Tool Support for Prometheus

- Jack Development Environment: Design Tool
  Developed by Agent Oriented Software

- Prometheus Design Tool (PDT)
  - Developed first half of 2003 by Anna Edberg & Christian Andersson at RMIT
  - Later extended by Claire Hennekam and Jason Khallouf.

- AUML tool
  - Prototype by Michael Winikoff
Prometheus Design Tool

- Design tool supporting the Prometheus methodology developed at RMIT

- Features:
  - Enter and edit designs (descriptor forms, diagrams)
  - Cross checking:
    - undefined and unused entities
    - interface consistency
    - …
  - Generate design report (HTML) + images (PNG)

- Written in Java, freely available on web
  www.cs.rmit.edu.au/agents/pdt

- Beta – feedback welcome!

- Demo …
Scenarios

System goals

Initial Functionality descriptors

Actions, percepts

Interaction diagrams

data coupling

agent acquaintance

messages

System Overview

Agent descriptors

Capability descriptors

Capability overview

Plan descriptors

Data descriptions

Event descriptors

System specification

Architectural design

Detailed design

Protocols

Process

Support for Prometheus
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
finish

FIPA query protocol

Initiator

Participant

alt

query-if

query-ref

alt

not-understood

refuse

failure

inform

www.cs.rmit.edu.au/~winikofff/auml
Additional Slides

(including screenshots of PDT, for those reading the slides after the demo/presentation)
Goal Overview Diagram

- Schedule meeting
- Minimise clashes
- Include essential members
- Cancel meeting
- Permeate meeting
- Reschedule meeting
- Form ideal schedule
- Avoid cascading reschedules
- Consider user availability
- User interaction
- Consider meeting priority
- Update unavailable times
- Update available hours
- Notify user of meetings
GUI – Descriptor

Name: GUI
Description: Interacts with the user and stores user-related data.
Cardinality minimum: 1
Cardinality maximum: 1
Lifetime: System.
Initialisation:

Demise:
<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Meeting accepted</td>
</tr>
<tr>
<td>Action</td>
<td>Meeting cancelled</td>
</tr>
<tr>
<td>Action</td>
<td>Meeting declined</td>
</tr>
<tr>
<td>Action</td>
<td>Request User Intervention</td>
</tr>
<tr>
<td>Action</td>
<td>InformUser</td>
</tr>
<tr>
<td>Agent</td>
<td>CUI</td>
</tr>
<tr>
<td>Agent</td>
<td>Messaging</td>
</tr>
<tr>
<td>Agent</td>
<td>Meeting Scheduler</td>
</tr>
<tr>
<td>Data</td>
<td>Meetings</td>
</tr>
<tr>
<td>Data</td>
<td>Available hours</td>
</tr>
<tr>
<td>Data</td>
<td>Unavailable times</td>
</tr>
<tr>
<td>Capability</td>
<td>manageMeetings</td>
</tr>
<tr>
<td>Capability</td>
<td>manageAvailability</td>
</tr>
<tr>
<td>Functionality</td>
<td>Meeting Planner</td>
</tr>
<tr>
<td>Functionality</td>
<td>Interaction Manager</td>
</tr>
<tr>
<td>Functionality</td>
<td>Rescheduler</td>
</tr>
<tr>
<td>Functionality</td>
<td>Personal Planner</td>
</tr>
<tr>
<td>Message</td>
<td>proposeMeeting</td>
</tr>
<tr>
<td>Message</td>
<td>proposedMeetingOK</td>
</tr>
<tr>
<td>Message</td>
<td>checkPriority</td>
</tr>
<tr>
<td>Plan</td>
<td>addMeeting1</td>
</tr>
<tr>
<td>Plan</td>
<td>addMeeting2</td>
</tr>
<tr>
<td>Message</td>
<td>AvailabilityQuery</td>
</tr>
<tr>
<td>Message</td>
<td>AvailabilityResponse</td>
</tr>
<tr>
<td>Data</td>
<td>Available hours set</td>
</tr>
<tr>
<td>Goal</td>
<td>Avoid cascading reschedules</td>
</tr>
<tr>
<td>Goal</td>
<td>Cancel meet</td>
</tr>
<tr>
<td>Goal</td>
<td>Cancel meeting</td>
</tr>
<tr>
<td>Plan</td>
<td>cancelMeeting</td>
</tr>
<tr>
<td>Plan</td>
<td>CancelMeetingplan</td>
</tr>
<tr>
<td>Message</td>
<td>Check meetings</td>
</tr>
<tr>
<td>Message</td>
<td>Check user availability</td>
</tr>
<tr>
<td>Message</td>
<td>checkAttendance</td>
</tr>
<tr>
<td>Message</td>
<td>checkPriority</td>
</tr>
<tr>
<td>Message</td>
<td>checkPriorityplan</td>
</tr>
<tr>
<td>Plan</td>
<td>Clash</td>
</tr>
<tr>
<td>Plan</td>
<td>Clash occurs with high priority</td>
</tr>
<tr>
<td>Plan</td>
<td>Clash occurs with low priority</td>
</tr>
<tr>
<td>Goal</td>
<td>Consider meeting priority</td>
</tr>
</tbody>
</table>
### Consistency checking

**Consistency check alternatives**

- [x] Check project agents
- [x] Check project capabilities

**Find unused:**

- [ ] Actions
- [ ] Capabilities
- [ ] Functionalities
- [ ] Messages
- [ ] Plans
- [ ] Scenarios
- [ ] Agents
- [ ] Data
- [ ] Goals
- [ ] Percepts
- [ ] Protocols

---

OK: The message "checkPriority" is sent.
OK: The message "priorityResponse" is received.
OK: The message "priorityResponse" is sent.
OK: The message "scheduleMeeting" is received.
OK: The message "scheduleMeeting" is sent.
OK: The message "notifyCancelMeeting" is received.
OK: The message "notifyCancelMeeting" is sent.
OK: The message "notifyNewMeeting" is received.
OK: The message "notifyNewMeeting" is sent.
OK: The message "AvailabilityQuery" is received.
OK: The message "AvailabilityQuery" is sent.
OK: The message "AvailabilityResponse" is received.
OK: The message "AvailabilityResponse" is sent.
OK: The message "Request Manual Scheduling" is received.
OK: The message "Request Manual Scheduling" is sent.
OK: The message "cancelMeeting" is received.

---

[Check] [Close] [Clear]
Goals
Reschedule meeting, Avoid cascading reschedules, Consider meeting priority

Functionality Personal Planner
Name: Personal Planner
Description: Concerned with the user interface. Obtains from the user when they are not available, reports to the user their meetings.
Triggers: Unavailable times set, Available hours set, User requests meetings

System Overview Diagram

- Meetings
- Available hours
- Unavailable times
- Request User Intervention
- notifyCancelMeeting
- notifyNewMeeting
- proposedMeeting
- User requests meetings
- New meeting
- Check Availability
- Check priority
- Request Manual Scheduling
- Messaging
- Meeting proposed
- Meeting declined
- Meeting cancelled
- Meeting accepted

notifyCancelMeeting
notifyNewMeeting
proposedMeeting
Check Availability
Check priority
Request Manual Scheduling
Messaging
Meeting proposed
Meeting declined
Meeting cancelled
Meeting accepted
### Scenario New meeting received

<table>
<thead>
<tr>
<th>Name</th>
<th>New meeting received</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The agent has received a meeting from another user's agent.</td>
</tr>
</tbody>
</table>

#### Steps

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Name</th>
<th>Functionality</th>
<th>Description</th>
<th>Data produced</th>
<th>Data used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process</td>
<td>Meeting proposed</td>
<td>Interaction Manager</td>
<td>New meeting has been received from another user's agent.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Goal</td>
<td>Consider user availability</td>
<td>Interaction Manager</td>
<td>Check whether user can make the meeting with their current calendar.</td>
<td>Meetings, Available hour</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Action</td>
<td>Meeting accepted</td>
<td>Interaction Manager</td>
<td>New meeting is accepted, reply is sent back to the agent.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Goal</td>
<td>Notify user of meeting</td>
<td>Personal Planner</td>
<td>Notify user of new meeting.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Variation

If the user is unavailable for some reason, this is considered a clash and is handled as per the 'Clash occurs' scenario. If the meeting has been rescheduled too many times, it is flagged for manual rescheduling according to the 'Meeting cannot be rescheduled' scenario.

### Scenario Modify unavailable times

<table>
<thead>
<tr>
<th>Name</th>
<th>Modify unavailable times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>The user modifies their rates for unavailable times, either modifying or deleting existing rules, or adding a new rule.</td>
</tr>
</tbody>
</table>

#### Steps

<table>
<thead>
<tr>
<th>#</th>
<th>Type</th>
<th>Name</th>
<th>Functionality</th>
<th>Description</th>
<th>Data produced</th>
<th>Data used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process</td>
<td>Unavailable times set</td>
<td>Personal Planner</td>
<td>User submits new unavailable time or modifies existing rules.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Goal</td>
<td>Update unavailable times</td>
<td>Personal Planner</td>
<td>Update Unavailable times XML with new details.</td>
<td></td>
<td>Unavailable times</td>
</tr>
</tbody>
</table>

#### Variation

If a rule to be modified or deleted does not exist, an appropriate error is returned to the user.
Consistency checking

Consistency check alternatives

☑ Check project agents
☑ Check project capabilities

Find unused:

☑ Actions ☑ Capabilities ☑ Functionalities ☑ Messages ☑ Plans ☑ Scenarios
☑ Agents ☑ Data ☑ Goals ☑ Percepts ☑ Protocols

OK: The message "checkPriority" is sent.
OK: The message "priorityResponse" is received.
OK: The message "priorityResponse" is sent.
OK: The message "scheduleMeeting" is received.
OK: The message "scheduleMeeting" is sent.
OK: The message "notifyCancelMeeting" is received.
OK: The message "notifyCancelMeeting" is sent.
OK: The message "notifyNewMeeting" is received.
OK: The message "notifyNewMeeting" is sent.
OK: The message "AvailabilityQuery" is received.
OK: The message "AvailabilityQuery" is sent.
OK: The message "AvailabilityResponse" is received.
OK: The message "AvailabilityResponse" is sent.
OK: The message "Request Manual Scheduling" is received.
OK: The message "Request Manual Scheduling" is sent.
OK: The message "cancelMeeting" is received.