Lecture 10: Practical BDI Programming II

Autonomous Agents and Multiagent Systems
DIS, La Sapienza - PhD Course

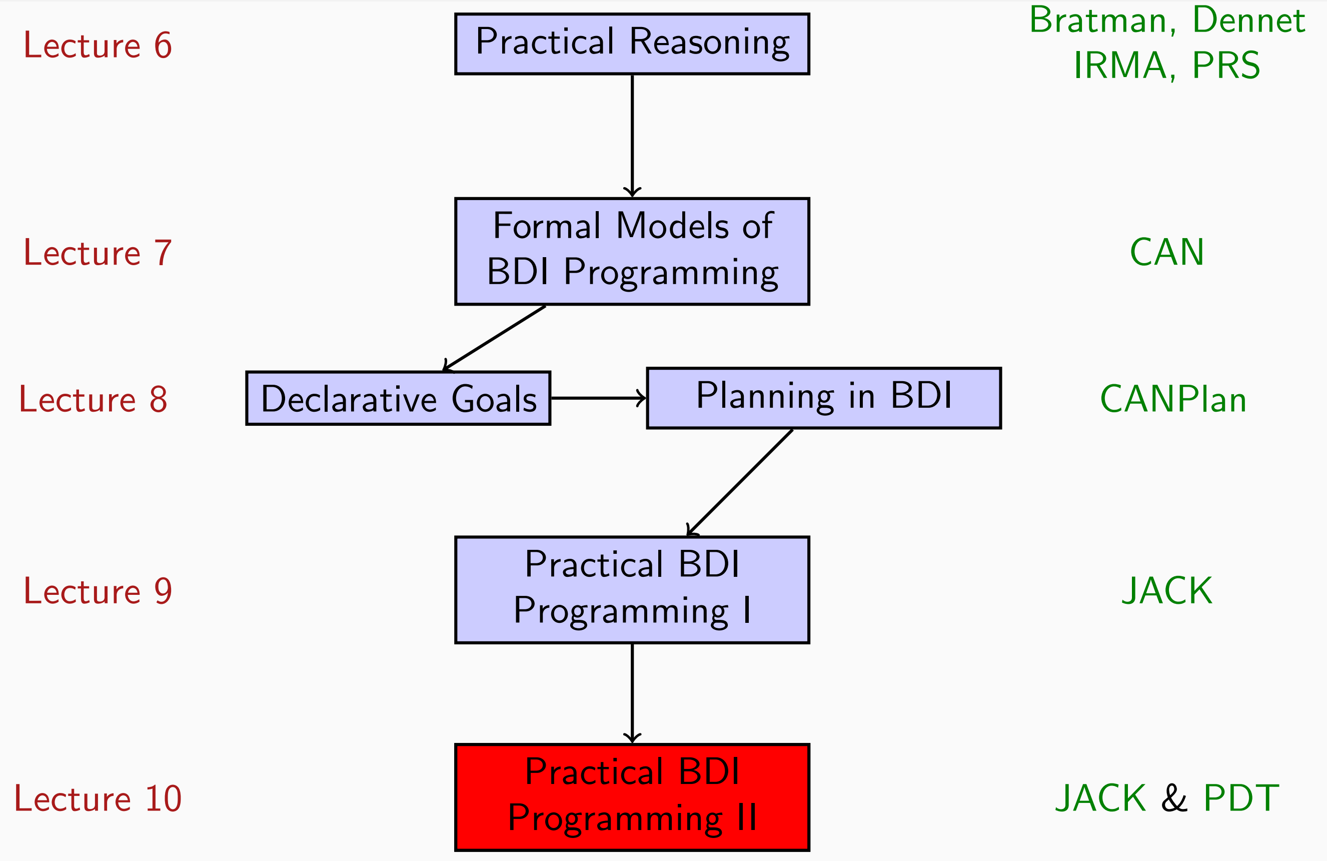
Sebastian Sardina[1]

[1]Department of Computer Science and Information Technology
RMIT University
Melbourne, AUSTRALIA

RMIT University

December 10, 2007

Roadmap for Next Lectures



Last Lecture Review

In the last lecture we reviewed the basics of BDI programming in JACK:

1. Reviewed the general agent-approach by extending Java:
   - The JACK Agent Language.
   - The JACK compiler;
   - The JACK kernel.
2. Analyzed the different structures of the agent language:
   - Agents;
   - Events;
   - Plans;
   - Beliefsets;
   - Capabilities.

This Lecture

PDT: Prometheus Design Tool

1. Captures the basic notions of BDI programming.
2. Is just an extension of Java.
3. Main components: agents, beliefsets, events, plans, capabilities, etc.
4. Special syntax style.

Outline

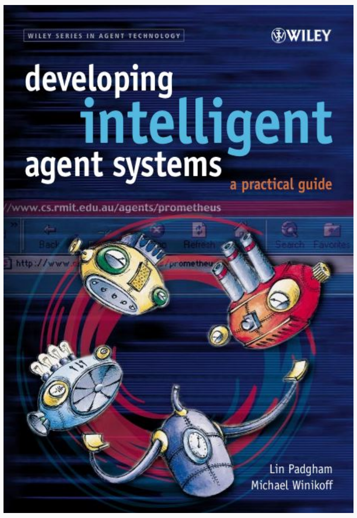
Agent-oriented Analysis and Design

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

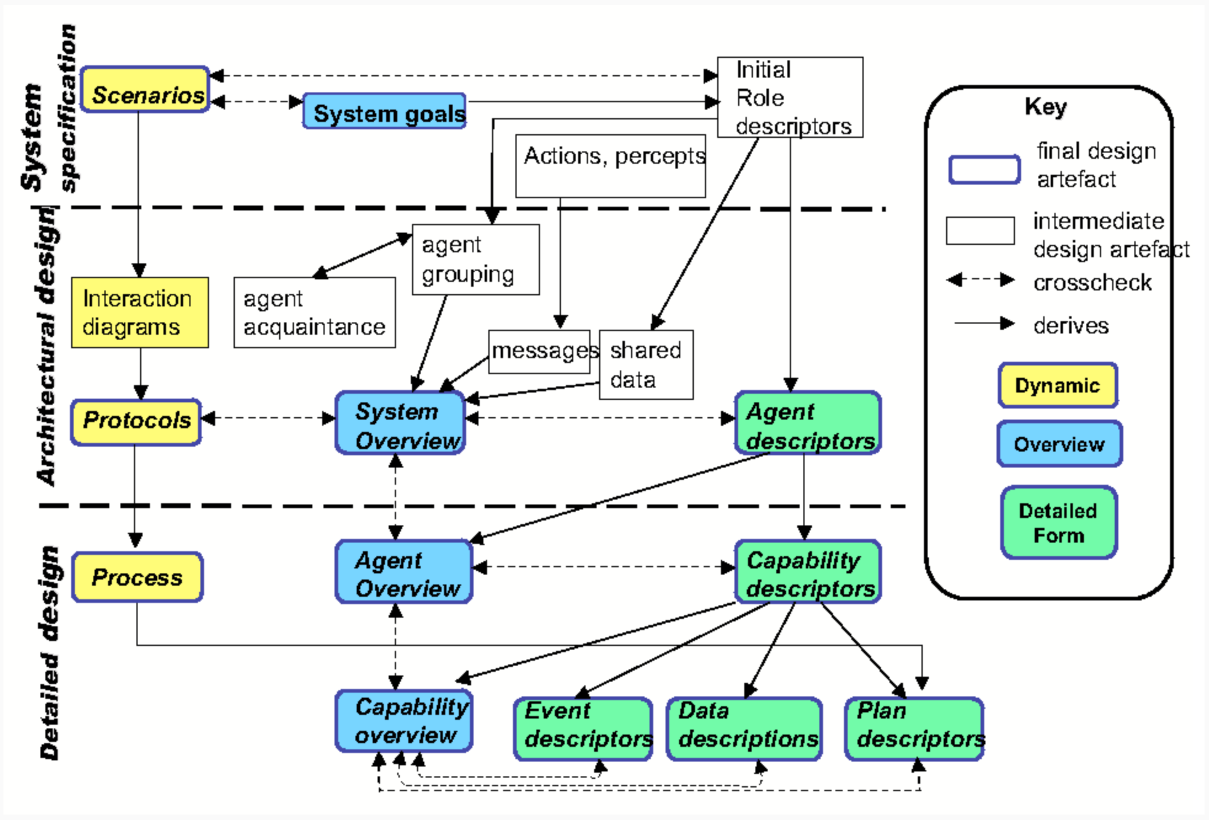
- A specialized methodology for agent systems – uses agent concept.
- Includes many standard aspects of SE: cohesion, coupling, modularity, etc.
- Has been developed as a result of experience in teaching and developing agent systems – both with students and with industry partners.
- Described in book "Developing Intelligent Agent Systems: A practical guide"


WILEY SERIES IN AGENT TECHNOLOGY
WILEY
developing intelligent agent systems
a practical guide
www.cs.rmit.edu.au/agents/prometheus
Lin Padgham
Michael Winikoff

Prometheus Design Tool (PDT)

1. Nearly impossible to design and develop largish systems without tool support for methodology.
2. PDT a substantial project within RMIT Agents group
3. Downloadable from the webpage `www.cs.rmit.edu.au/agents/pdt/`
4. Under development - new versions coming

Prometheus Design Tool (PDT)



Prometheus Design Tool (PDT)

- 3 phases:
  1. System specification.
  2. Architectural design.
  3. Detailed design. (really has 2 sub-phases.)
- 3 “columns” in Prometheus grid:
  1. System structure (middle column).
  2. System dynamics (left column).
  3. System details (right column – descriptors).
- Implementation, testing, debugging, maintenance: all part of current work.
- Suitable for, but not limited to BDI agents.
- Prometheus style diagrams can be drawn in JDE for detailed design.

System Specification

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.

Actor Scenario Diagram

1. The specific use of Analysis Overview diagram within Prometheus.
2. Focus on “Actors” and “Scenarios”.
3. Actors are people or SW interacting with the system to be built.
4. Scenarios are the top level interactions of the Actors with the system.
5. Also helps identify percepts and actions.
6. Will illustrate by example.

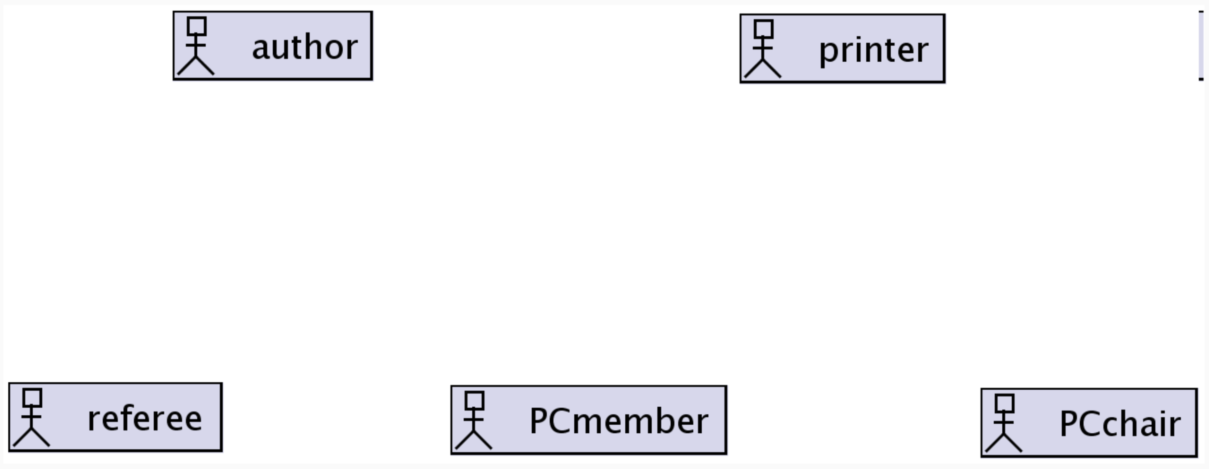
Conference Management Example

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

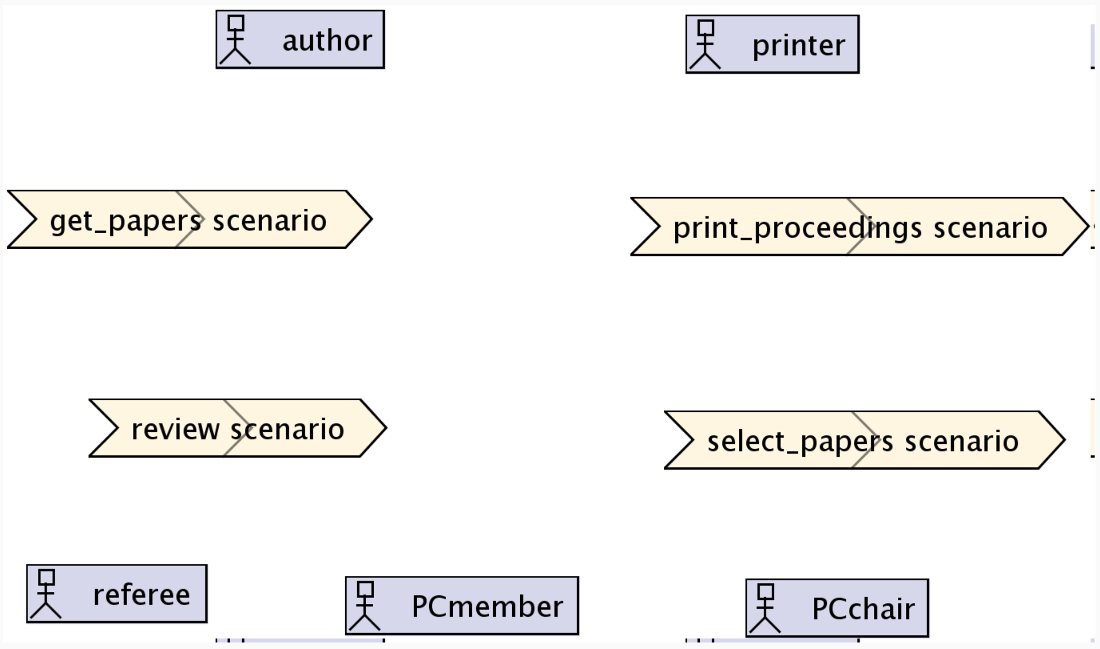
Description

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.

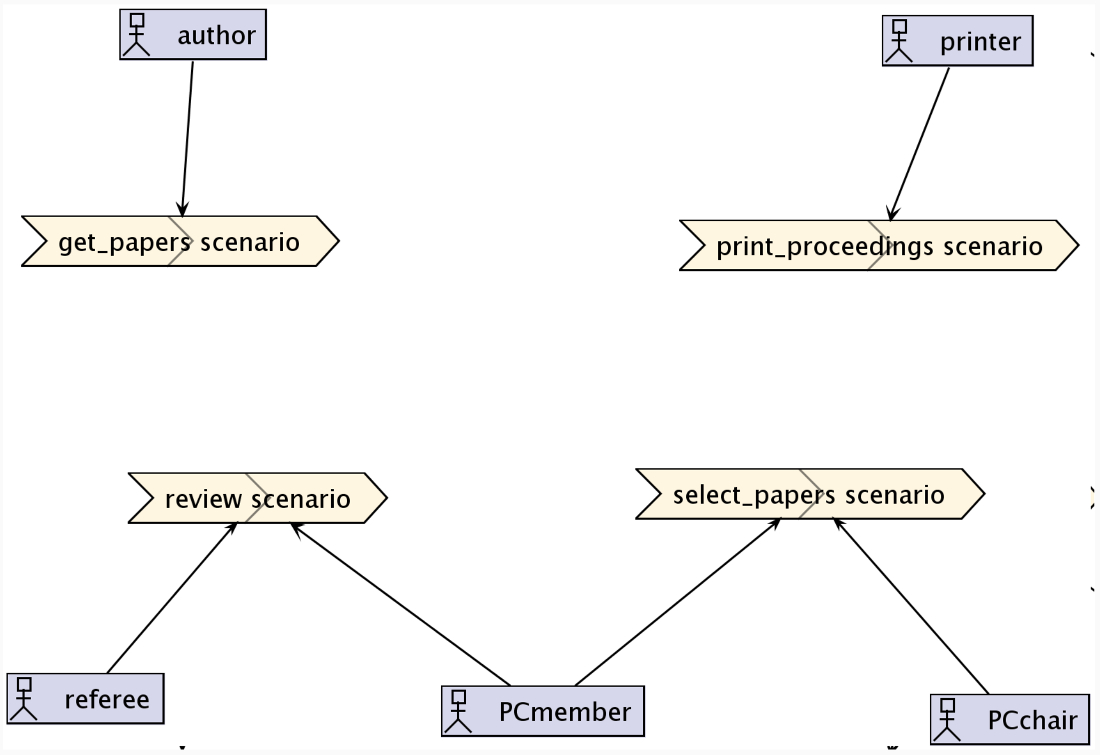
Identify the Actors



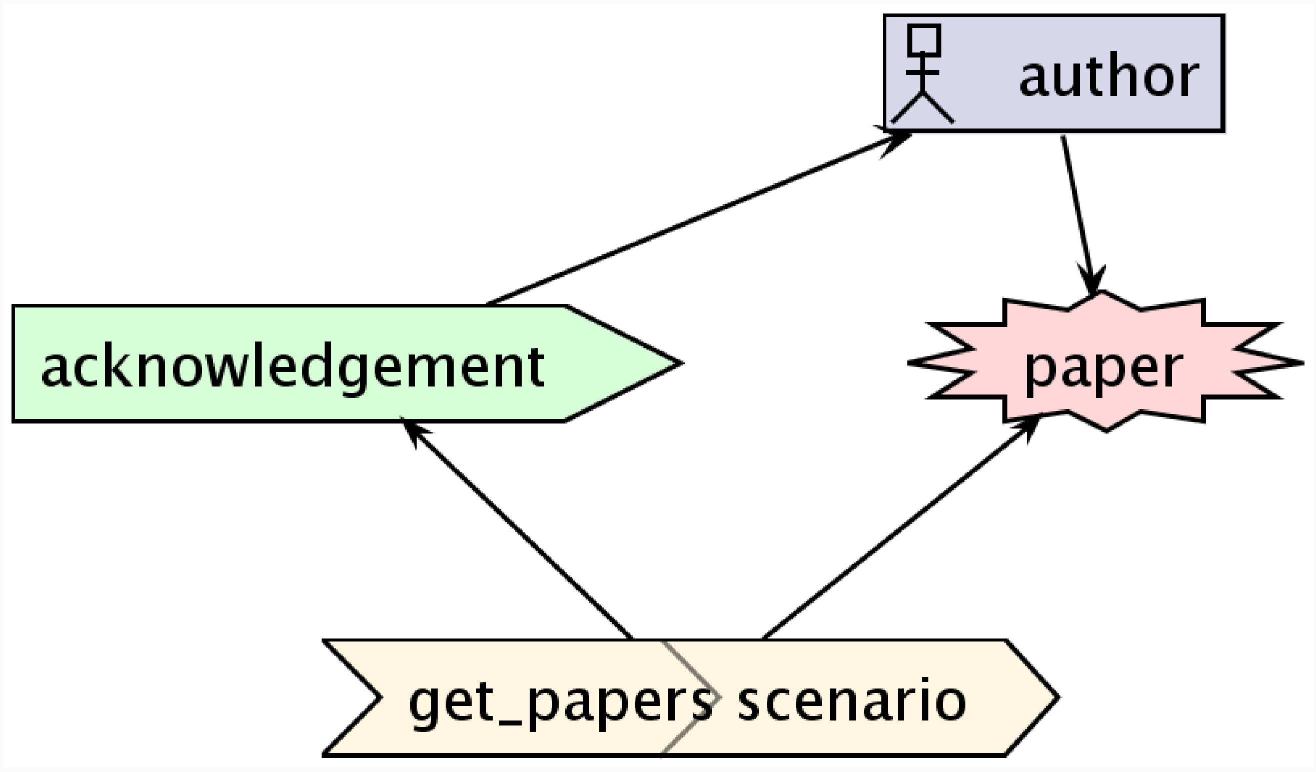
Identify the Scenarios



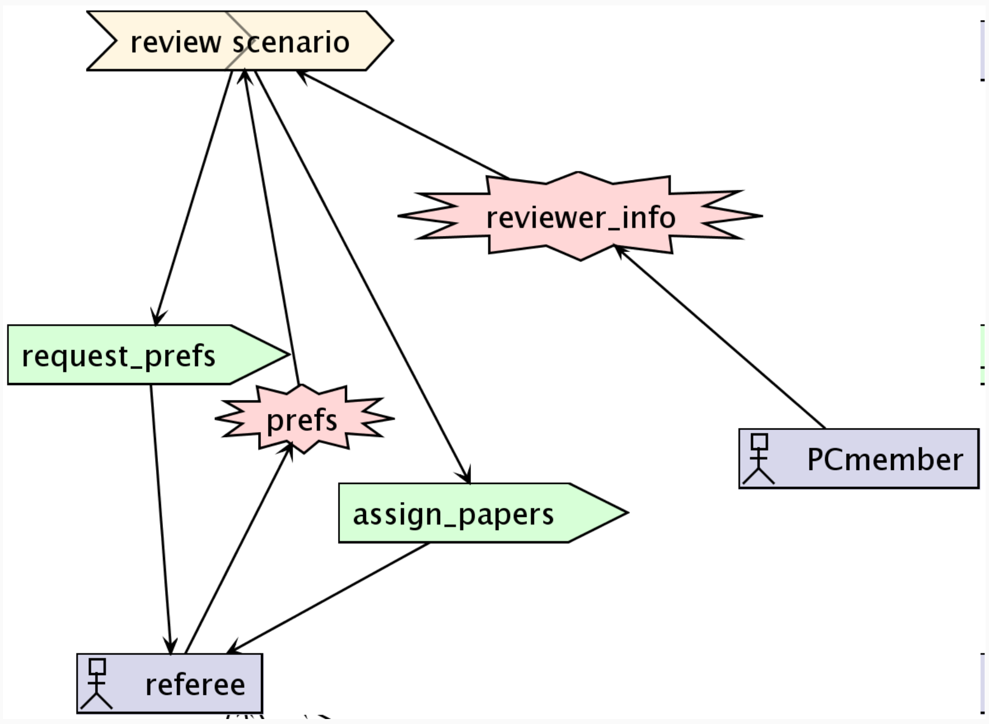
Link Actors and Scenarios



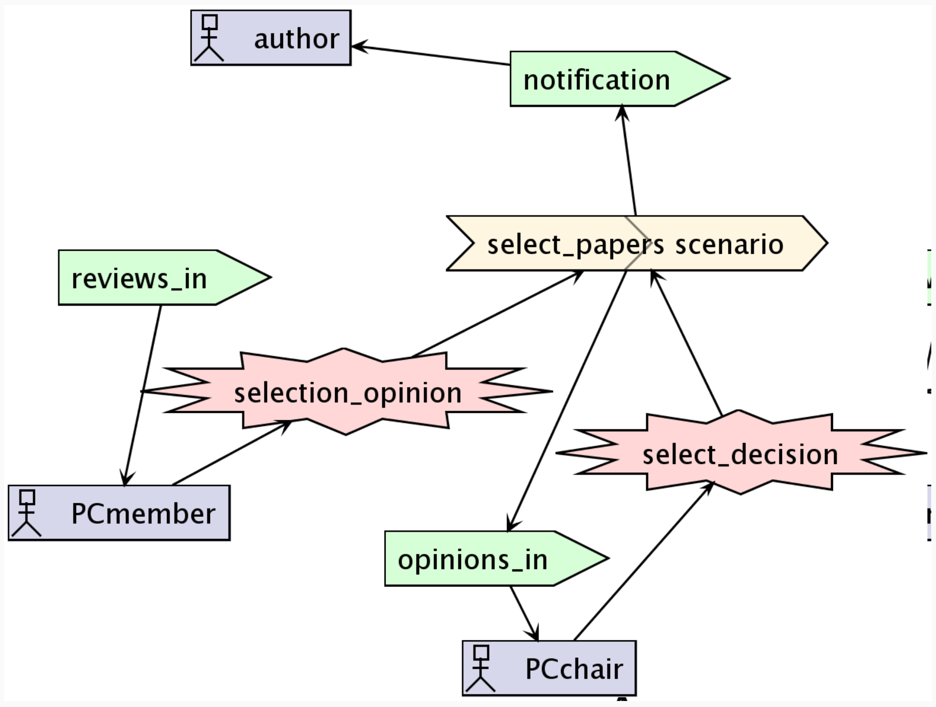
Add Actions and Percepts: Submission



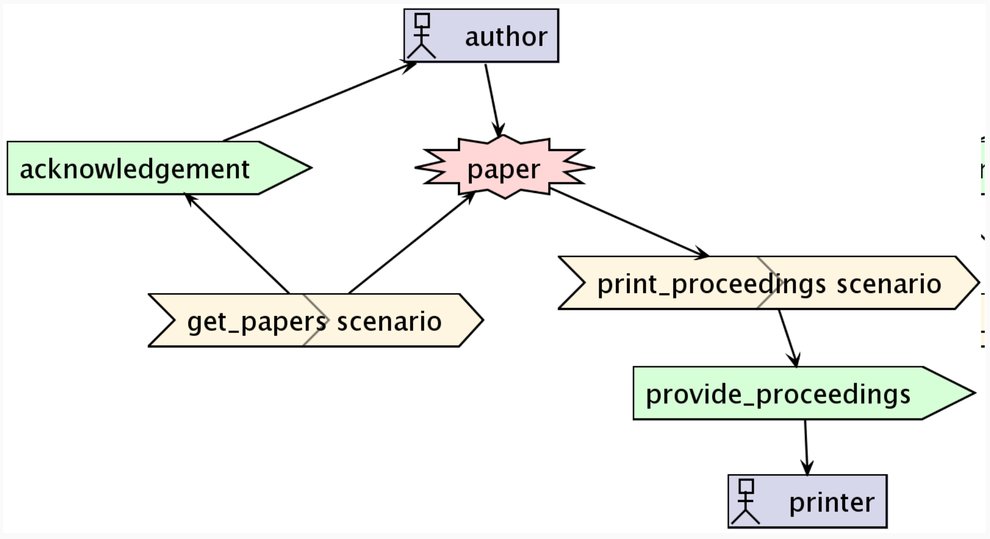
The "Review" Scenario



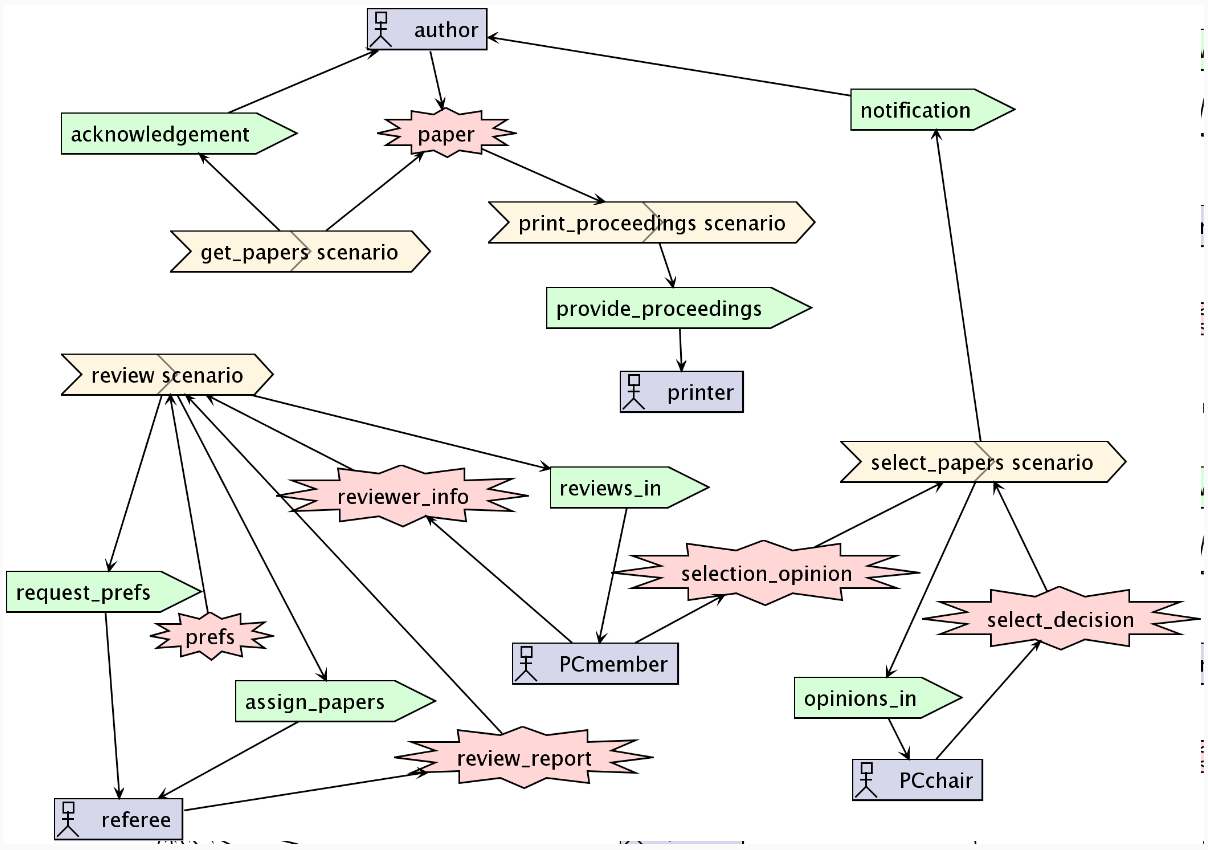
The "Selection" Scenario



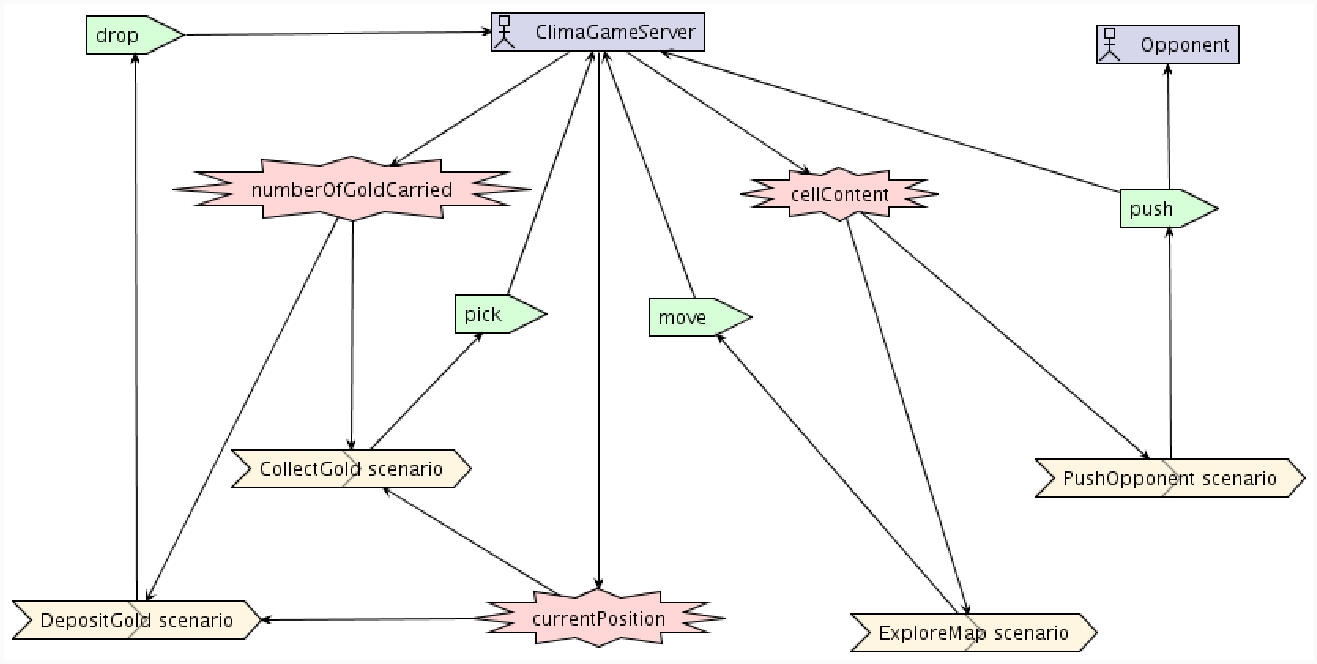
The "Print Proceedings" Scenario



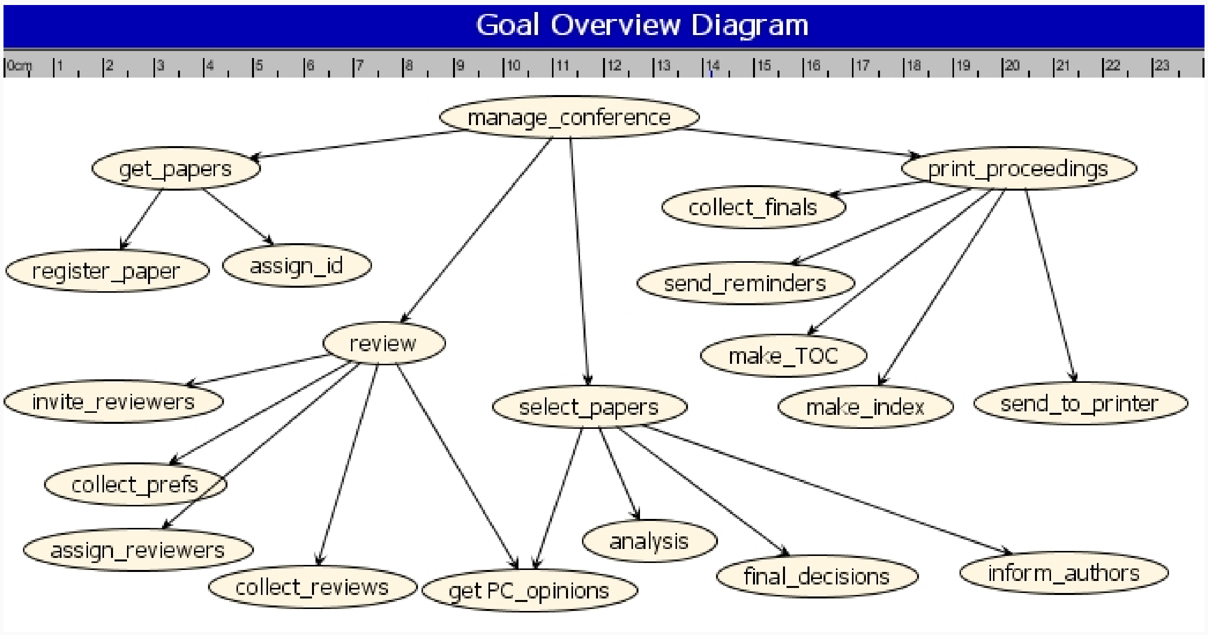
Completed Analysis Overview



Completed Analysis Overview (CLIMA)

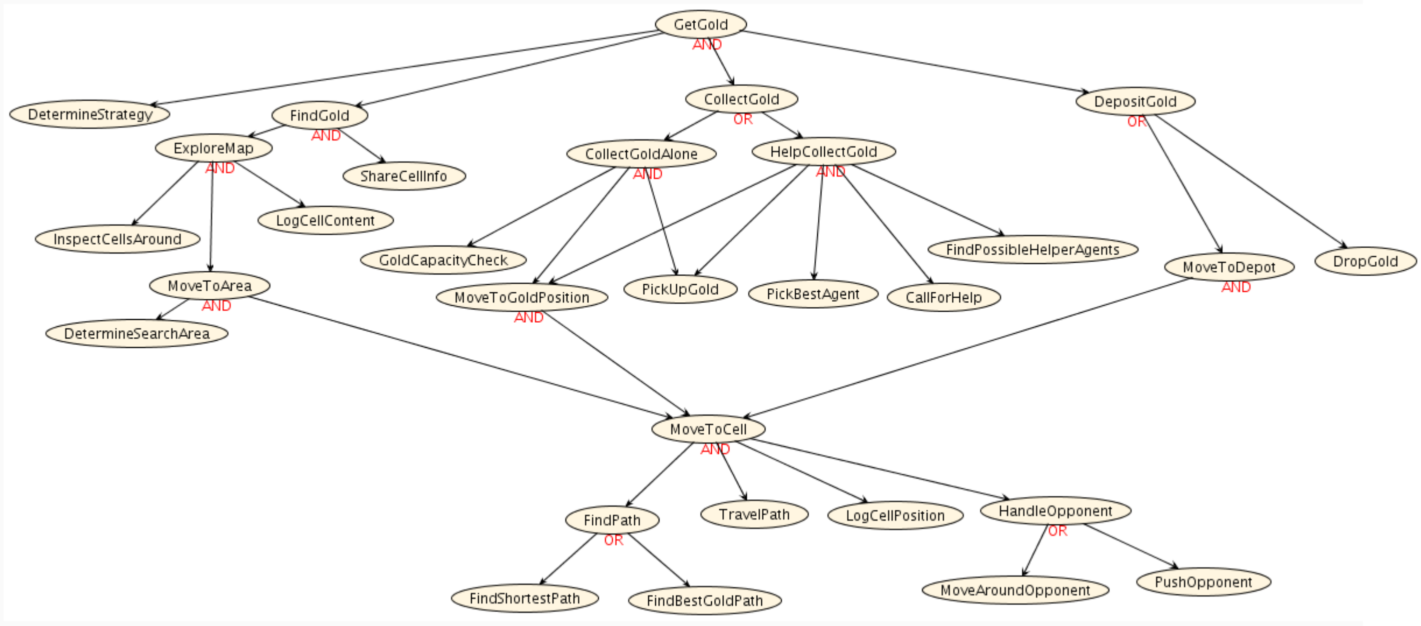


Goal Overview Diagram

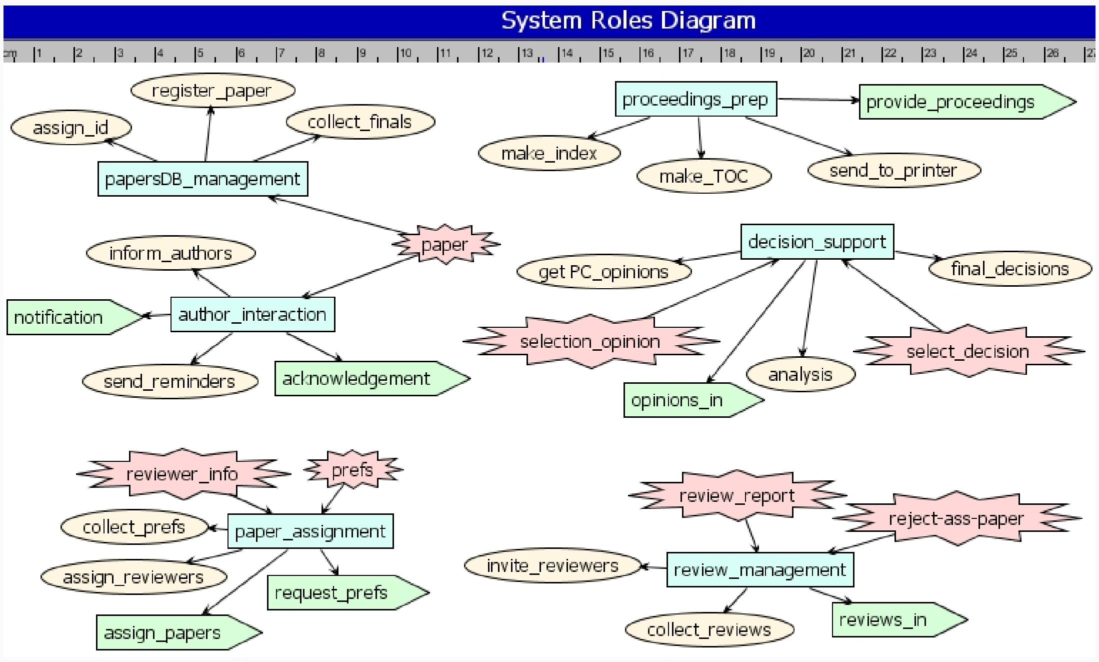


- Coupled with the Scenarios Diagram; identifying the goals of the system.
- For each goal, the sub-goals are identified by informally asking "how can we achieve this goal?"
- Sub-goals are either "AND" or "OR" branches which is specified in the descriptor of the top level goal. (default is "AND")

Goal Overview Diagram (CLIMA)



System Role Diagrams



- Goals are then grouped into roles
- The appropriate actions and percepts are also attached. They are identified from the scenarios & the analysis overview diagram

Architectural Design

Uses the artifacts produced in the system specification phase to identify the agents within the system and the interactions between these agent.

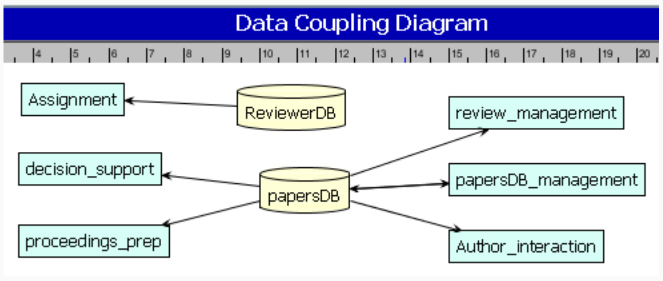
Data Coupling Couple the data with the roles that access them.

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

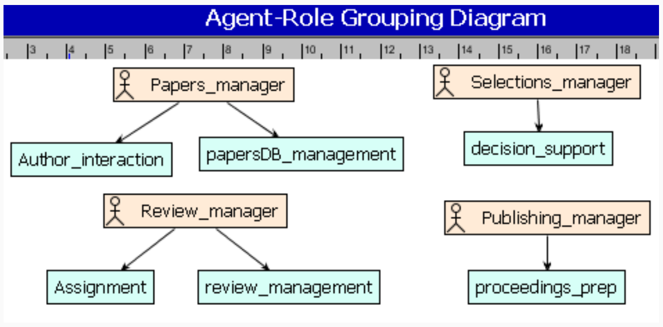
Agent Acquaintance Considers the initialization aspects and cardinality of the agents. This aspect is currently not supported by PDT.

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

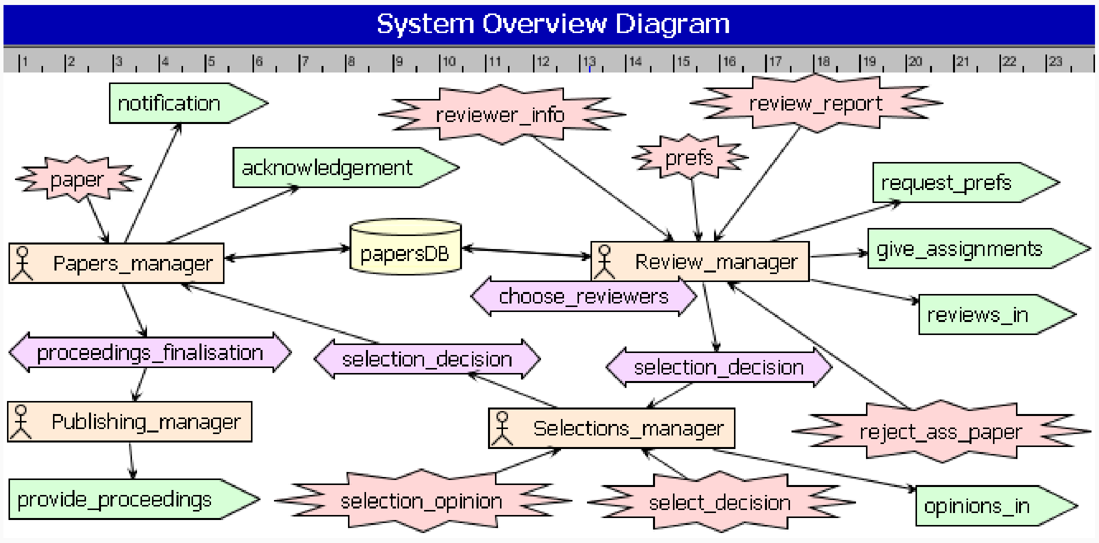


- Data are coupled with Roles that read and write from them.



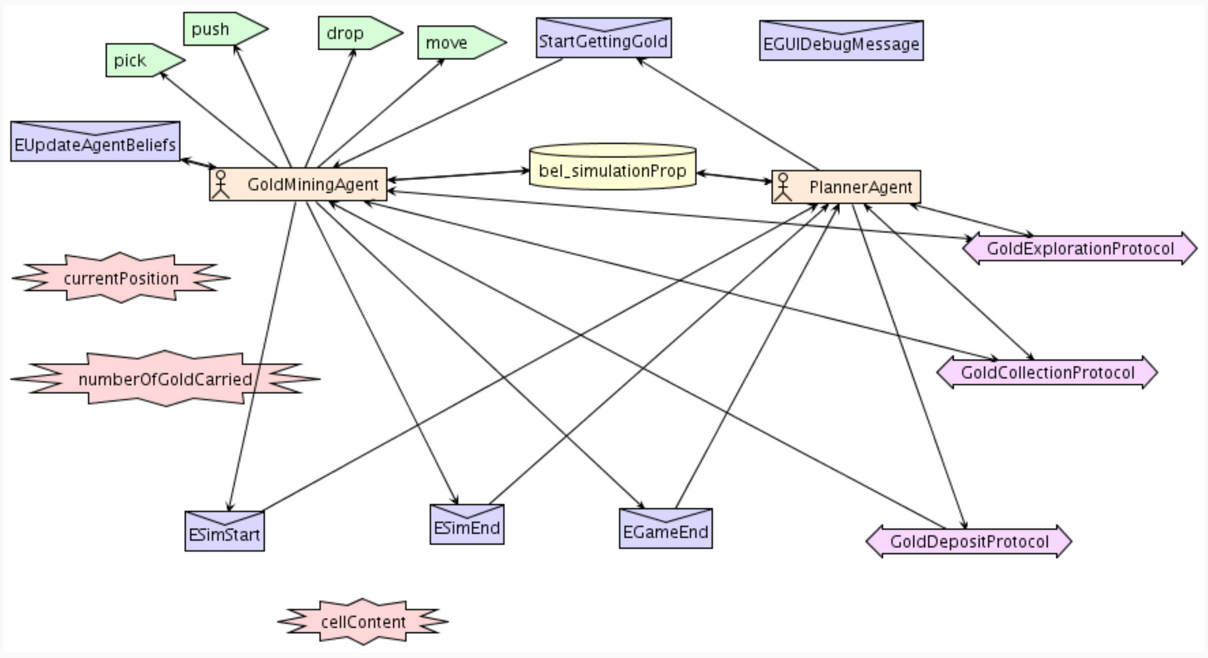
- Roles are assigned to Agents.

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.

System Analysis Overview (CLIMA)



Detailed Design

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

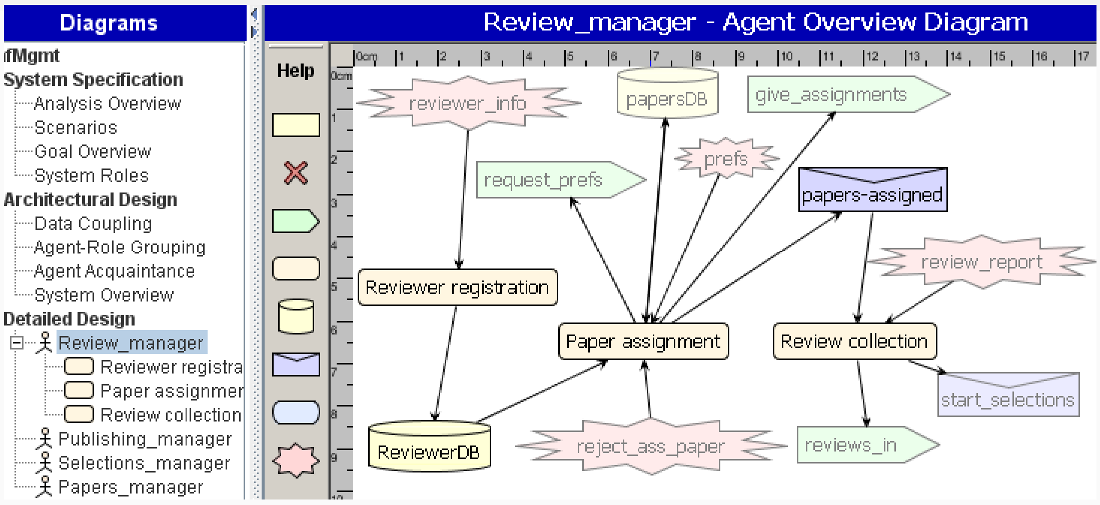
Each agent is refined in terms of capabilities, internal events, plans and internal data structures.

A capability (which is basically a collection of plans) allows the agent to be modularized.

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

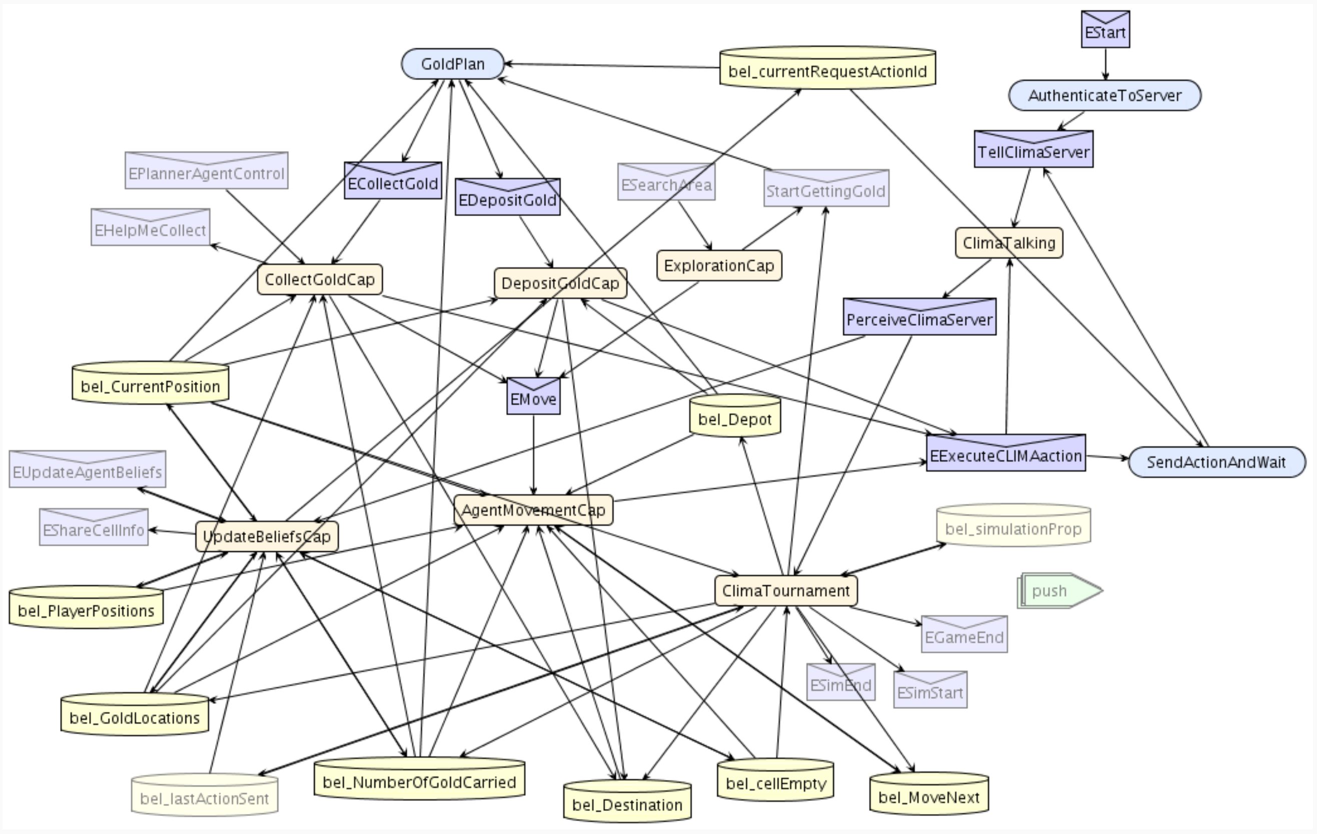
Capability Overview Describes the plans and the events that are handled by the plans

Agent Overview Diagram

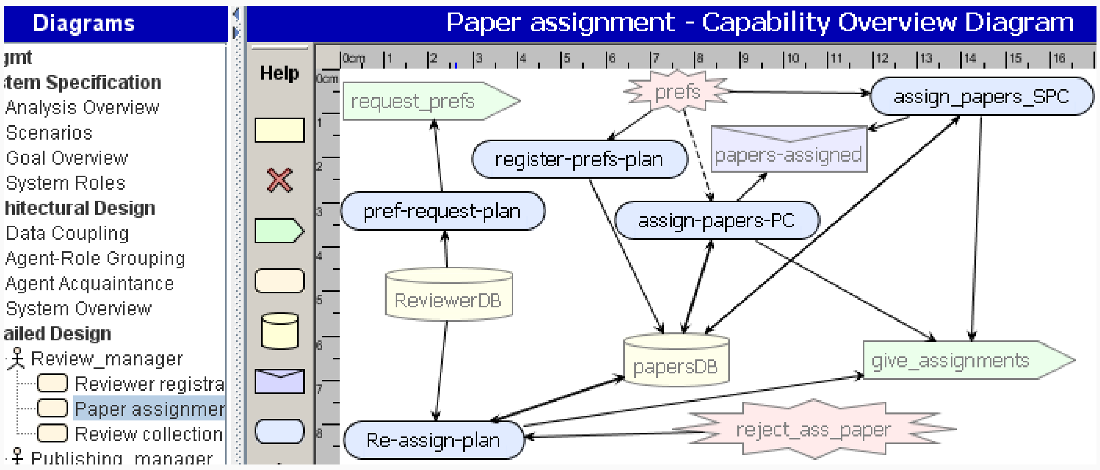


- Provides an overview of the capabilities of the agent.
- Percepts and actions are propagated from the architectural design stage.
- Note that percepts are inputs and from external entities (actors) and actions are output to external entities.
- Entities that are faded are those that were automatically propagated. The rest are internal to the agent.

Agent Overview Diagram (CLIMA)

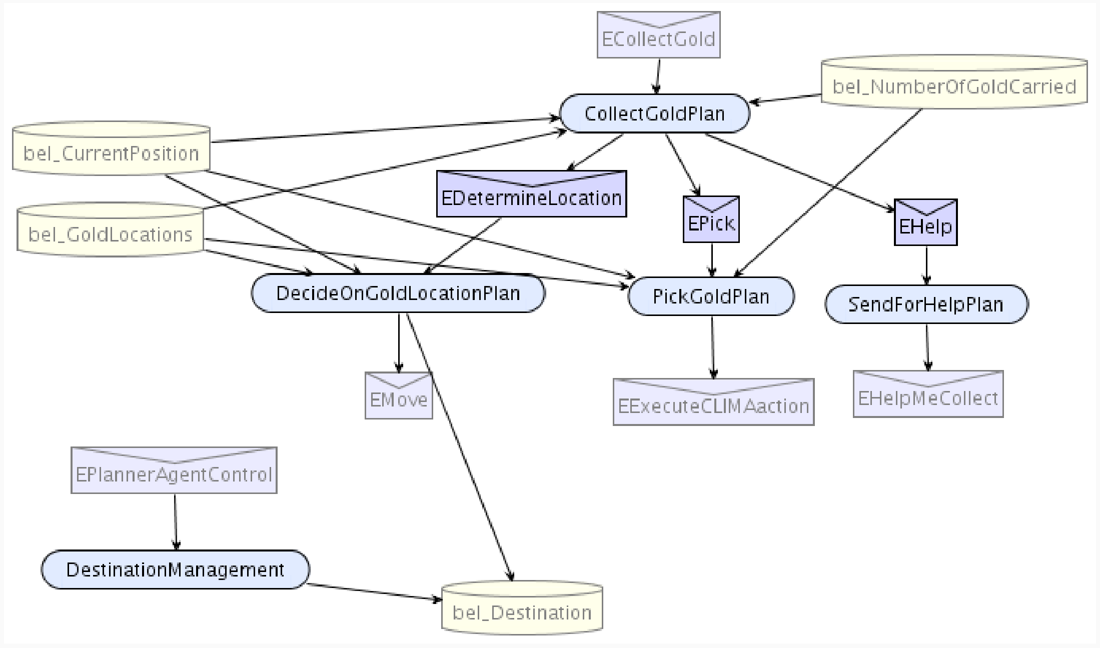


Capabilities Overview



- 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 (CLIMA)



PDT Tools

1. Crosscheck allows for on-demand consistency checking.
2. All the diagrams can be saved as images in PNG format.
3. The tool allows for a report to be generated in HTML Format.
4. Create backup allows the user to save the current version of the project into a different filename and continue working on the existing file.
5. Code generation. Creates skeleton code into the JACK agent language.

Consistency Checking

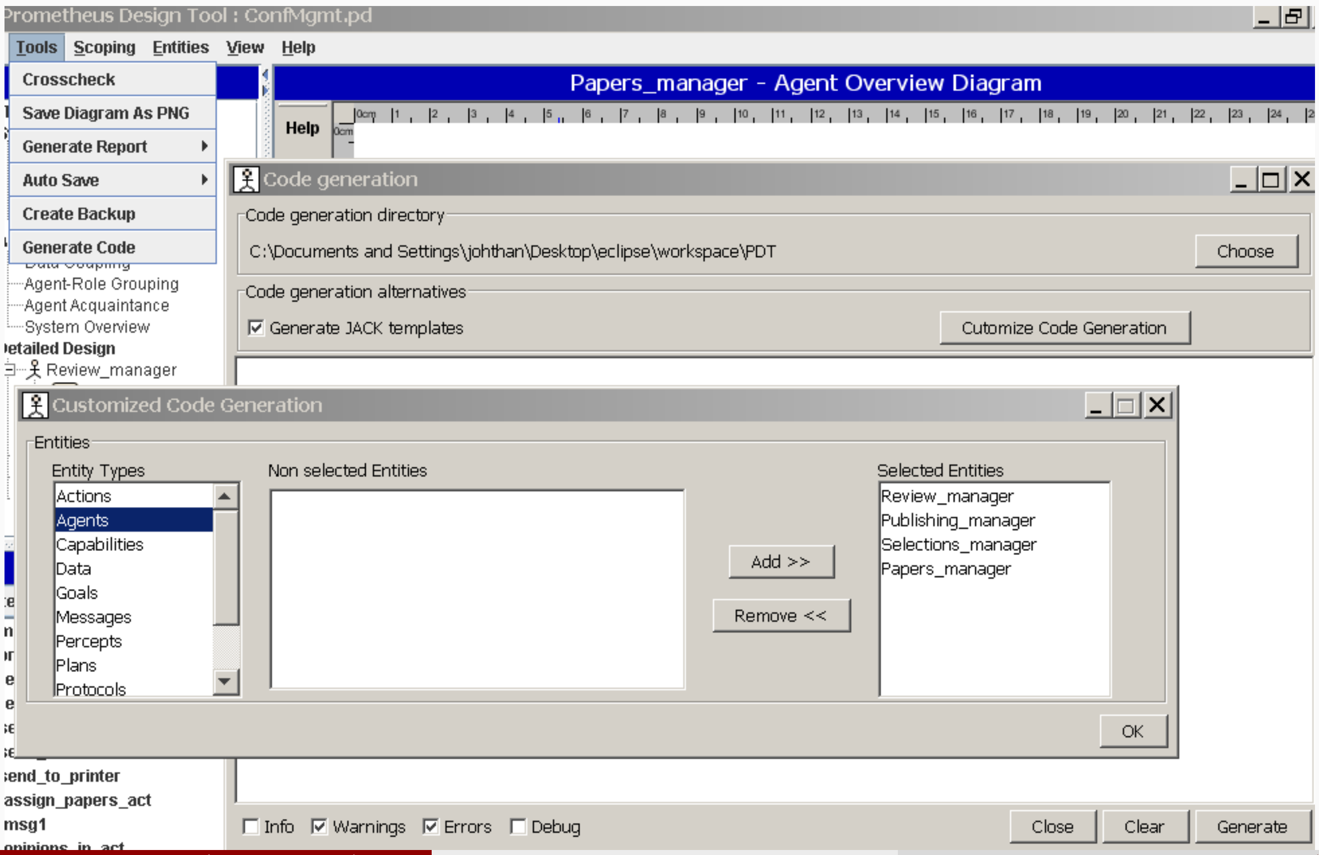
The tool prevents certain errors from being made, such as:

1. Creating references to non-existent entities.
2. Having two entities with the same name.
3. "Type errors" such as connecting two actions.

The tool also prevents certain inconsistencies between levels. For instance, if the system overview diagram specifies that agent A reads data B and receives percept P, then the agent overview diagram for the agent will automatically contain the percept P and data B, furthermore the tool will prevent the user from creating a write arrow to B.

Additionally, the tool offers an "on-demand" consistency check that generates a list of errors and warnings that can be checked by the developer. Examples of a warning are writing of internal data that is never read, while an example of an error is a mismatch between the interaction protocol specified between two agents and the messages actually sent and received by processes within those agents.

Code Generation to JACK



Code generation

Code generation directory
C:\Documents and Settings\johthan\Desktop\eclipse\workspace\PDT
Choose

Code generation alternatives
☑ Generate JACK templates
Cutomize Code Generation

Customized Code Generation

Entities

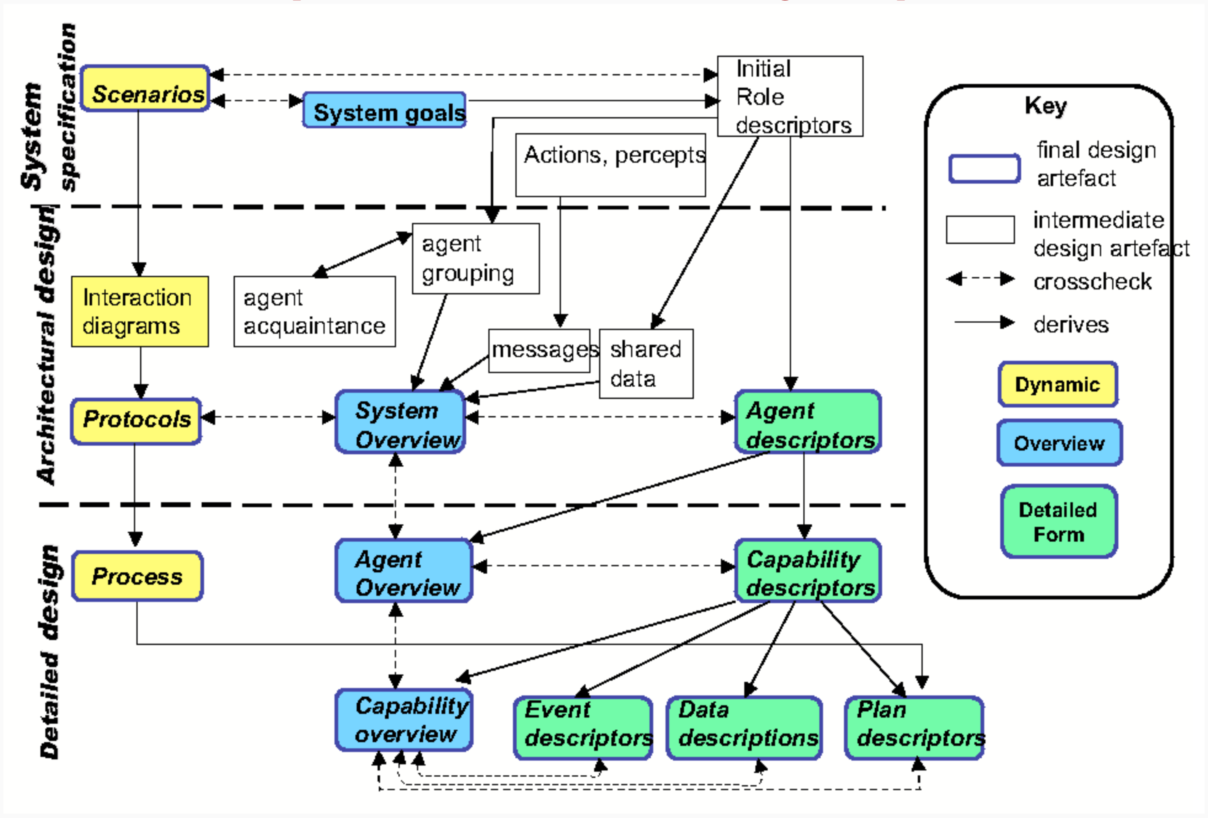
Entity Types	Non selected Entities	Selected Entities
Actions		Review_manager
Agents		Publishing_manager
Capabilities		Selections_manager
Data		Papers_manager
Goals		
Messages		
Percepts		
Plans		
Protocols		

Add >>
Remove <<
OK

☐ Info ☑ Warnings ☑ Errors ☐ Debug
Close Clear Generate

Review

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



Agent Methodologies http://www.science.unitn.it/~recla/aose/

Lin Padgham and Michael Winikoff.
Developing Intelligent Agent Systems: A Practical Guide.
John Wiley and Sons, June 2004.

Lin Padgham and Michael Winikoff.
Prometheus: A Pragmatic Methodology for Engineering Int. Agents
In *Proceedings of the workshop on Agent-oriented methodologies*, 2002.

C. A. Iglesias, M. Garijo and J. C. Gonzlez.
A Survey of Agent-Oriented Methodologies.
In *Proceedings of ATAL-98*, LNAI, 1999.

N. R. Jennings.
On agent-based software engineering
Artificial Intelligence, March, 2000, vol. 117, no. 2, p. 277-96.

A. Tveit.
A survey of Agent-Oriented Software Engineering.
In *Proc of NTNU Computer Science Graduate Student Conference.*
Norwegian University of Science and Technology, May (2001).