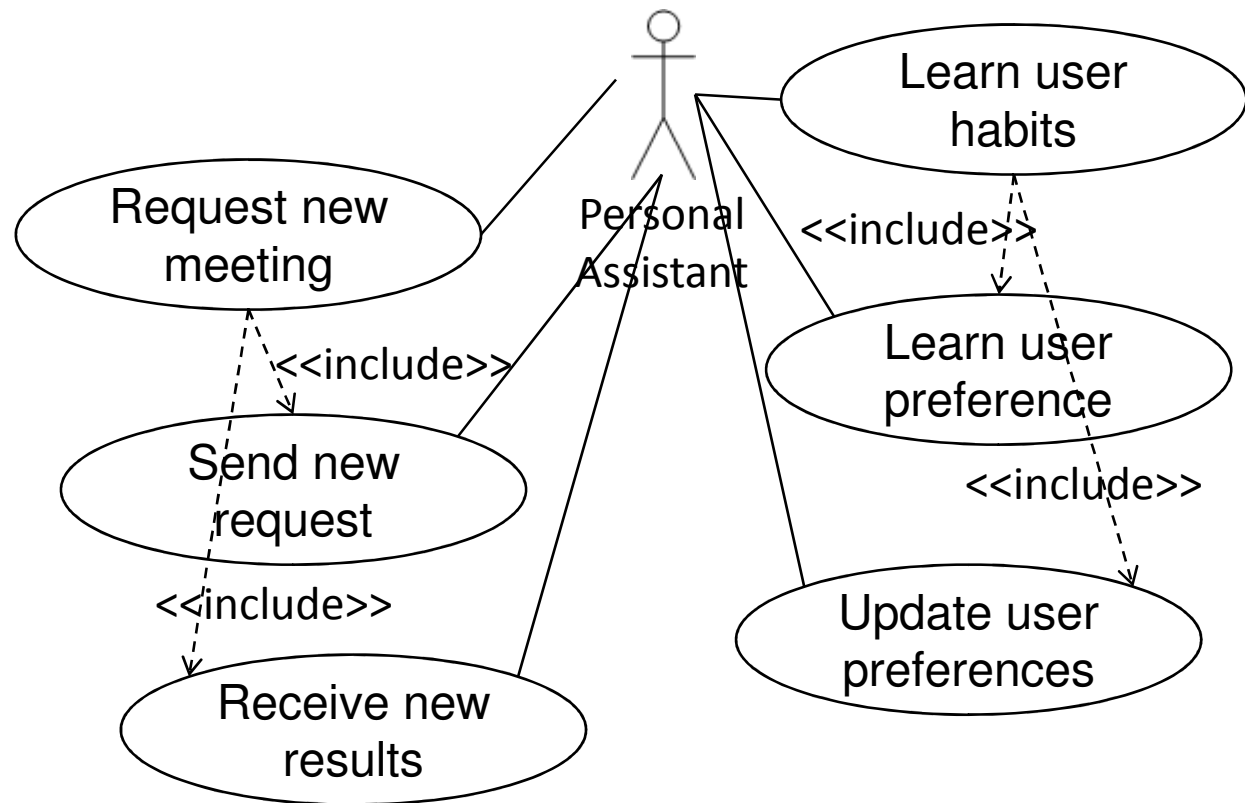
**Activities:**

Learn user preference

Update user preferences

Send new request

Receive new results



---

# 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:**

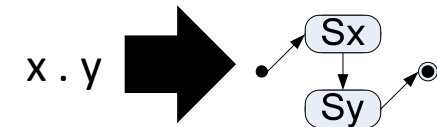
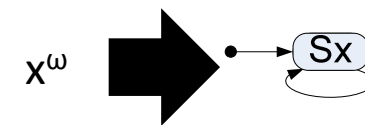
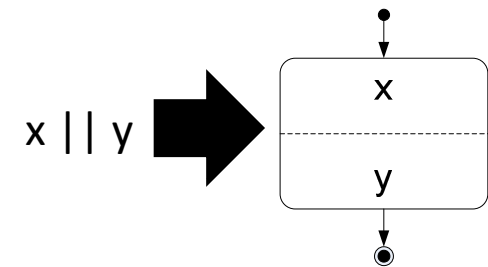
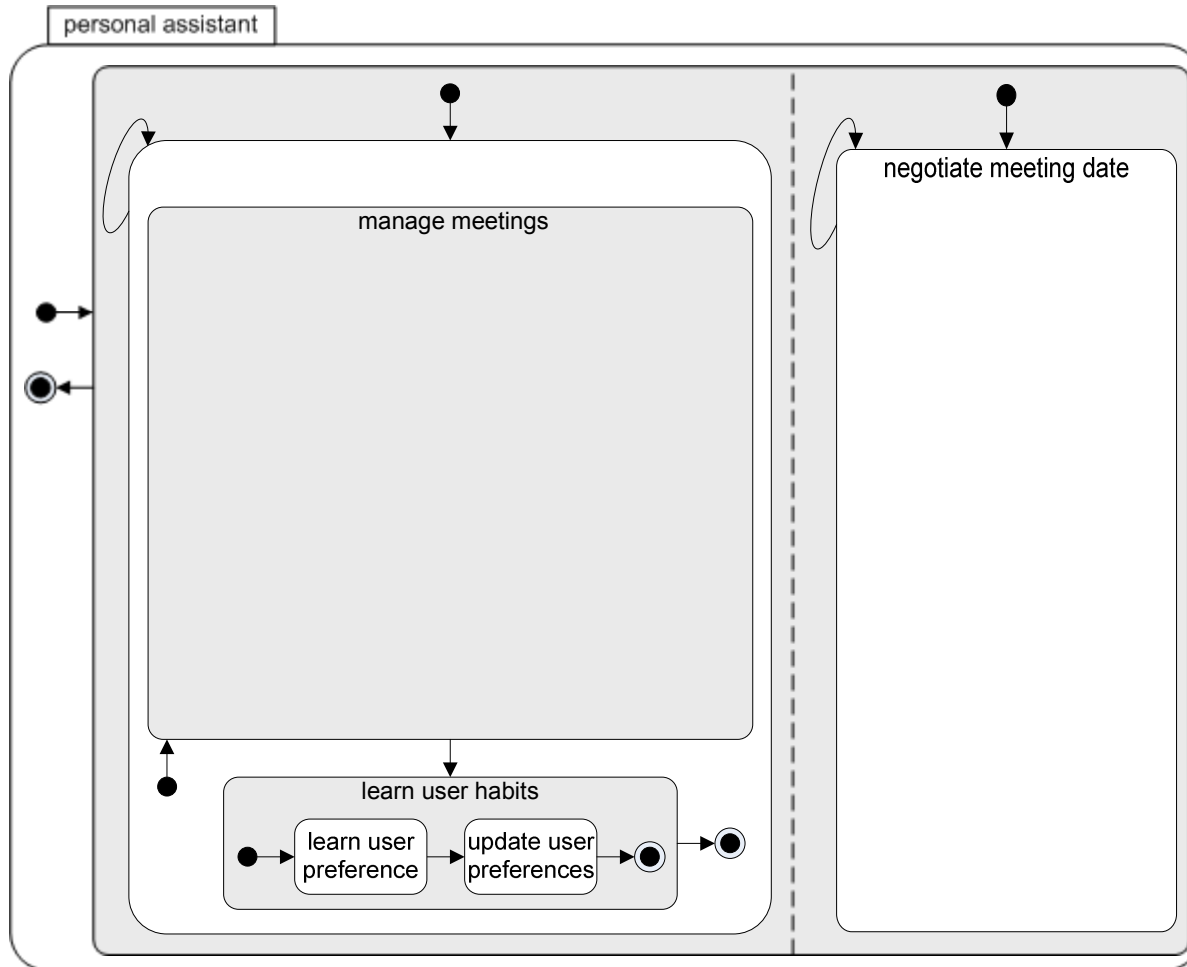
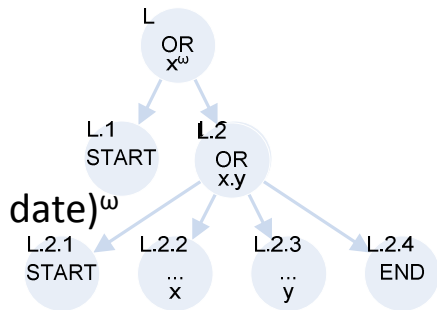
personal assistant = **(manage meetings. learn user habits)<sup>ω</sup> || (negotiate meeting date)<sup>ω</sup>**  
manage meetings = **get user request. (read schedule | request change meeting | request new meeting). show results**  
learn user habits = **learn user preference. update user preferences**  
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***

# SRM 2 IAC Transformation

## Liveness:

personal assistant = (manage meetings. learn user habits)<sup>ω</sup> || (negotiate meeting date)<sup>ω</sup>

learn user habits = learn user preference. update user preferences



---

# IAC 2 JADE transformation

For each node in S

  If node is root then create Agent class

  Else if  $\lambda(\text{node}) = \text{"BASIC"}$  then create a SimpleBehaviour

  Else if  $\lambda(\text{node}) = \text{"AND"}$  then create a ParallelBehaviour (|| gaia op.)

  Else if  $\text{sons}(\text{node}).\text{size}() = 2$  and there exists transitionExpression  $x \mid (\text{node}.2, x, \text{node}.2)$  belongs to  $\delta$  then create a CyclicBehaviour ( $\omega$  gaia op.)

  Else if  $\text{sons}(\text{node}).\text{size}() = 3$  and there exists transitionExpression  $x \mid (\text{node}.2, x, \text{node}.2)$  belongs to  $\delta$  then create a SimpleBehaviour (+ gaia op.)

  Else if there exists  $x$  belongs to  $\text{sons}(\text{node}) \mid \lambda(x) = \text{CONDITION}$  then

    If  $\text{sons}(\text{node}).\text{size}() = 4$  then create a SimpleBehaviour (\* gaia op.)

    Else create a SequentialBehaviour (| gaia op.)

  End if

  Else create a SequentialBehaviour (. gaia op.)

  End if

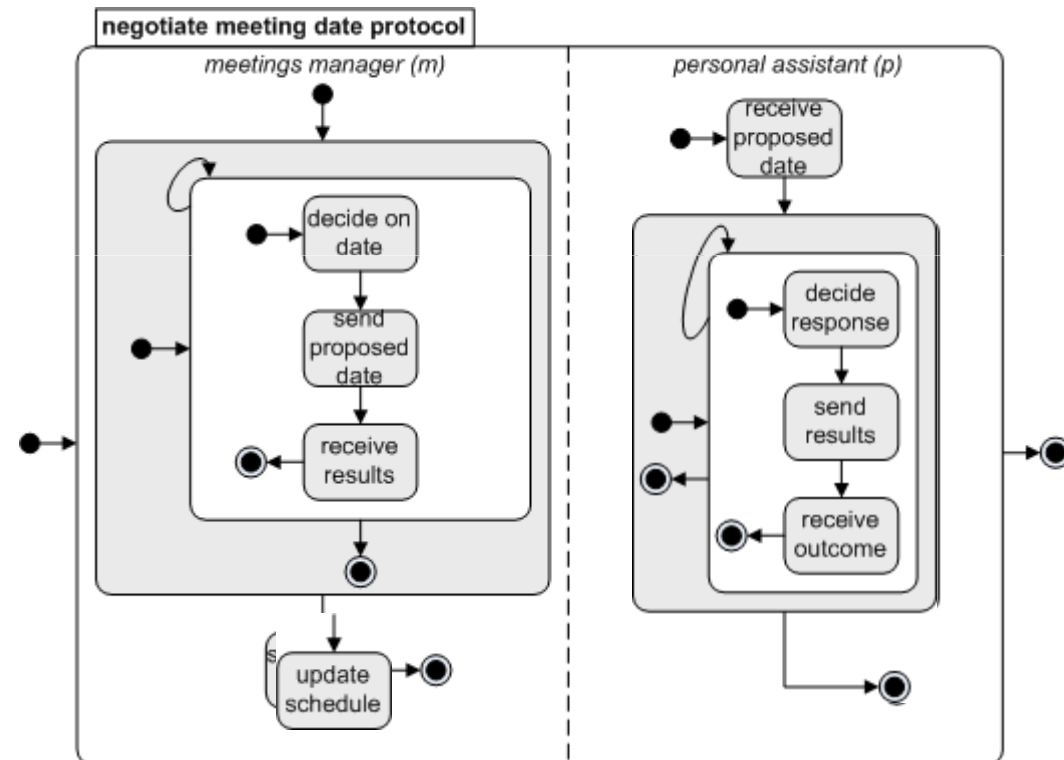
End for

---

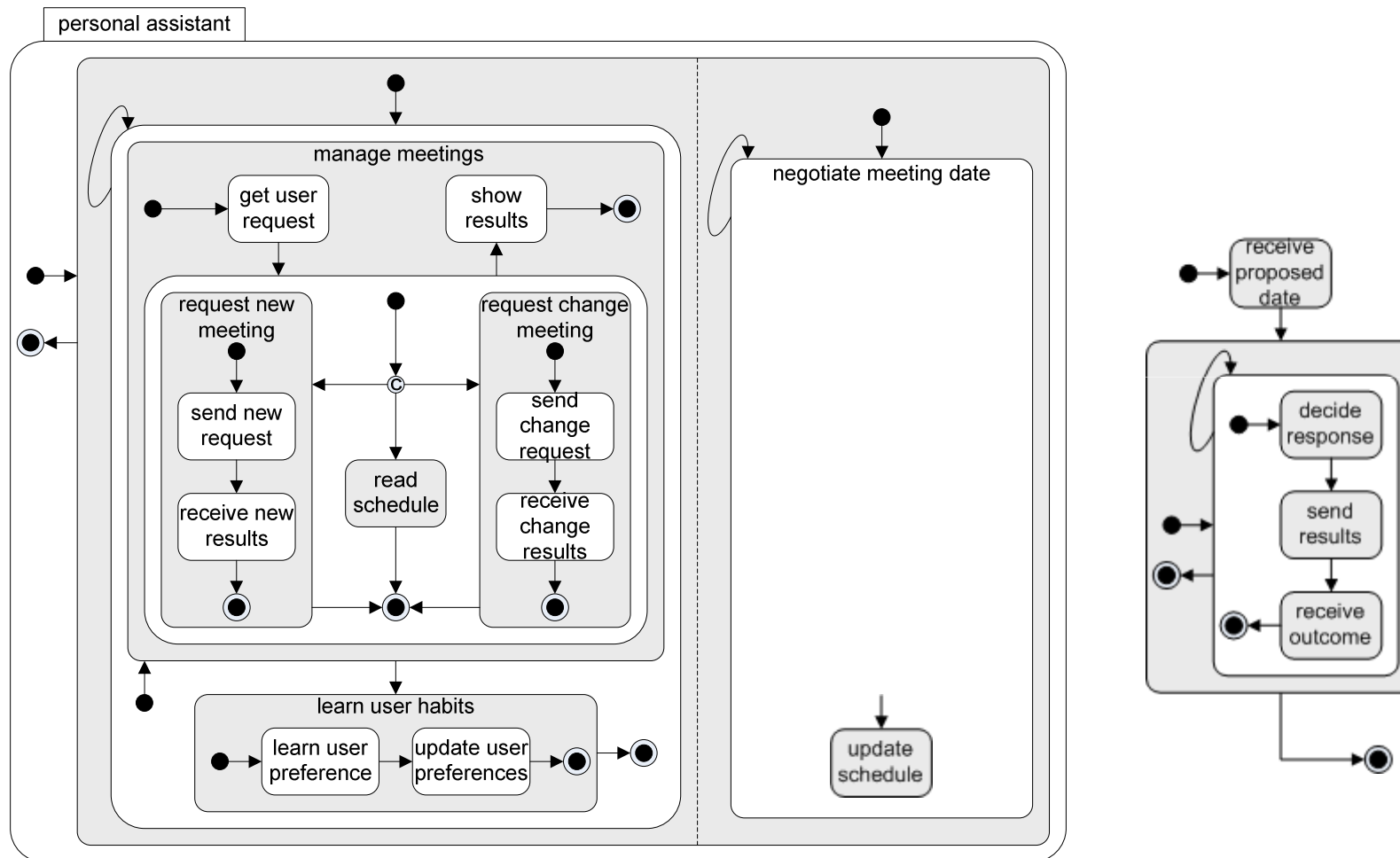
# The Inter- and Intra-Agent Control

- The inter-agent control (EAC) is a statechart defining the parallel behaviour of two or more agents
  - 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
    - By adding states with expressionless transitions after the START state (initialize) or before the END state
  - IAC allows the parallel execution of multiple protocols
-

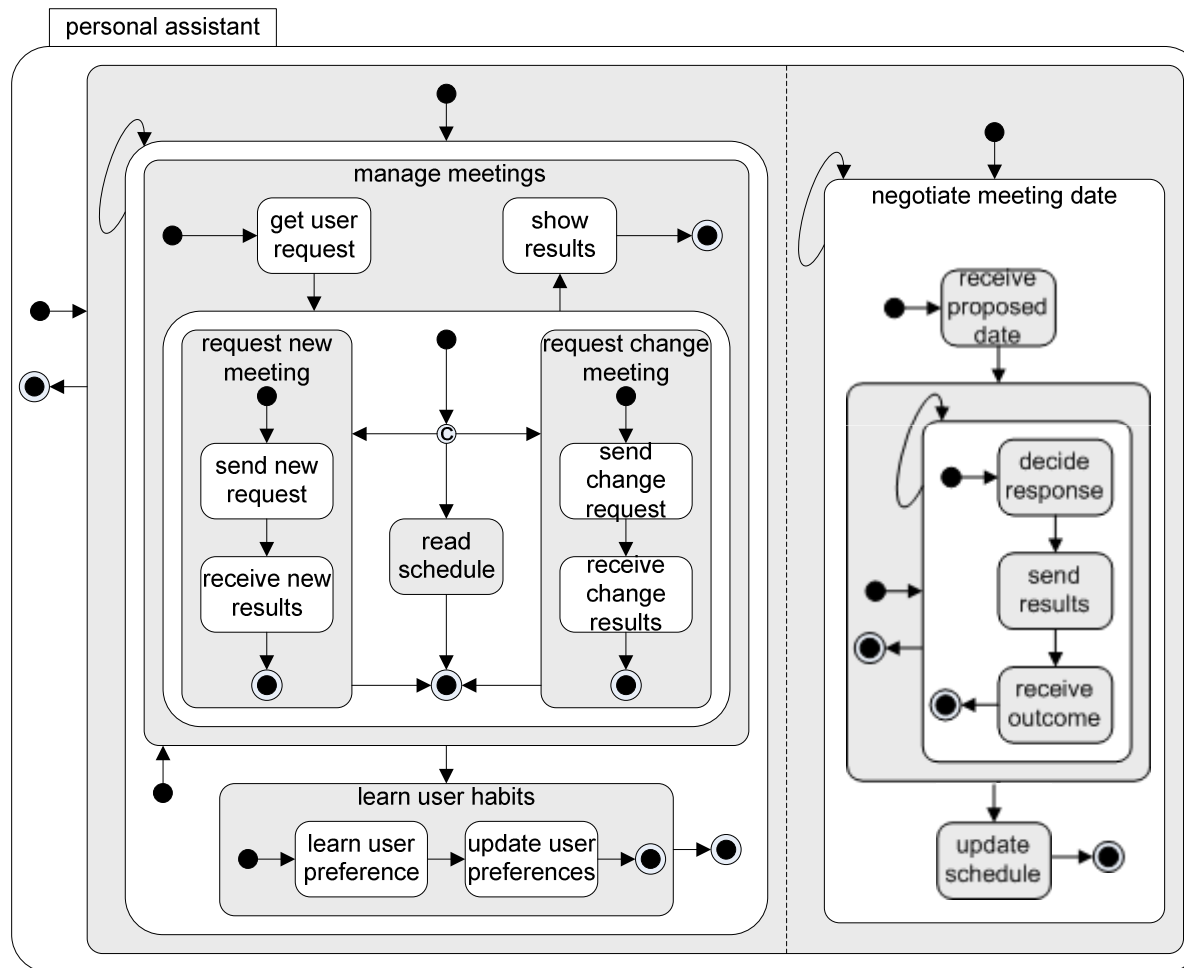
# Integrating an EAC model in an IAC model



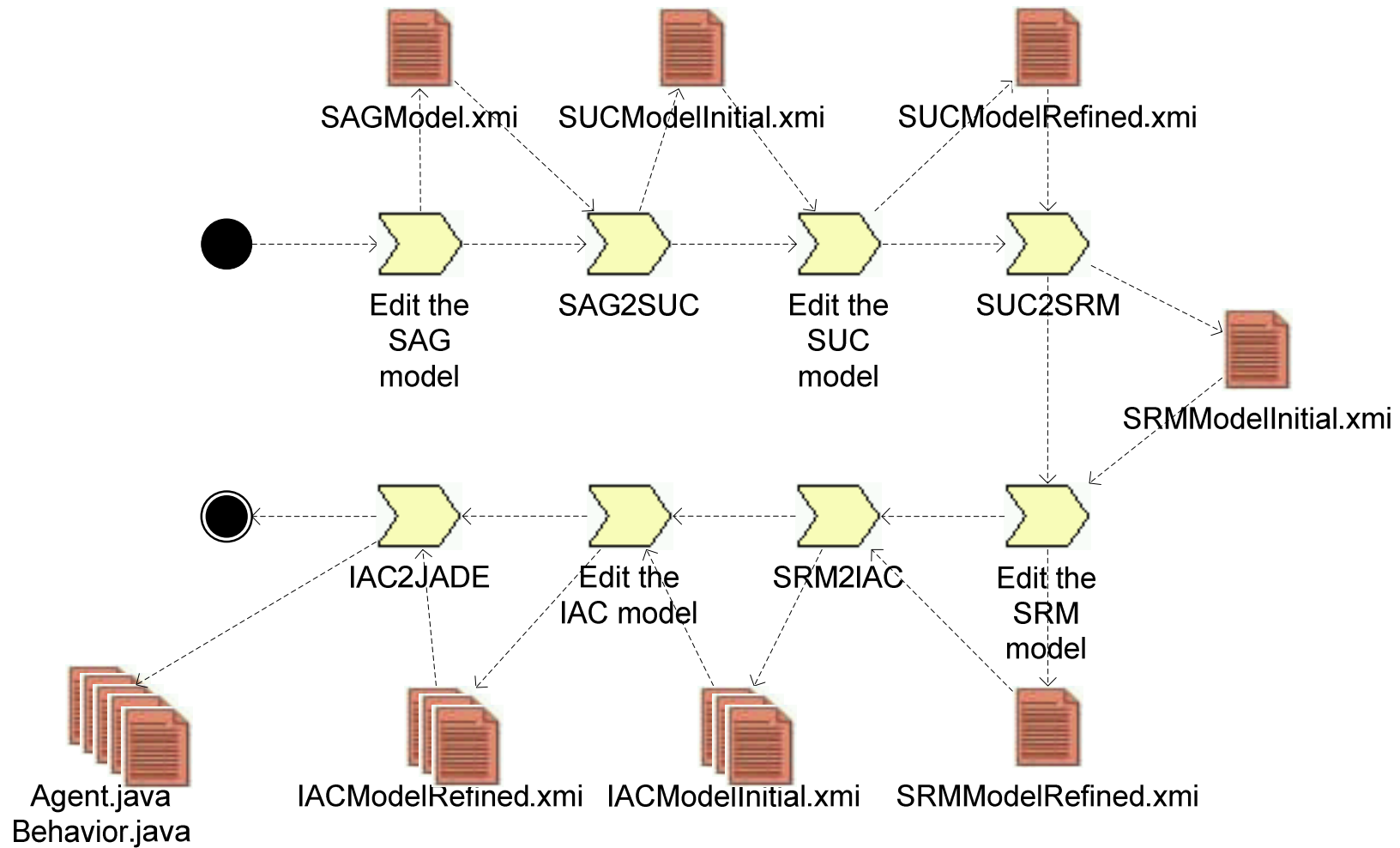
# Integrating an EAC model in an IAC model



# Integrating an EAC model in an IAC model



# ASEME MDE Process



# Conclusion

- ASEME has been used validated empirically through three case studies, two from developed real world systems (ASK-IT and Market-miner projects) and the third a complete example for a meetings management system.

Methodology	ASEME	Gaia	Tropos	INGENIAS	PASSI	Prometheus	MaSE
Abstraction	all phases	n/a	phase-based	n/a	phase-based	phase-based	n/a
MDE phases	all	n/a	some	defined by the modeler	some	n/a	some
Phases coverage	all	some	all	some	some	some	some
Agent nature	heterogeneous	heterogeneous	BDI-like agents	agents with goals and states	heterogeneous	BDI-like agents	not specified
Intra-agent control (IAC)	yes	no	no	no	no	no	yes
Uniform representation of IAC and inter-agent protocols	yes	no	no	no	no	no	no

---

# Employed technologies

- JADE (target agent platform)
- Rhapsody (target java platform)
- Eclipse (software development)
- EMF – ecore (metamodel definition)
- OMG MOF (XML representation of models)
- OMG HUTN (free text model representation)
- ATL (M2M transformation)
- Epsilon (T2M transformation)
- Xpand (M2T transformation)

---

# We are currently working on...

- Release the graphical editing models based on Eclipse GMF technology
- Develop automatic code generation toolkit for the NAO robot for the robocup 2010 in Singapore including a message exchanging scheme based on a blackboard
- Create templates for the analysis, design and implementation phases for use of specific technologies (e.g. OSGi services, web services, HMI, etc)
- Automate the Process Model generation using the Business Process Modeling Notation (BPMN) metamodel
- Perform a tools scalability analysis
- Try to systematically gather and analyze empirical results aiming to show the added value of ASEME

---

# Thank you!

Follow us on:

<http://www.amcl.tuc.gr/aseame>