

Designing BDI Systems: The Prometheus Methodology

AOS Agent Course 2017

Sebastian Sardina
sebastian.sardina@rmit.edu.au



October 26 - 27, 2016
Melbourne, Australia

<https://sites.google.com/site/ssardina/teaching/aosagt17>

Outline

- 1 Prometheus: Design & Analysis
- 2 System Specification
- 3 Architectural Design
- 4 Detailed Design
- 5 Protocols
- 6 Interaction Diagrams

Building Agents: BDI Agent-oriented Programming

A new **programming model** to simplify the construction of todays **large complex** systems situated in dynamic environments:

Building Agents: BDI Agent-oriented Programming

A new **programming model** to simplify the construction of today's **large complex** systems situated in dynamic environments:

- 1 View a system as composed of **autonomous** interacting entities: **agents**.

Building Agents: BDI Agent-oriented Programming

A new **programming model** to simplify the construction of today's **large complex** systems situated in dynamic environments:

- 1 View a system as composed of **autonomous** interacting entities: **agents**.
- 2 Internal state and decision process of agents is modelled in an intuitive manner following the notions of **mental attitudes** and **rationality**.

Building Agents: BDI Agent-oriented Programming

A new **programming model** to simplify the construction of today's **large complex** systems situated in dynamic environments:

- 1 View a system as composed of **autonomous** interacting entities: **agents**.
- 2 Internal state and decision process of agents is modelled in an intuitive manner following the notions of **mental attitudes** and **rationality**.
- 3 **Goal orientation**: instead of directly requesting the agents to perform certain actions, the developer can define more **abstract goals** for the agents.
 - provides a certain degree of flexibility on how to achieve the goals.

Building Agents: BDI Agent-oriented Programming

A new **programming model** to simplify the construction of today's **large complex** systems situated in dynamic environments:

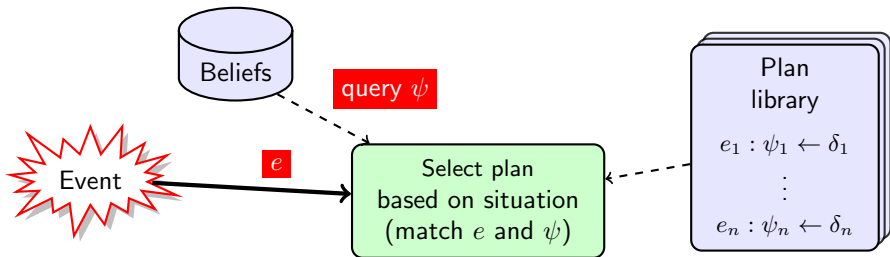
- 1 View a system as composed of **autonomous** interacting entities: **agents**.
- 2 Internal state and decision process of agents is modelled in an intuitive manner following the notions of **mental attitudes** and **rationality**.
- 3 **Goal orientation**: instead of directly requesting the agents to perform certain actions, the developer can define more **abstract goals** for the agents.
 - provides a certain degree of flexibility on how to achieve the goals.

Can be seen as a “successor” of object-oriented programming

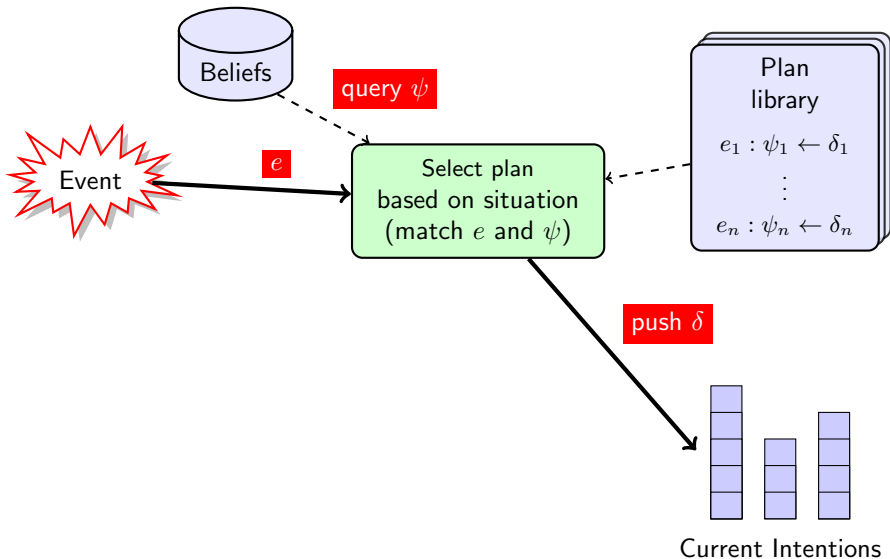
The BDI Execution Cycle [Rao&Georgeff 92]



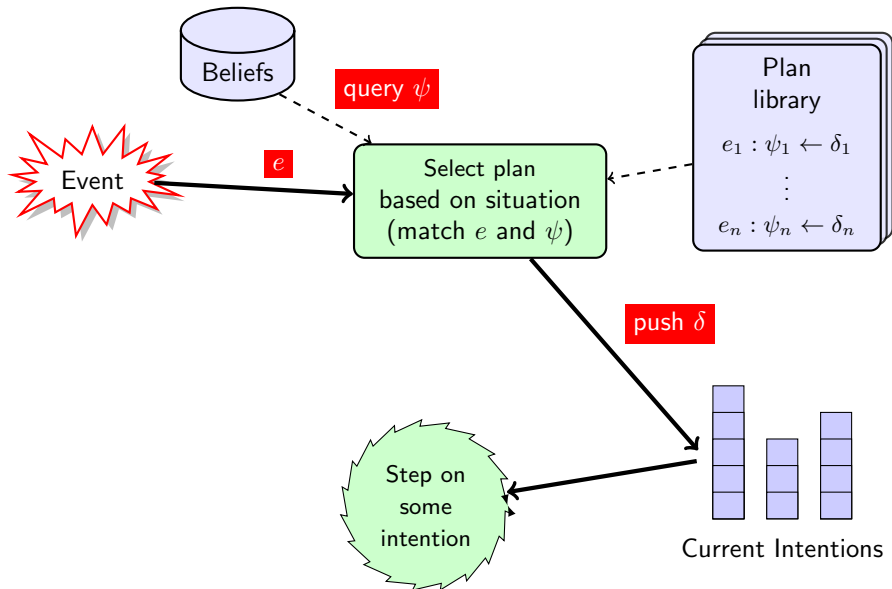
The BDI Execution Cycle [Rao&Georgeff 92]



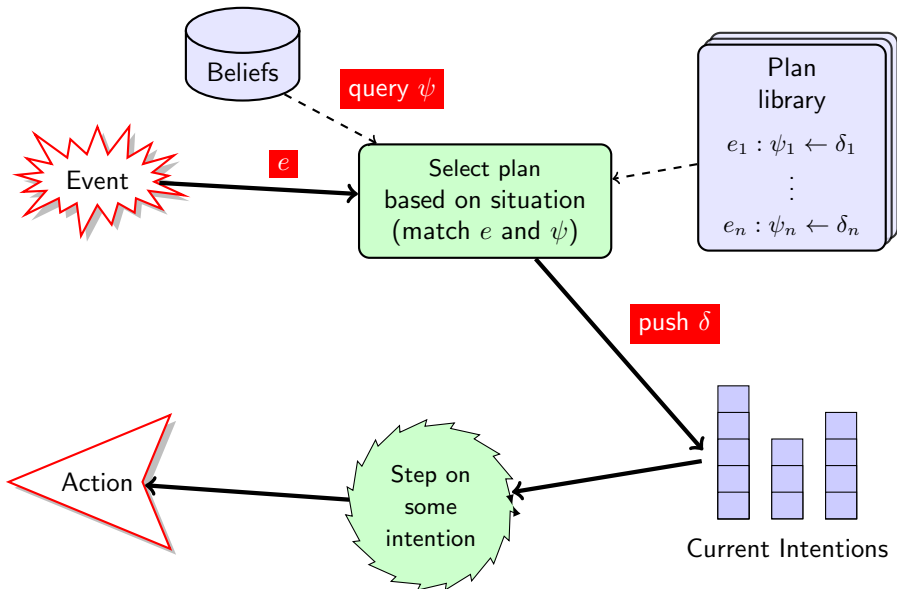
The BDI Execution Cycle [Rao&Georgeff 92]



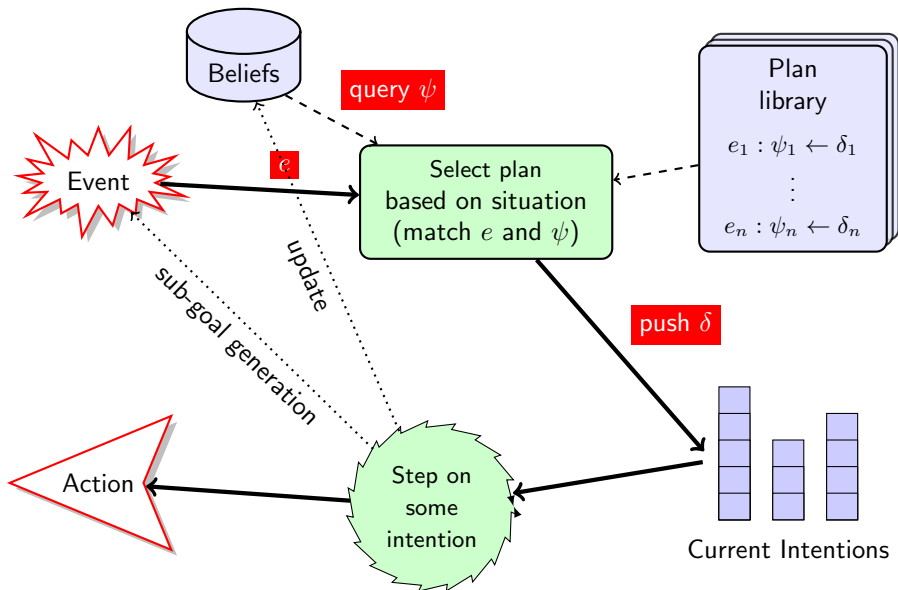
The BDI Execution Cycle [Rao&Georgeff 92]



The BDI Execution Cycle [Rao&Georgeff 92]



The BDI Execution Cycle [Rao&Georgeff 92]



Review so far...

1 Understanding Rational Behavior

- Intentional stance.
- Practical reasoning.

2 Agent-Oriented Programming

- agents as autonomous, proactive, reactive, flexible, etc. entities.
- agent technology to program complex intelligent systems;
- **Basic concepts of BDI programming:**
 - programming using mentalistic concepts: beliefs, desires, intentions.
 - goal-oriented programming – via **events**;
 - implicit programming – via **plan library & context conditions**;
 - rational execution cycle: on-the-fly recombination of plans.

Prometheus: Design & Analysis

Agent-oriented Programming

- agents;
- environments;
- percepts;
- actions;
- goals;
- beliefs;
- plans;
- messages;
- organizations;
- ...

Building a BDI Agent System

- What agents will be used?

player(s), coordinator, team leader, etc.

- Are there *external* entities/agents?

simulator, humans, etc.

- What information comes in/out from the system?

cell sensors, human instructions, movement actions, etc.

Building a BDI Agent System

- What agents will be used?

player(s), coordinator, team leader, etc.

- Are there *external* entities/agents?

simulator, humans, etc.

- What information comes in/out from the system?

cell sensors, human instructions, movement actions, etc.

- What are the events-goals to be used?

go to a location, block opponent, discover area, etc.

- What are the standard operational procedures of the domain?

random walk, picking gold, etc.

Building a BDI Agent System

- What agents will be used?

player(s), coordinator, team leader, etc.

- Are there *external* entities/agents?

referee, simulator, humans, etc.

- What information

agent actions, etc.

- What actions

move, block opponent, discover area, etc.

- What are the standard operational procedures of the domain?

random walk, picking gold, etc.

We need a principled analysis & design strategy!

Agent-oriented Analysis and Design

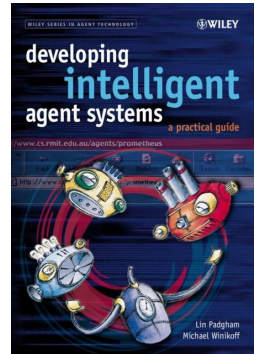
Dealt with in the area of Agent-Oriented Software Engineering (AOSE).

- Analysis and design methodologies usually consist of a set of **models** and a set of **guidelines** for using these.
- Should aid in understanding the system (often by formalizing (parts of) it) and designing it.
- A&D process typically evolves from abstract to concrete in the process.
- Tension between formal view of designer/programmer and informal user requirements.

Prometheus

- A specialized **methodology** for agent systems – uses agent concept.

A body of methods, rules, and procedures.

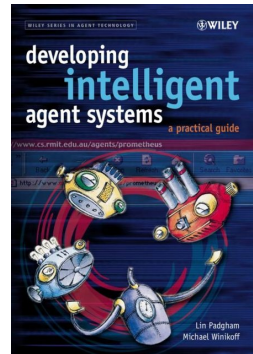


Prometheus

- A specialized **methodology** for agent systems – uses agent concept.

A body of methods, rules, and procedures.

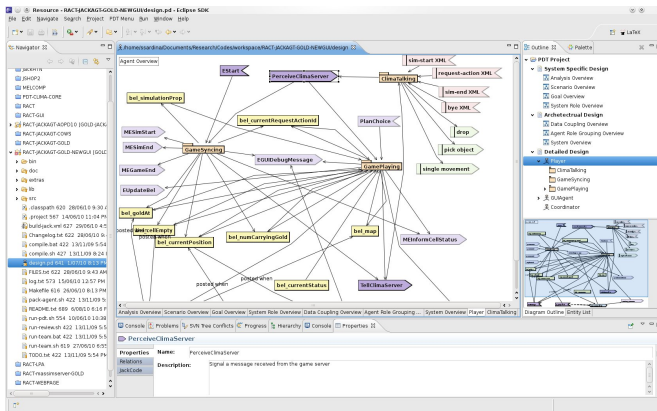
- Includes many standard aspects of SE: *cohesion, coupling, modularity, etc.*
- Has been developed as a result of experience in teaching and developing agent systems – both with students and with industry partners.
- Described in book “*Developing Intelligent Agent Systems: A practical guide*” by Lin Padgham and Michael Winikoff, 2004, Wiley.



Prometheus Design Tool (PDT)

Nearly impossible to design+develop largish systems without support tool:

- **Design:** of an agent system in 3 interrelated phases.
- **Crosscheck:** on-demand consistency checking.
- **Code generation:** skeleton code in JACK agent language.

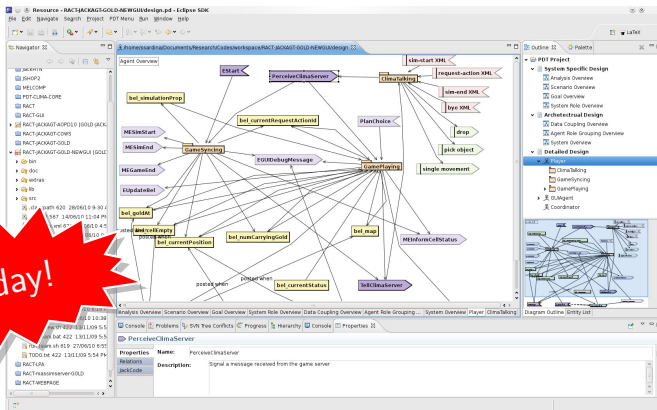


www.cs.rmit.edu.au/agents/pdt/

Prometheus Design Tool (PDT)

Nearly impossible to design+develop largish systems without support tool:

- **Design:** of an agent system in 3 interrelated phases.
- **Crosscheck:** on-demand consistency checking.
- **Code generation:** skeleton code in JACK agent language.



Today!

www.cs.rmit.edu.au/agents/pdt/

Developing your Agent: 3 Components

In developing our agent systems we use the following components:

PDT to do the analysis & design of the system;
to generate skeleton code.

ECLIPSE to do actual JACK programming (from the skeleton code).

JACK to compile & run the system.

The JACK BDI Programming Language

1 JACK Agent Language

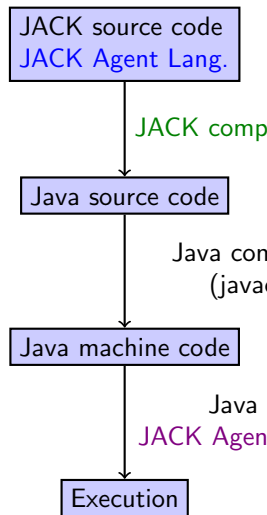
- Used to describe an agent-oriented software system.
- Super-set of Java (agent-oriented features extensions).

2 The JACK Agent Compiler

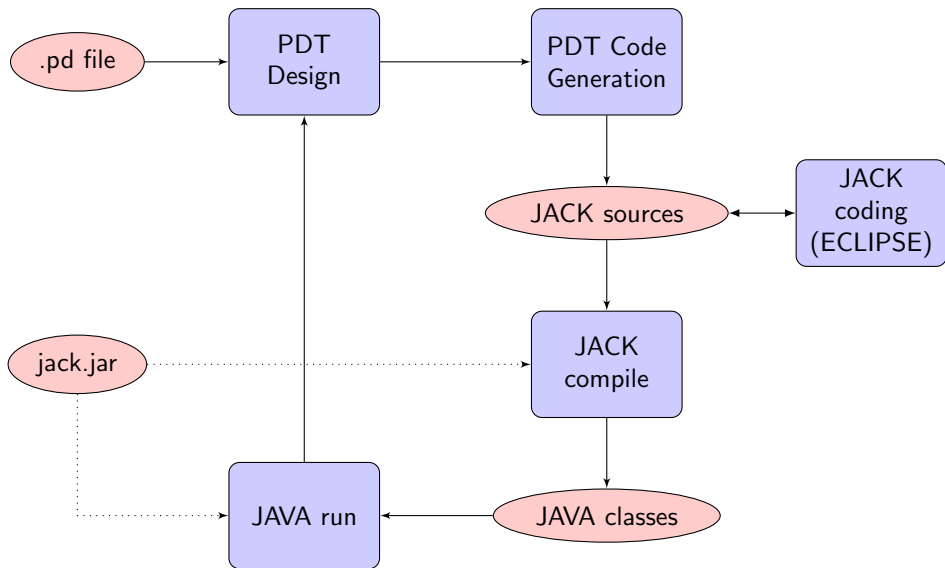
- Converts JACK Agent Language into pure Java.
- Java source can be compiled into Java VM code.

3 The JACK Agent Kernel

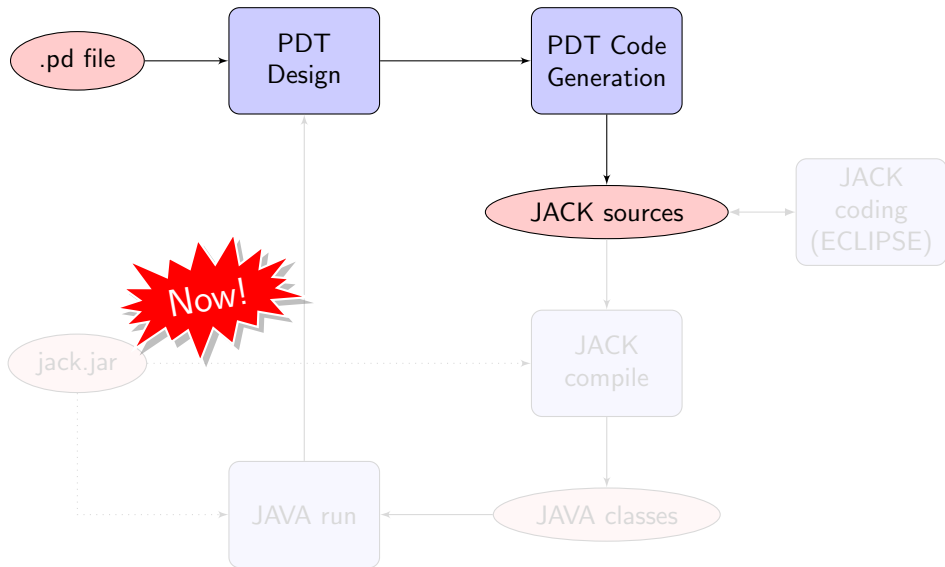
- Runtime engine for programs written in the JACK Agent Language.
- Set of classes that give JACK Agent Language programs their agent-oriented functionality.
- Run behind the scenes.
- Implement the underlying infrastructure and functionality for agents.



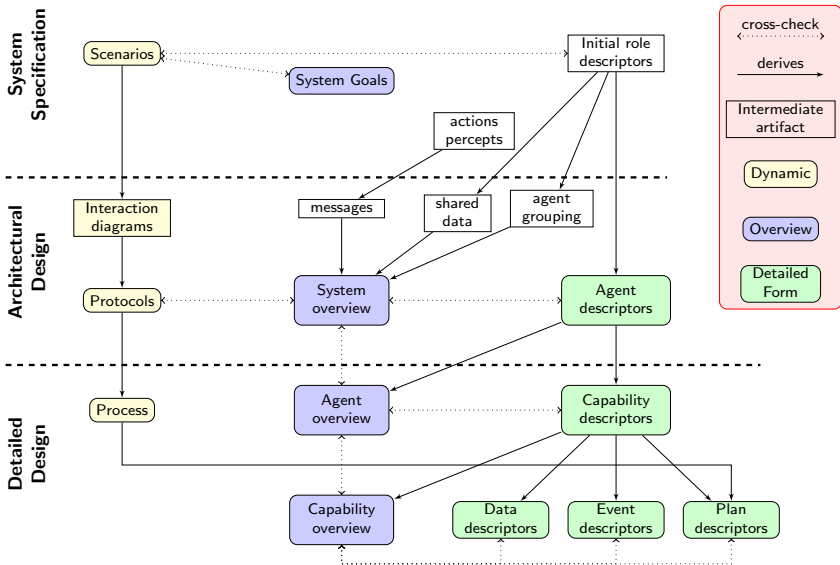
Developing the Agent System: PDT + JACK + ECLIPSE



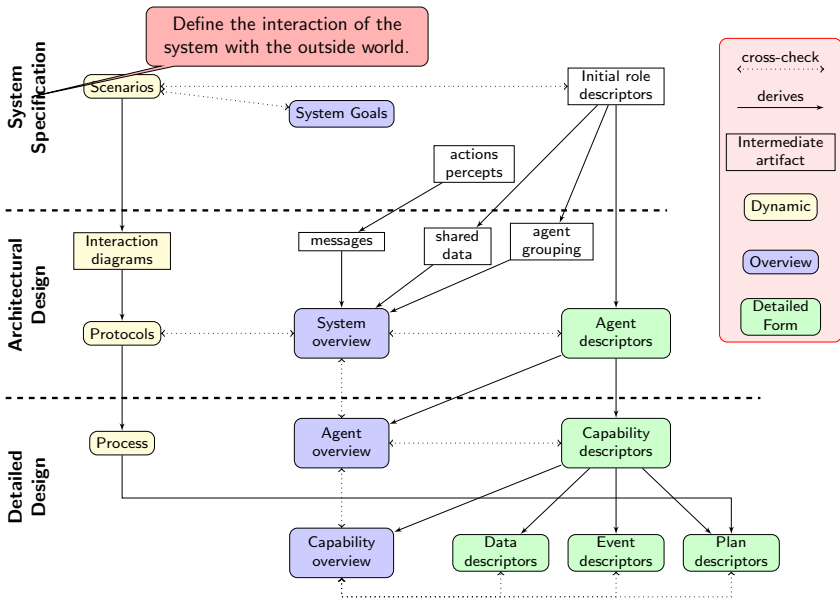
Developing the Agent System: PDT + JACK + ECLIPSE



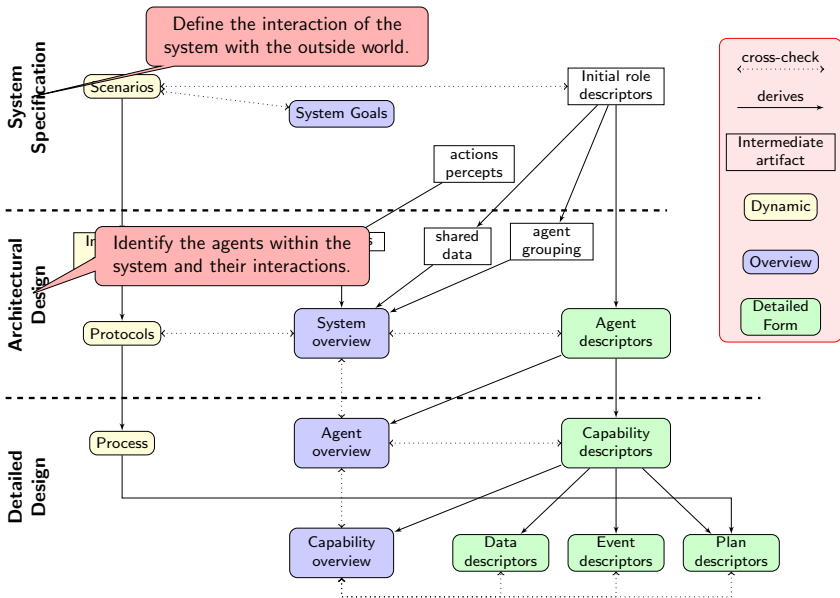
Prometheus Overview: 3 Phases



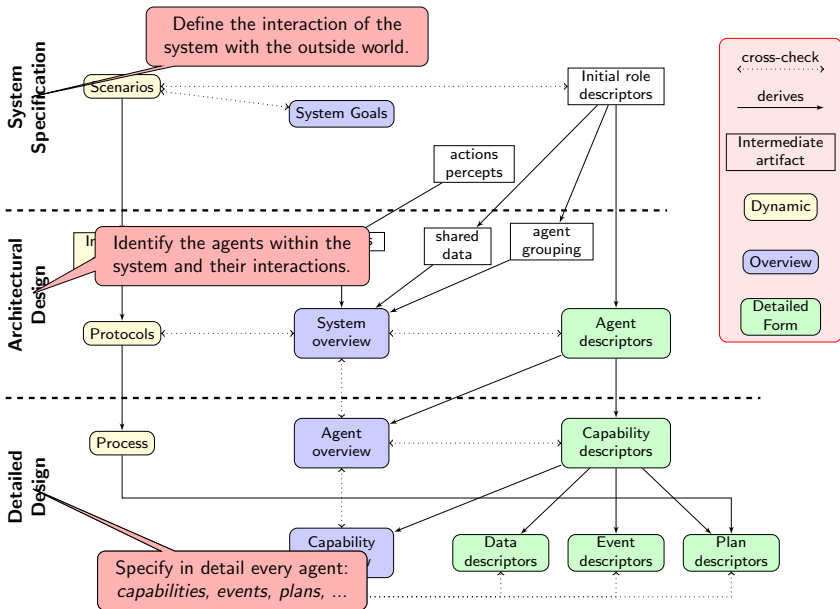
Prometheus Overview: 3 Phases



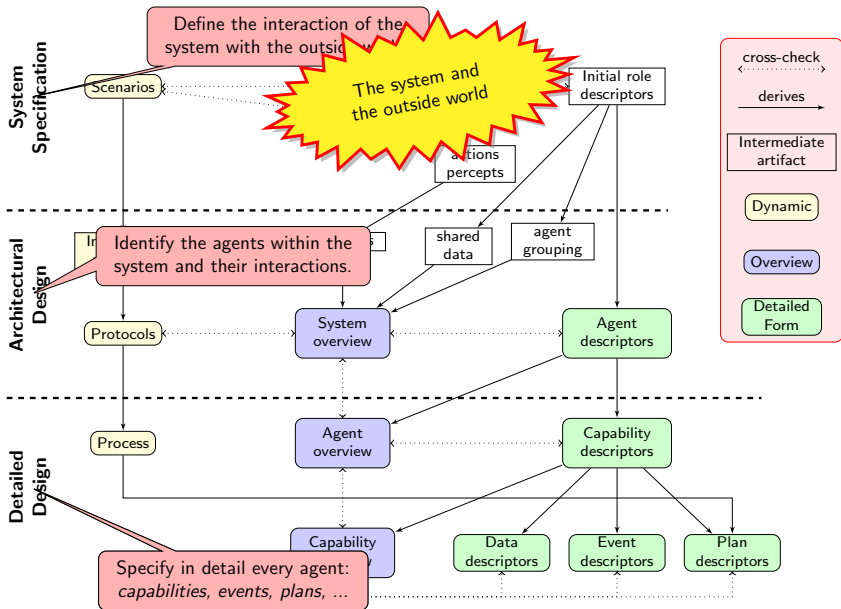
Prometheus Overview: 3 Phases



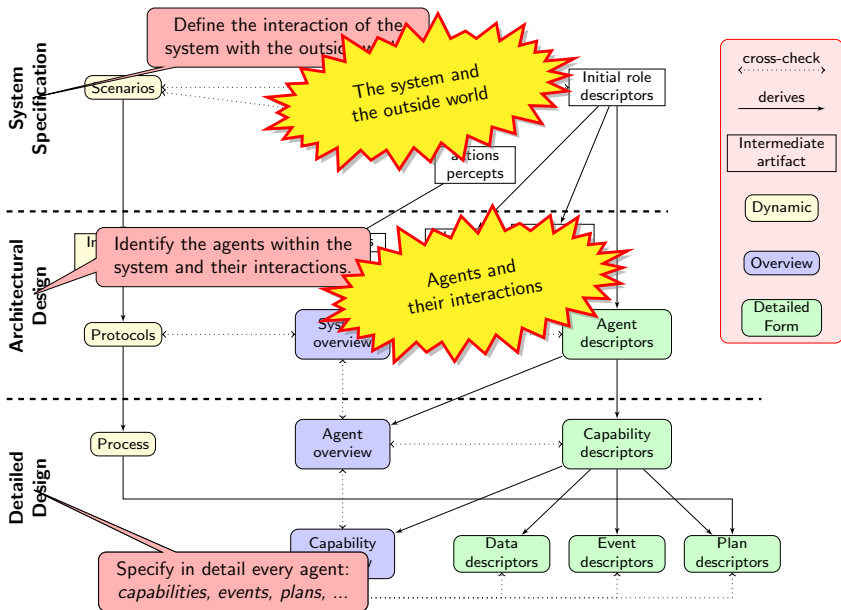
Prometheus Overview: 3 Phases



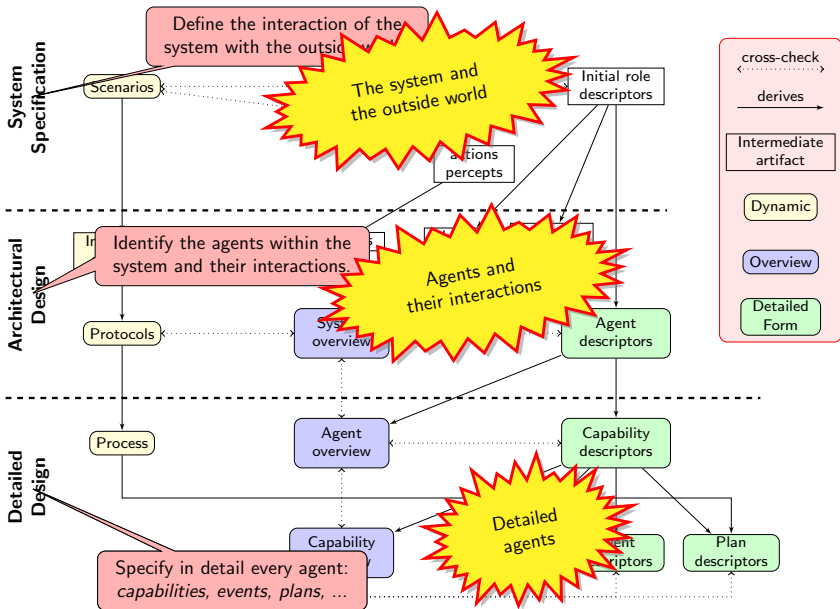
Prometheus Overview: 3 Phases



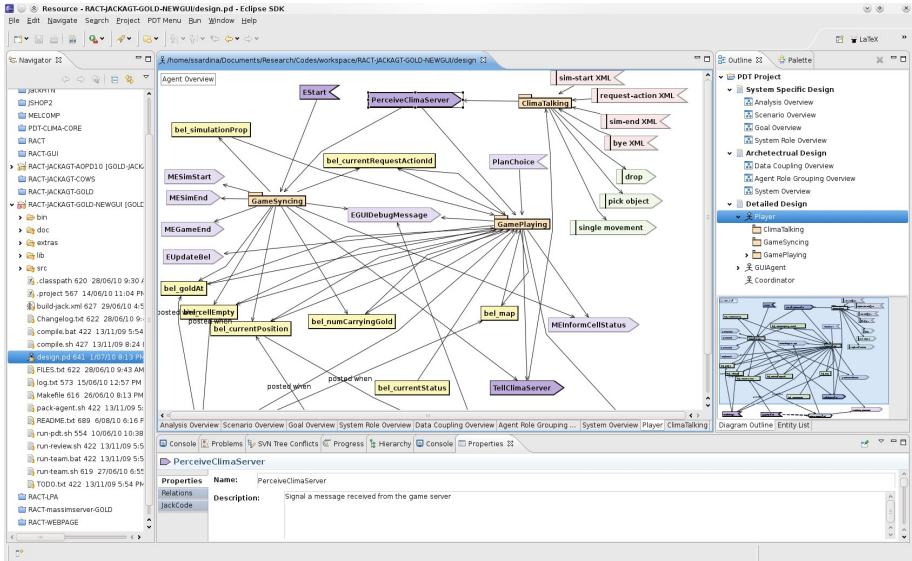
Prometheus Overview: 3 Phases



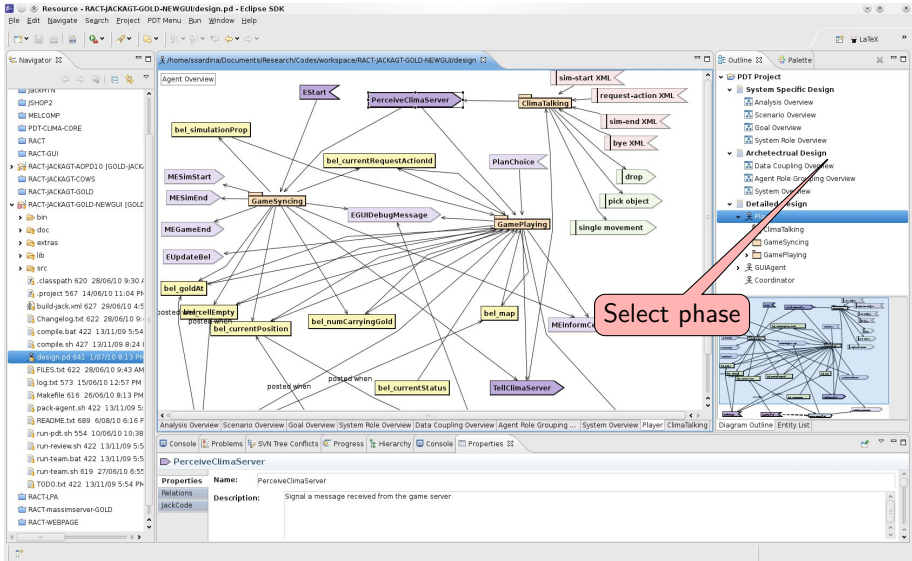
Prometheus Overview: 3 Phases



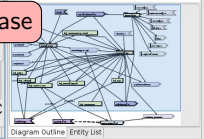
Prometheus Design Tool (PDT)



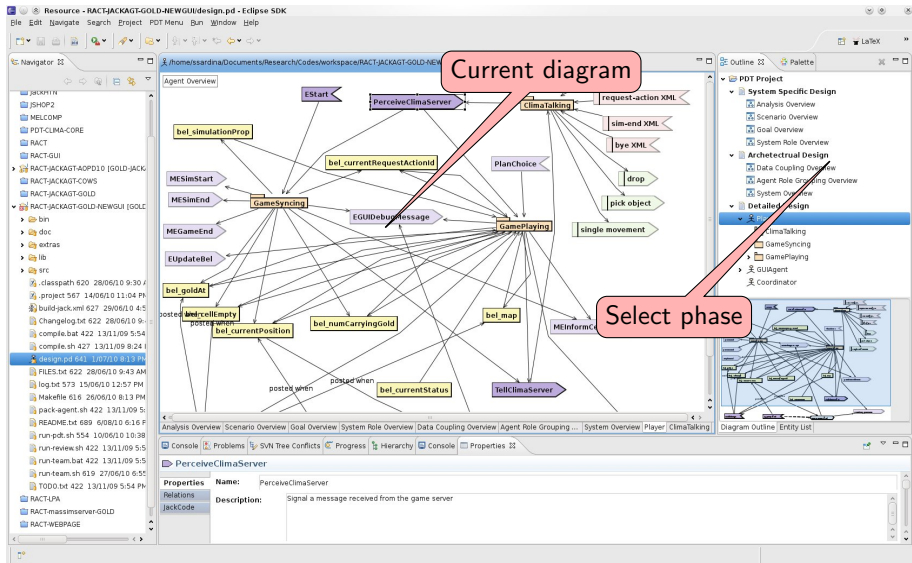
Prometheus Design Tool (PDT)



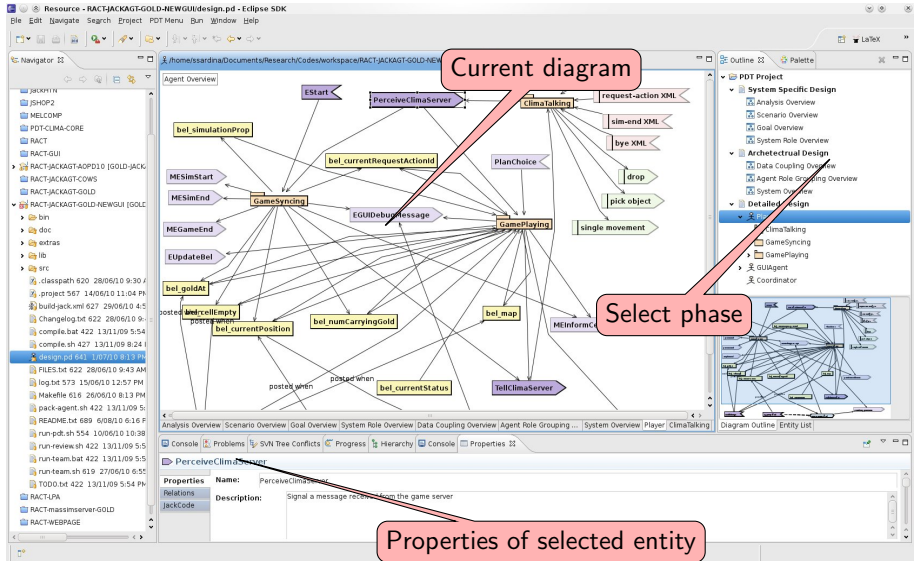
Select phase



Prometheus Design Tool (PDT)



Prometheus Design Tool (PDT)



Prometheus Design Tool (PDT)

The screenshot displays the Prometheus Design Tool (PDT) interface, which is used for designing BDI systems. The interface is divided into several panes:

- File Navigator:** Located on the left, it shows a hierarchical view of the project files. The file `design.pd` is selected.
- Current diagram:** The central pane displays a complex interaction diagram. It features various entities such as `PerceiveClimaServer`, `ClimaTalking`, `GamePlaying`, and `TellClimaServer`. The diagram is composed of numerous nodes and edges, representing the system's behavior. A red arrow points to the `GamePlaying` entity, which is highlighted in the diagram.
- Select phase:** A red arrow points to the `GamePlaying` entity, indicating the selected phase of the diagram.
- Properties of selected entity:** A red arrow points to the `PerceiveClimaServer` entity in the diagram. Below the diagram, the **Properties** panel shows the details for this entity, including its name and description.

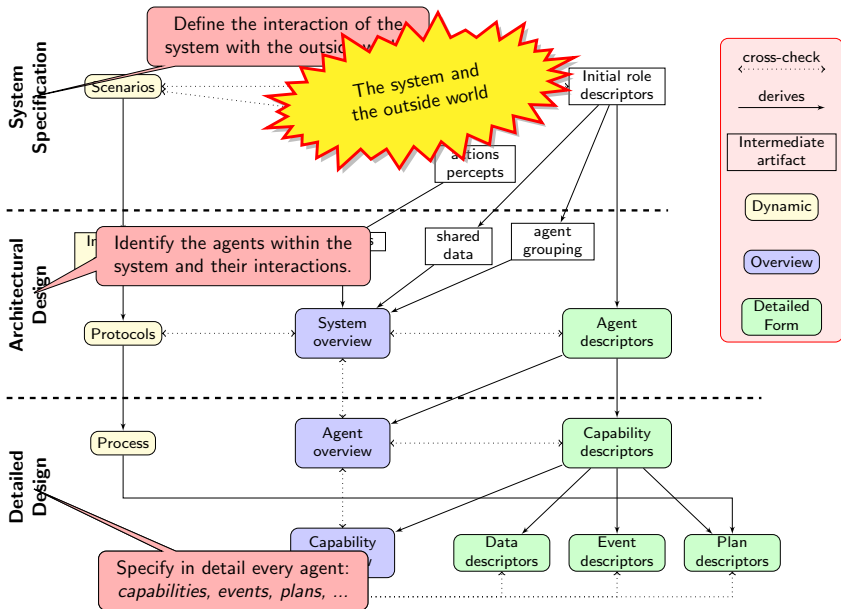
The **Properties** panel for `PerceiveClimaServer` shows:

- Name:** `PerceiveClimaServer`
- Description:** Signal a message received from the game server

The **Diagram Outline** pane on the right shows a hierarchical view of the diagram's structure, including the `GamePlaying` entity.

System Specification

Prometheus Overview: 3 Phases



System Specification: *The system and the outside world*

Consists of a number of interleaving, iterative steps, to define the **roles, goals, scenarios, actors, actions and percepts**.

Analysis Overview Identify the actors and scenarios around which they interact with the system. Identifying the actions and percepts forming the interface to the system.

Scenarios Details of the scenarios illustrating the systems operation.

Goal Overview Identifying the system goals and sub-goals

System Roles Grouping goals and other items into the basic roles of the system.

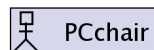
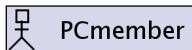
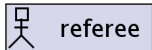
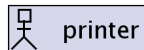
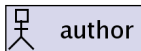
Conference Management Example

- 1 Used substantially in the literature.
- 2 Easy for academics to understand.
- 3 Adequate for illustration purposes.

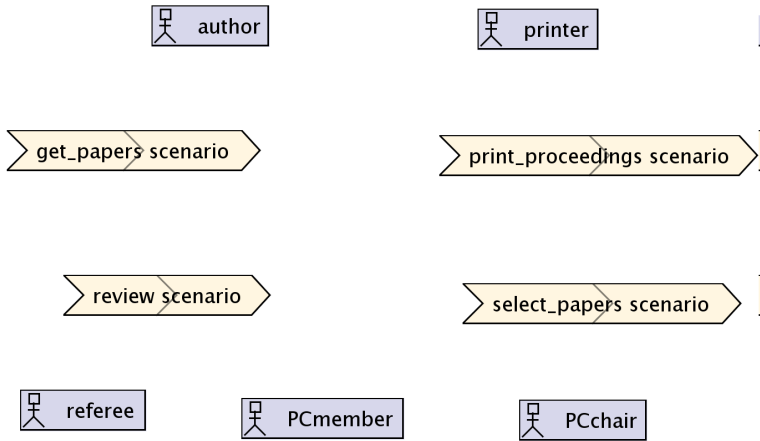
Description

Authors submit papers and get an ID. PC members are responsible for finding reviewers who then review papers. PC members make recommendations to the PC chair who makes final selections. Authors are notified. Final versions of accepted papers are collected and published.

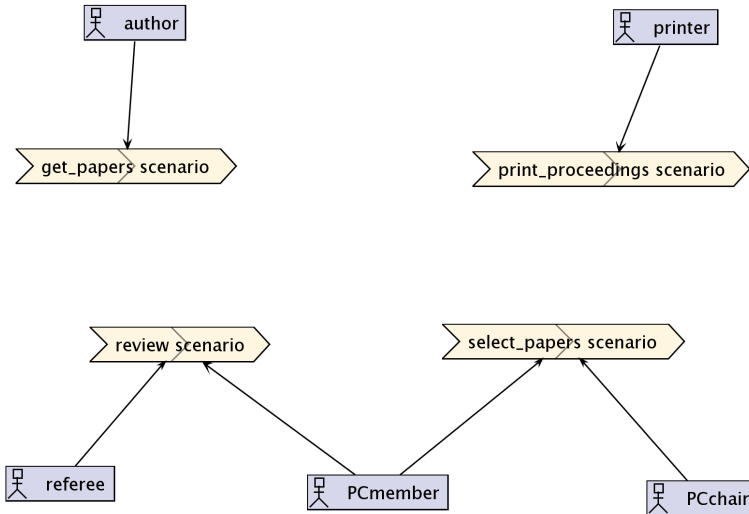
Identify the Actors



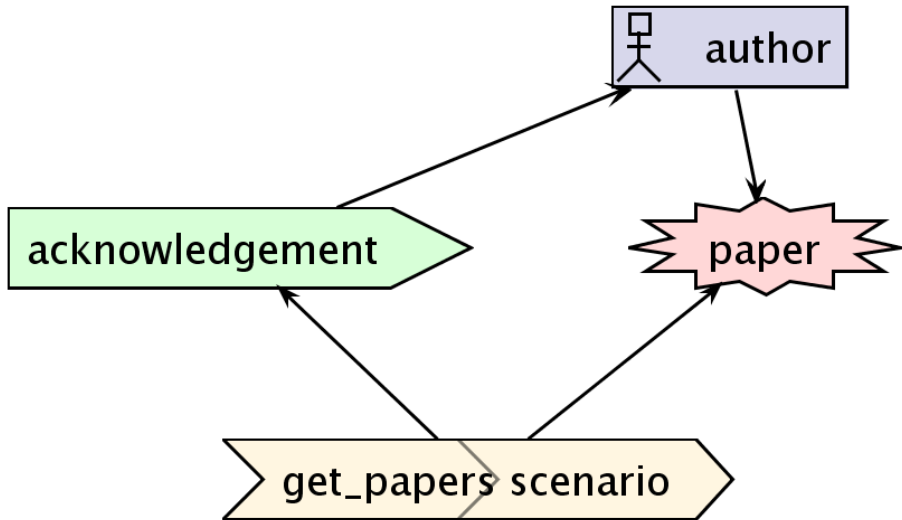
Identify the Scenarios



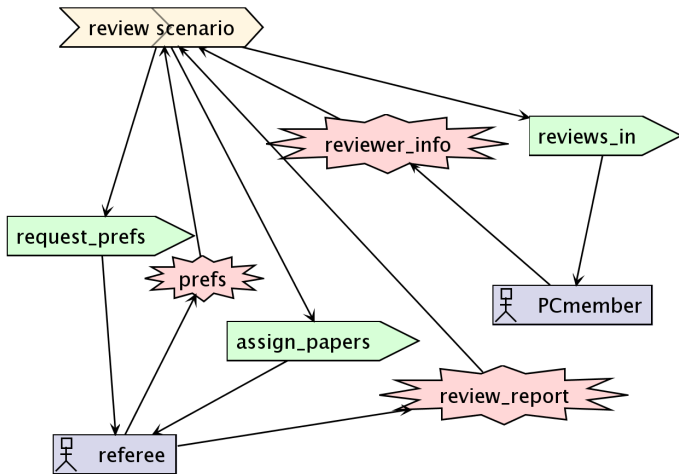
Link Actors and Scenarios



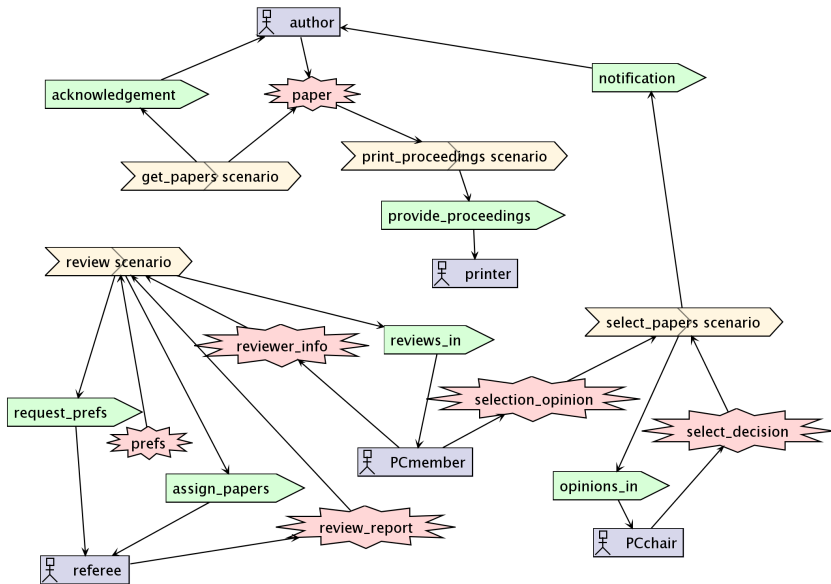
Add Actions and Percepts: Submission



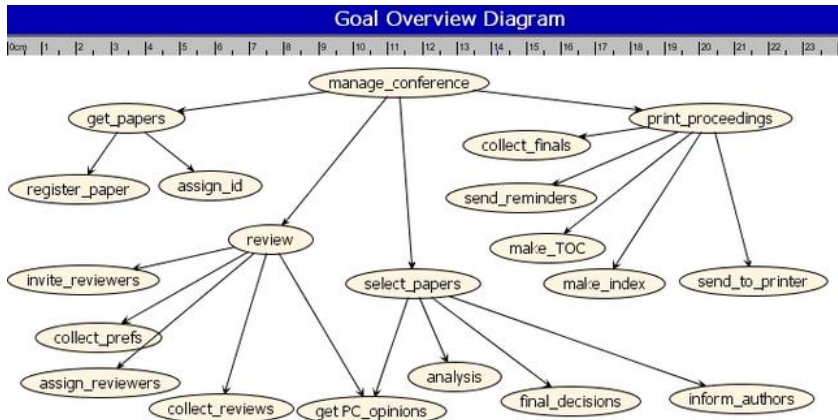
The “Review” Scenario



Completed Analysis Overview

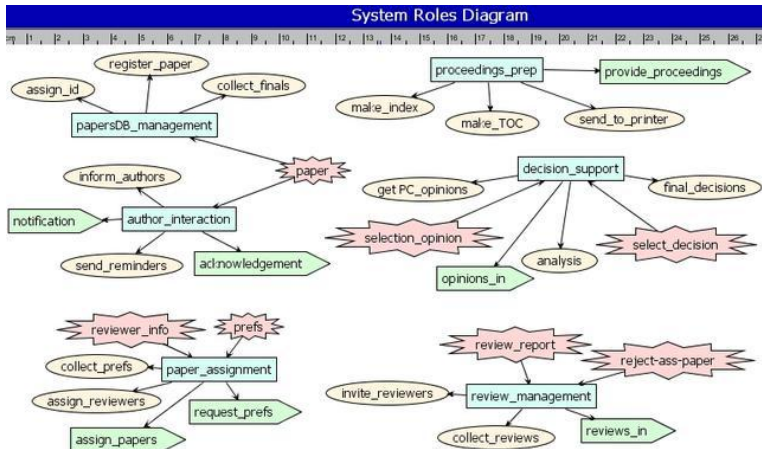


Goal Overview Diagram



- Coupled with the Scenarios Diagram; identifying the goals of the system.
- For each goal, identify the sub-goals: **how can we achieve this goal?**
- Sub-goals are either “AND” or “OR” (default is “AND”)

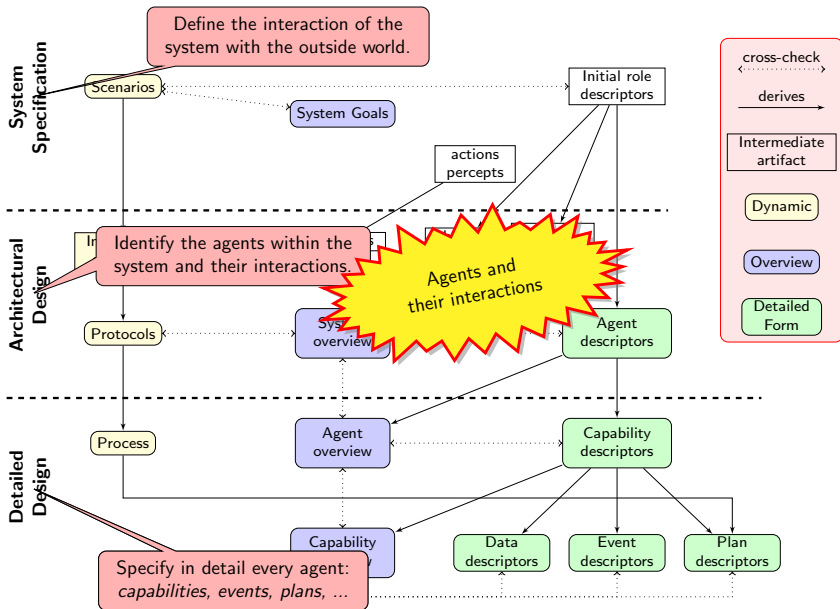
System Role Diagrams



- Goals are then grouped into roles (kind of “responsibilities”).
- Actions and percepts also attached (from scenarios & analysis overview).

Architectural Design

Prometheus Overview: 3 Phases



Architectural Design: *Agents and their interactions*

Uses the artifacts produced in the system specification phase to **identify the agents** within the system and **their interactions**.

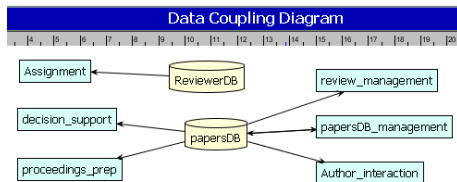
Data Coupling Couple the data with the roles that access them.

Agent-Role grouping Group roles into agents using standard SE principles of cohesion and modularity.

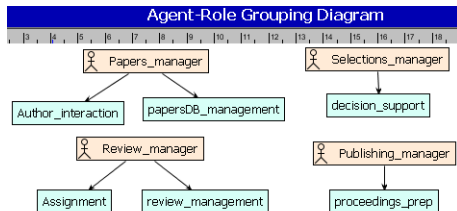
System Overview Provides an overview of the internals of the system in term of the agents, interaction protocols, actions, percepts, and shared data.

Data Coupling & Agent-Role Grouping

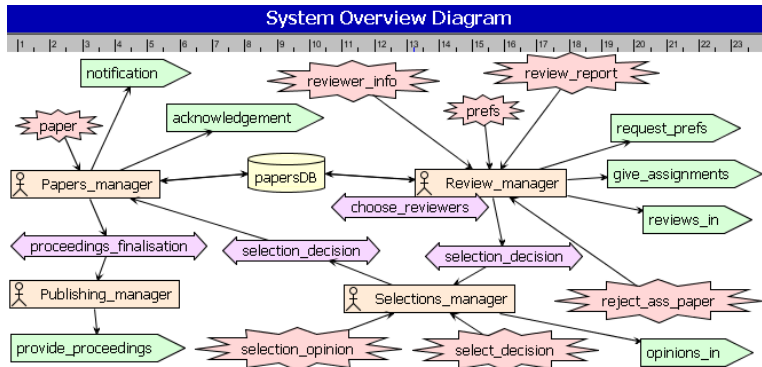
- 1 Data are coupled with Roles that read and write from them.



- 2 Roles are assigned to Agents.



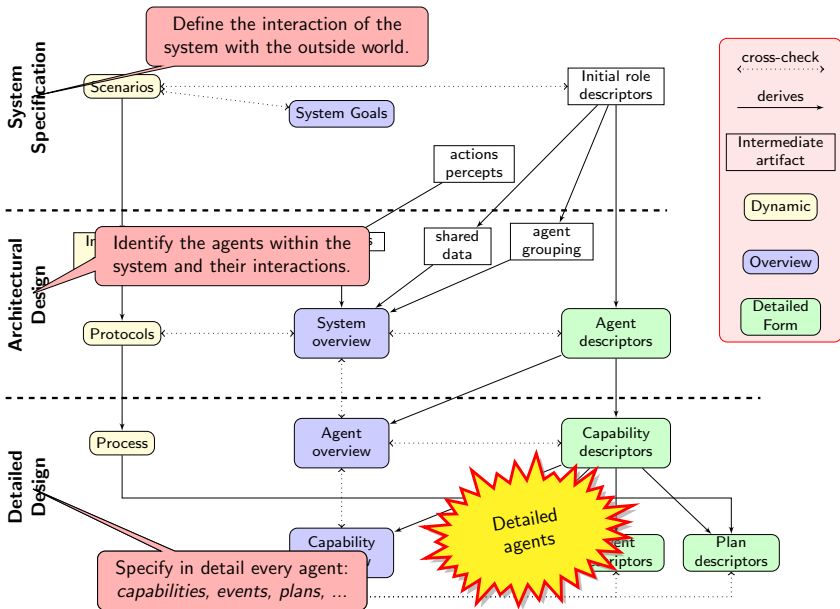
System Overview Diagram



- Provides an overview of the system.
- The links to percepts and actions are propagated from the coupling of agents to roles for consistency.
- The interaction protocols are developed in this diagram.
- Only shared data is shown.

Detailed Design

Prometheus Overview: 3 Phases



Detailed Design: *Detailed agents*

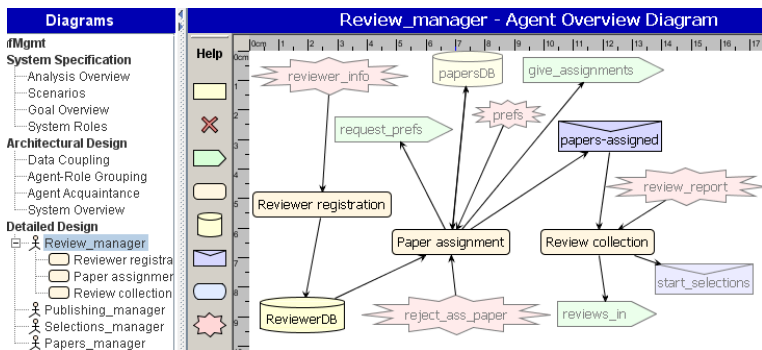
Develops the **specification of every agent** is the system to the level that it may be easily translated into agent code.

Each agent is refined in terms of **capabilities, internal events, plans and internal data structures**.

Agent Overview The agent overview diagram captures the capabilities within an agent and any event between capabilities as well as associated data.

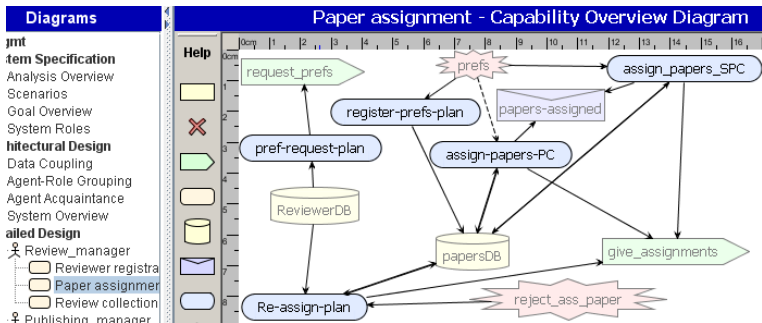
Capability Overview Describes the plans and the events that are handled by the plans

Agent Overview Diagram



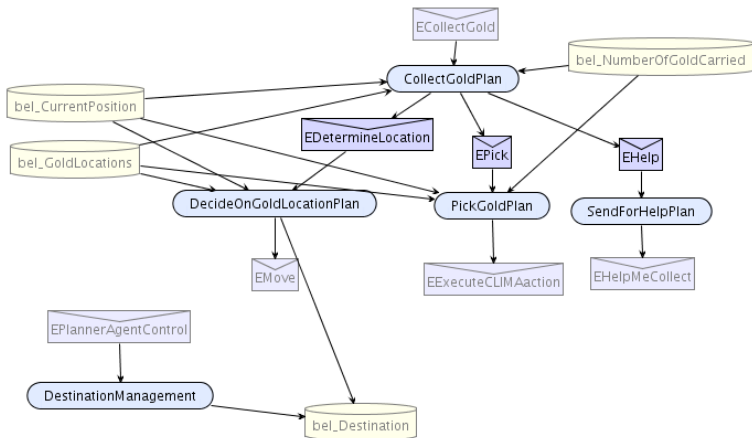
- Provides an overview of the capabilities of the agent.
- Percepts and actions: propagated from architectural design (via roles).
 - *Percepts are inputs* and from external entities (actors);
 - *Actions are output* to external entities.
- Faded entities: those that were automatically propagated.

Capabilities Overview



- The internals of the capability are described.
- Plans to handle each input are developed.
- The automatic propagation provides consistency between the architectural design and the detailed design.

Capabilities Overview (Agent Contest)



Review

In this lecture we have seen an overview of the three phases in the Prometheus Methodology:

- 1 **System specification:** interaction with the outside world.
 - identify roles, goals, scenarios, actors, actions and percepts.
- 2 **Architectural design:** agents and their internal interactions.
 - identify agents and their interactions.
- 3 **Detailed design:** details of every agent.
 - identify details of agents: capabilities, plans, events, data structures.


Protocols

Agent-oriented Programming

- **agents;**
- environments;
- percepts;
- actions;
- goals;
- beliefs;
- plans;
- **messages;**
- organizations;
- ...

Agent-oriented Programming


- **agents;**
- environments;
- percepts;
- actions;
- goals;
- beliefs;
- plans;
- **messages;**
- organizations;
- ...



What agents will be used?

Agent-oriented Programming

- **agents;**
- environments;
- percepts;
- actions;
- goals;
- beliefs;
- plans;
- **messages;**
- organizations;
- ...

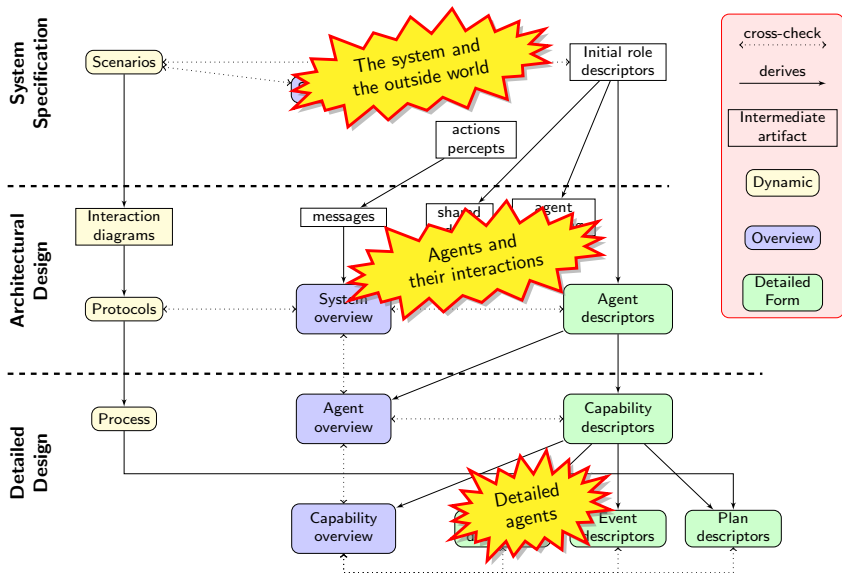


What agents will be used?

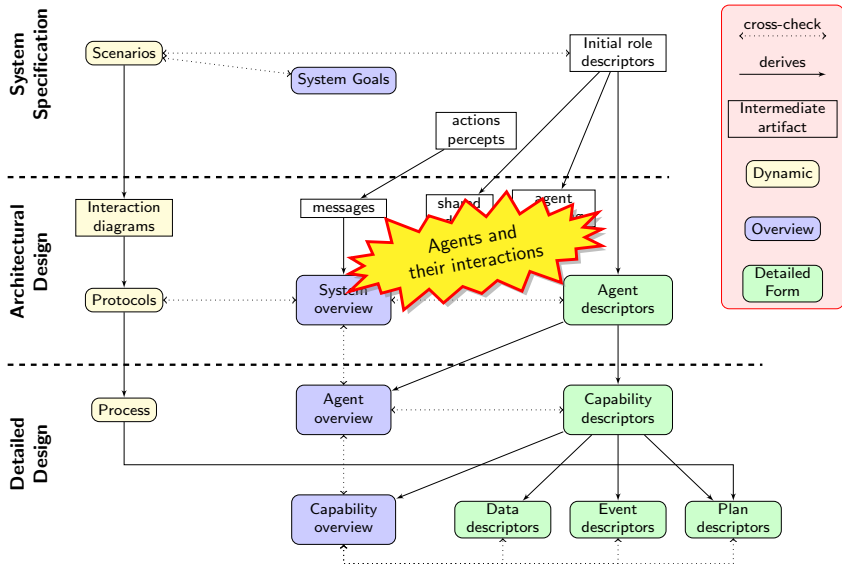


How will these agents communicate?

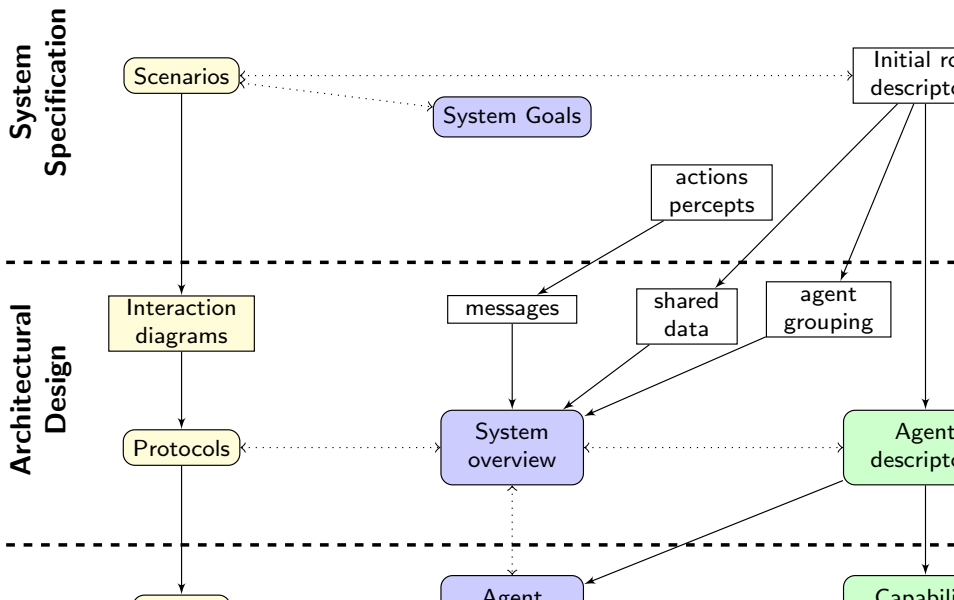
Prometheus Overview: 3 Phases



Prometheus Overview: 3 Phases

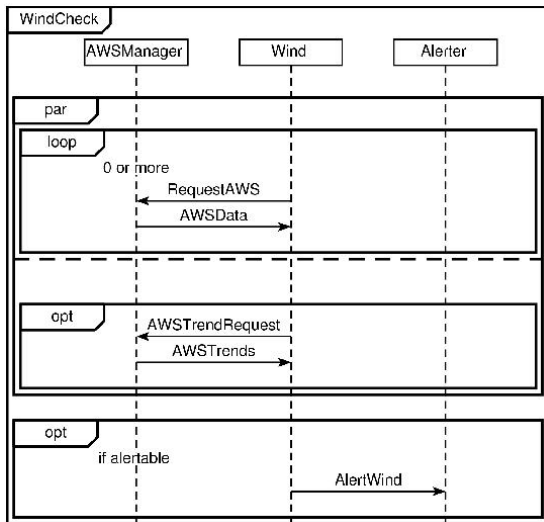


Prometheus Overview: 3 Phases

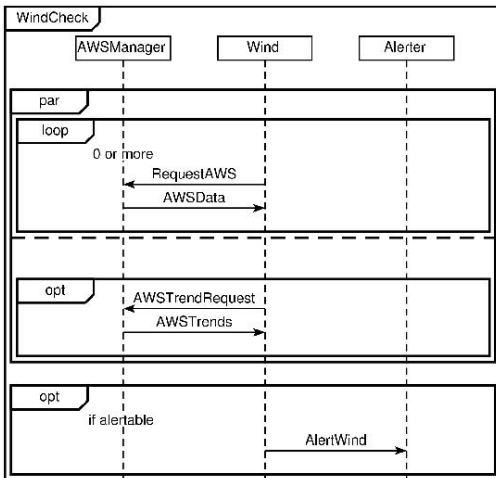


Protocols

- Specify agent interactions.
- Use **AUML2** like syntax.
- AUML2/AUML: standards developed by FIPA, agents standards body.
- AUML2 still a draft...
- UML2 contained much of what was needed;
- Web service standards are being developed and agent standards need to fit with these.

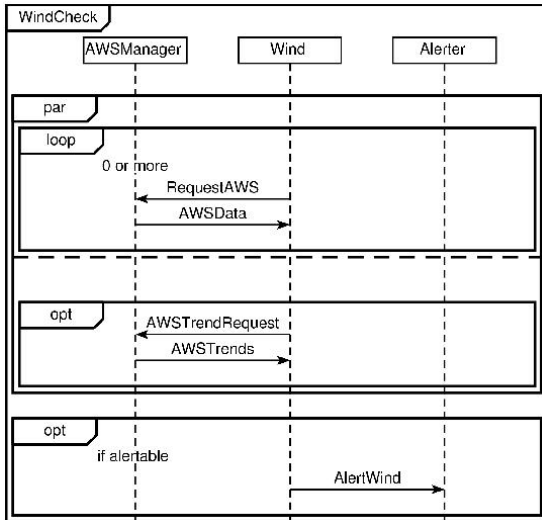


Protocols (cont.)



- Protocols have **time going down** the page.
- Have a range of **boxed control constructs** which can be nested:
 - alternative; loop; parallel; optional; etc.
- **Dotted lines** separate different parts of a construct such as alt or parallel.
- **Guards** indicate conditions on box constructs
- See summary in appendix of textbook!

Protocols (cont.)

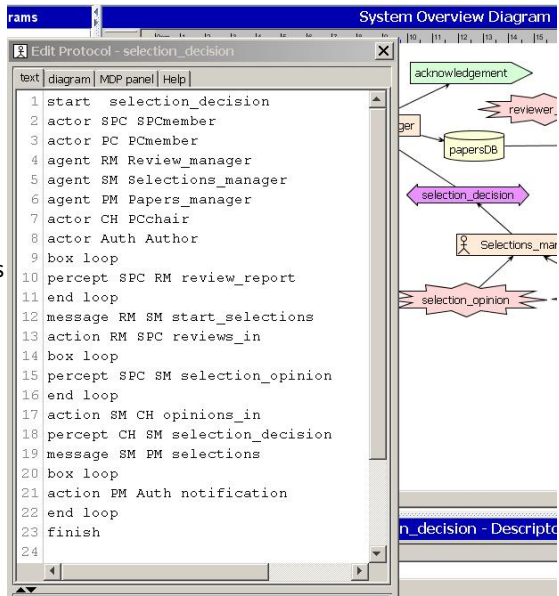


Additions to AUML2:

- Lifelines for actors (and/or environment).
- Percepts and actions.
- Messages.
- Use agents rather than roles.

Protocol Editor in PDT

- Protocols are created and edited via a textual editor.
- Access the protocol editor via the Tools menu.
- The help tab provides guidelines for the text syntax.
- The diagram tab shows the AUML style protocol.
- The links between agents in the system overview diagram are generated automatically according to the protocol spec.



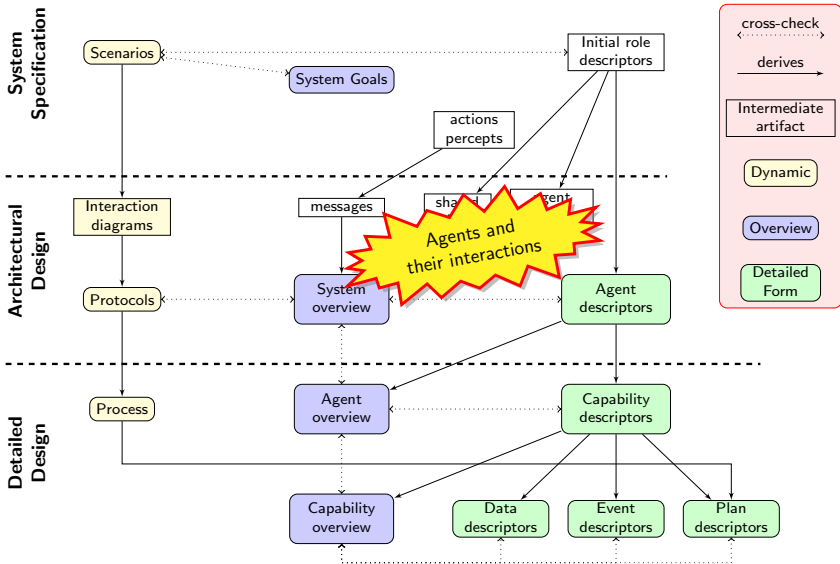
Interaction Diagrams

Determining Protocols

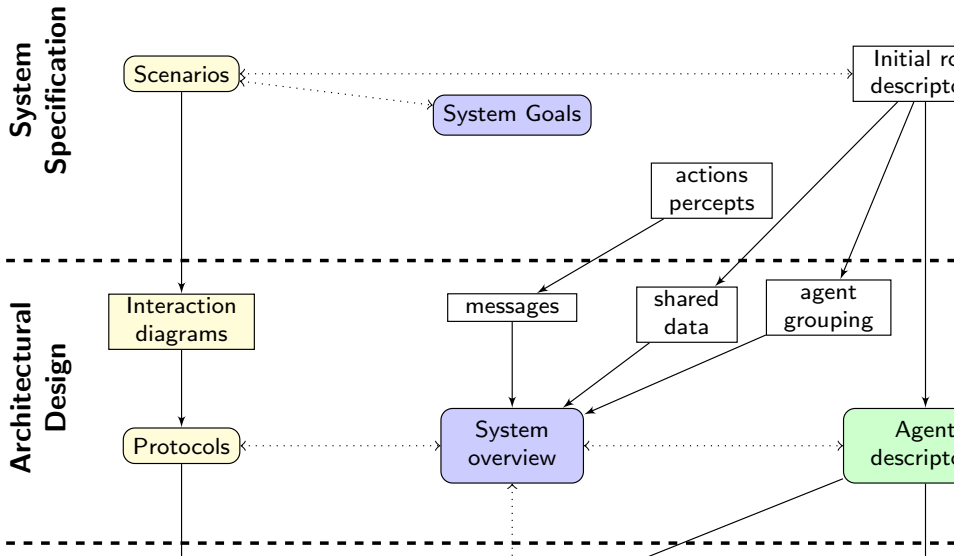
How to develop the protocols between agents? Where to start?

- Provide the rules for a **piece of interaction** in the system.
- Scenarios are often a good starting point.
- Can use **interaction diagrams** as intermediate step between scenarios and protocols.
- Prometheus methodology provides some **guidelines** for doing this.

Prometheus Overview: 3 Phases



Prometheus Overview: 3 Phases



Scenarios to Interaction Diagrams

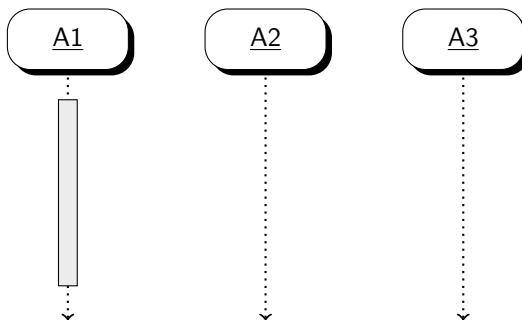
Idea: Start from one possible execution/trace and then generalize...

- 1 Take scenario steps and note agent against role.
- 2 Where next step is different agent, likely to need a message (heuristic rather than rule)
- 3 Agent must receive some trigger before starting to participate.

Interaction diagrams (like scenarios) show just one execution/trace.

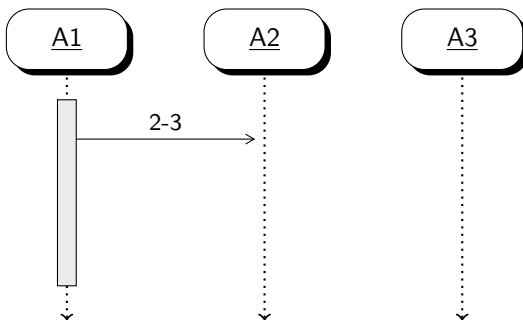
Deriving Interaction Diagrams

STEP	DESCRIPTION	ROLE	AGENT
1	Place delivery request	R1	A1
2	Log outgoing delivery	R2	A1
3	Log books outgoing	R3	A2
4	Update customer profile	R4	A3
5	Send email	R6	A3



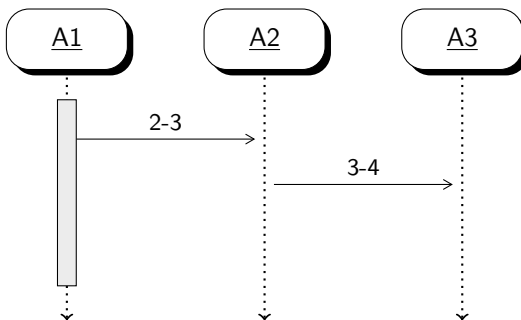
Deriving Interaction Diagrams

STEP	DESCRIPTION	ROLE	AGENT
1	Place delivery request	R1	A1
2	Log outgoing delivery	R2	A1
3	Log books outgoing	R3	A2
4	Update customer profile	R4	A3
5	Send email	R6	A3



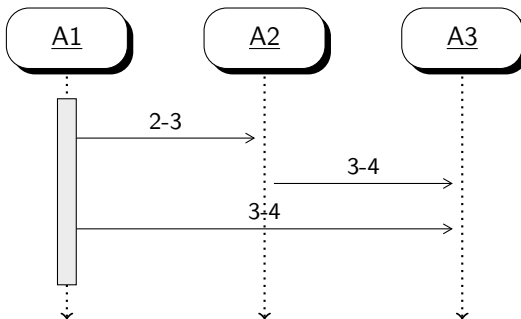
Deriving Interaction Diagrams

STEP	DESCRIPTION	ROLE	AGENT
1	Place delivery request	R1	A1
2	Log outgoing delivery	R2	A1
3	Log books outgoing	R3	A2
4	Update customer profile	R4	A3
5	Send email	R6	A3



Deriving Interaction Diagrams

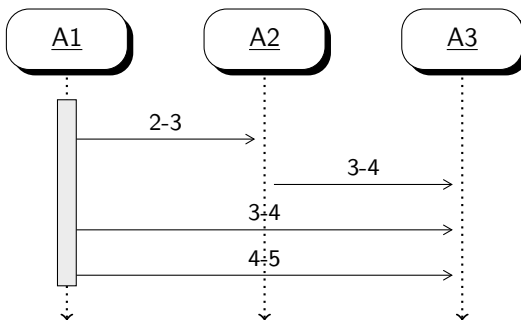
STEP	DESCRIPTION	ROLE	AGENT
1	Place delivery request	R1	A1
2	Log outgoing delivery	R2	A1
3	Log books outgoing	R3	A2
4	Update customer profile	R4	A3
5	Send email	R6	A3



message may
come from
earlier agent!

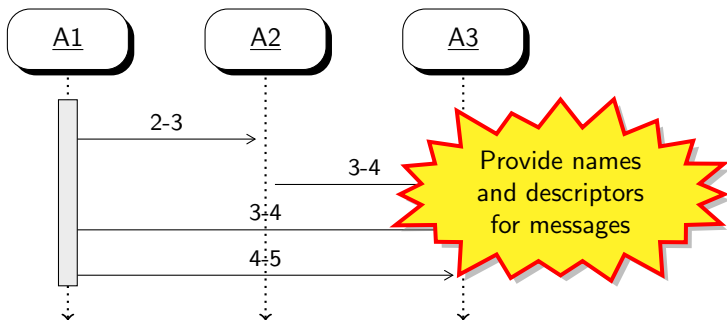
Deriving Interaction Diagrams

STEP	DESCRIPTION	ROLE	AGENT
1	Place delivery request	R1	A1
2	Log outgoing delivery	R2	A1
3	Log books outgoing	R3	A2
4	Update customer profile	R4	A3
5	Send email	R6	A3



Deriving Interaction Diagrams

STEP	DESCRIPTION	ROLE	AGENT
1	Place delivery request	R1	A1
2	Log outgoing delivery	R2	A1
3	Log books outgoing	R3	A2
4	Update customer profile	R4	A3
5	Send email	R6	A3



Scenarios to Interaction Diagrams

Idea: Start from one possible execution/trace and then generalize...

- 1 Take scenario steps and note agent against role.
- 2 Where next step is different agent, likely to need a message (heuristic rather than rule)
- 3 Agent must receive some trigger before starting to participate.

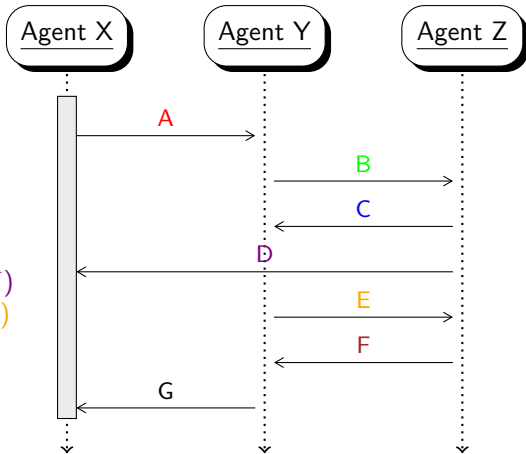
Interaction diagrams (like scenarios) show just one execution/trace.

Deriving Interaction Diagrams

Assume: X takes roles R1 & R2; Y takes role R3; Z takes roles R4 & R5

Scenario:

G1	R1	X	
G2	R2	X	
G3	R3	Y	<i>A</i>
G4	R4	Z	<i>B</i>
G5	R5	Z	
G6	R3	Y	<i>C</i>
G7	R2	X	<i>D (from X)</i>
G8	R5	Z	<i>E (from Y)</i>
G9	R3	Y	<i>F</i>
G0	R1	X	<i>G</i>

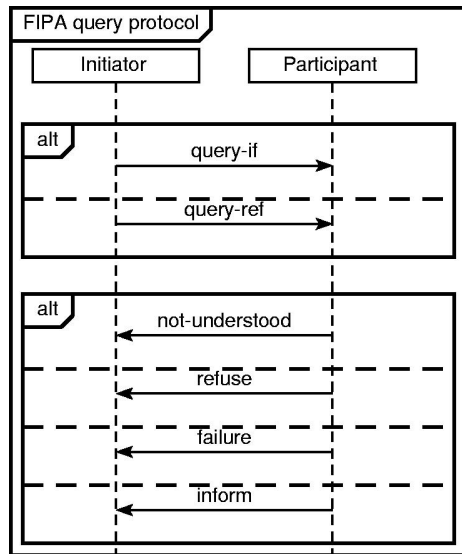
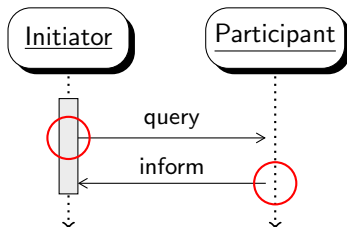


Generalizing to Protocols

- Interaction diagrams are only *partial*.
 - they just describe one possible (typical) “run”
- Need to be generalized to give full specification.
- **Process:** Consider alternatives at each point

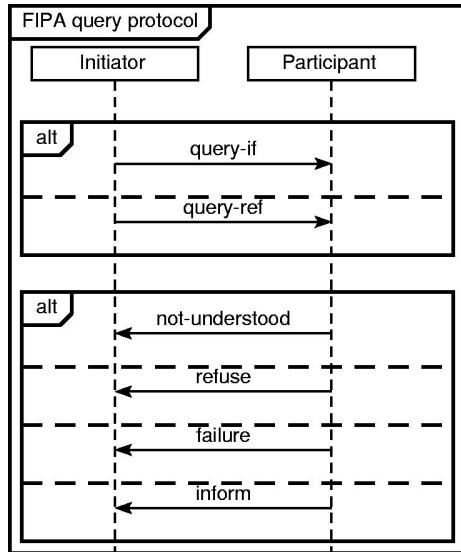
Protocol Editor in PDT

At each point, **consider options....**

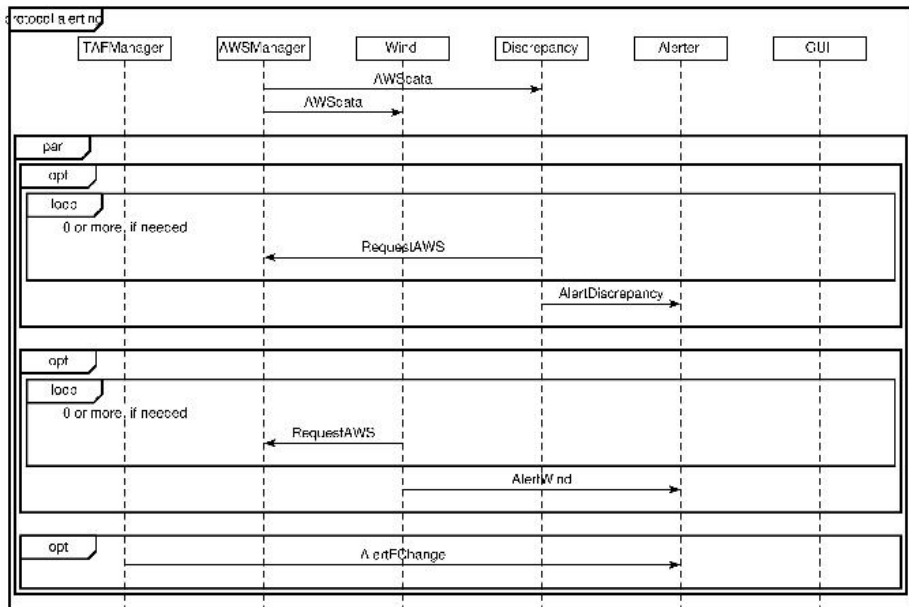


AUML-2

```
start FIPA query protocol
agent I Initiator
agent P Participant
box alt
  message I P query-if
  next
  message I P query-ref
end alt
box alt
  message P I not-understood
  next
  message P I refuse
  next
  message P I failure
  next
  message P I inform
end alt
```



Prometheus Design Tool (PDT)



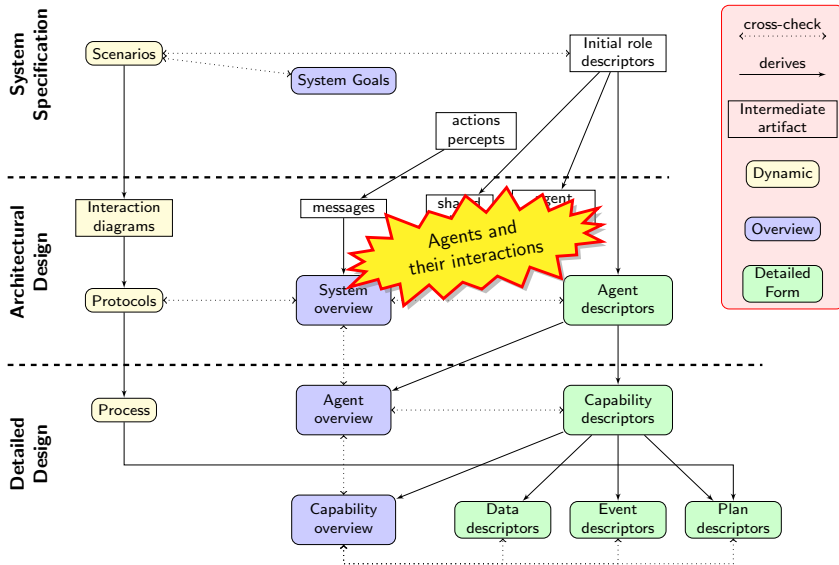
Protocol Descriptors

- Name
- Description
- Included Messages (*give source and destination agents*)
 - e.g., request(agent1 → agent2)
- Scenarios (*identify relevant scenarios*)
- Agents
- Notes

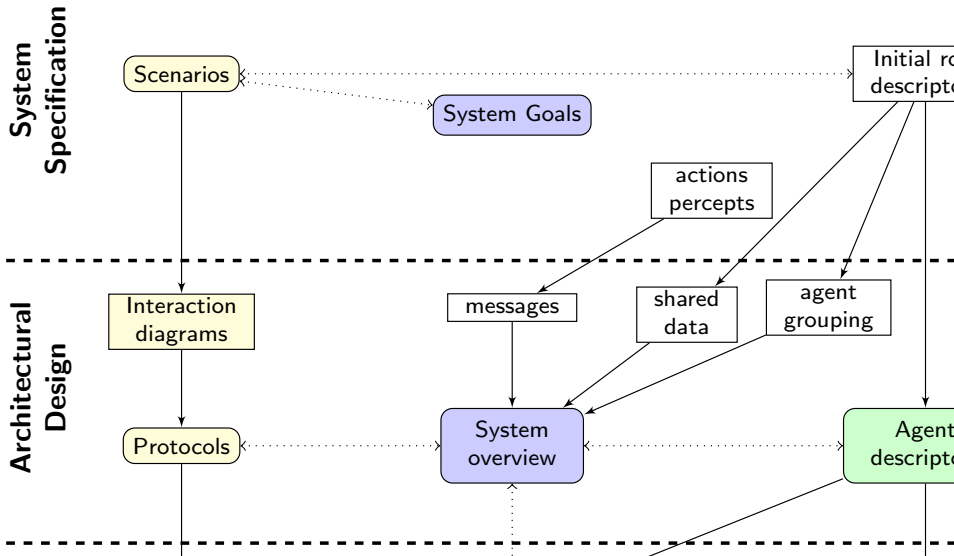
Specifying the Agent Interactions

- 1 Develop **interaction diagrams** from scenarios.
- 2 **Generalize** interaction diagrams to interaction protocols.
- 3 Develop **protocol**.
- 4 Develop **message descriptors**.

Prometheus Overview: 3 Phases

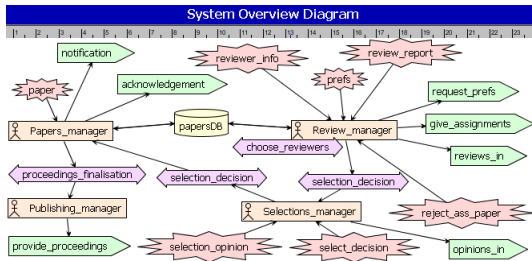


Prometheus Overview: 3 Phases



Protocols & System Overview Diagram

- Protocol links are determined from the protocol definitions.
- Protocols can be instantiated on this diagram and then edited.
- Protocol definitions can lead to creation of entities (watch for typos!!).



- Provides an overview of the system.
- The links to percepts and actions are propagated from the coupling of agents to roles for consistency.
- The interaction protocols are developed in this diagram.
- Only shared data is shown.

Review

Agent design phases:

- 1 **System specification:** interaction with the outside world.
 - identify roles, goals, scenarios, actors, actions and percepts.
- 2 **Architectural design:** agents and their internal interactions.
 - identify agents and their interactions.
- 3 **Detailed design:** details of every agent.
 - identify details of agents: capabilities, plans, events, data structures.

Agent interactions:

- 1 **Protocols:** specification of how agents are meant to communicate.
- 2 **Interaction diagrams:** mechanism to derive protocols (from scenarios).