

# Seminar AI Tools

## **JADE und FIPA**

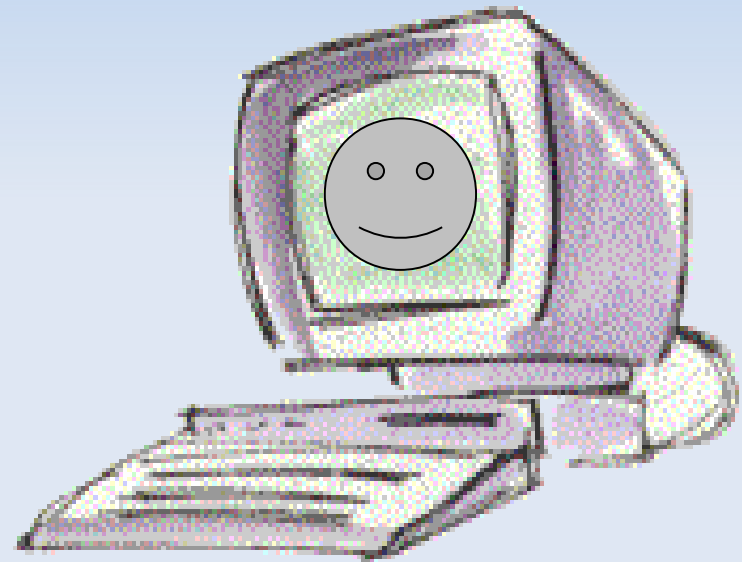
Cathrin Weiß

09. November 2006

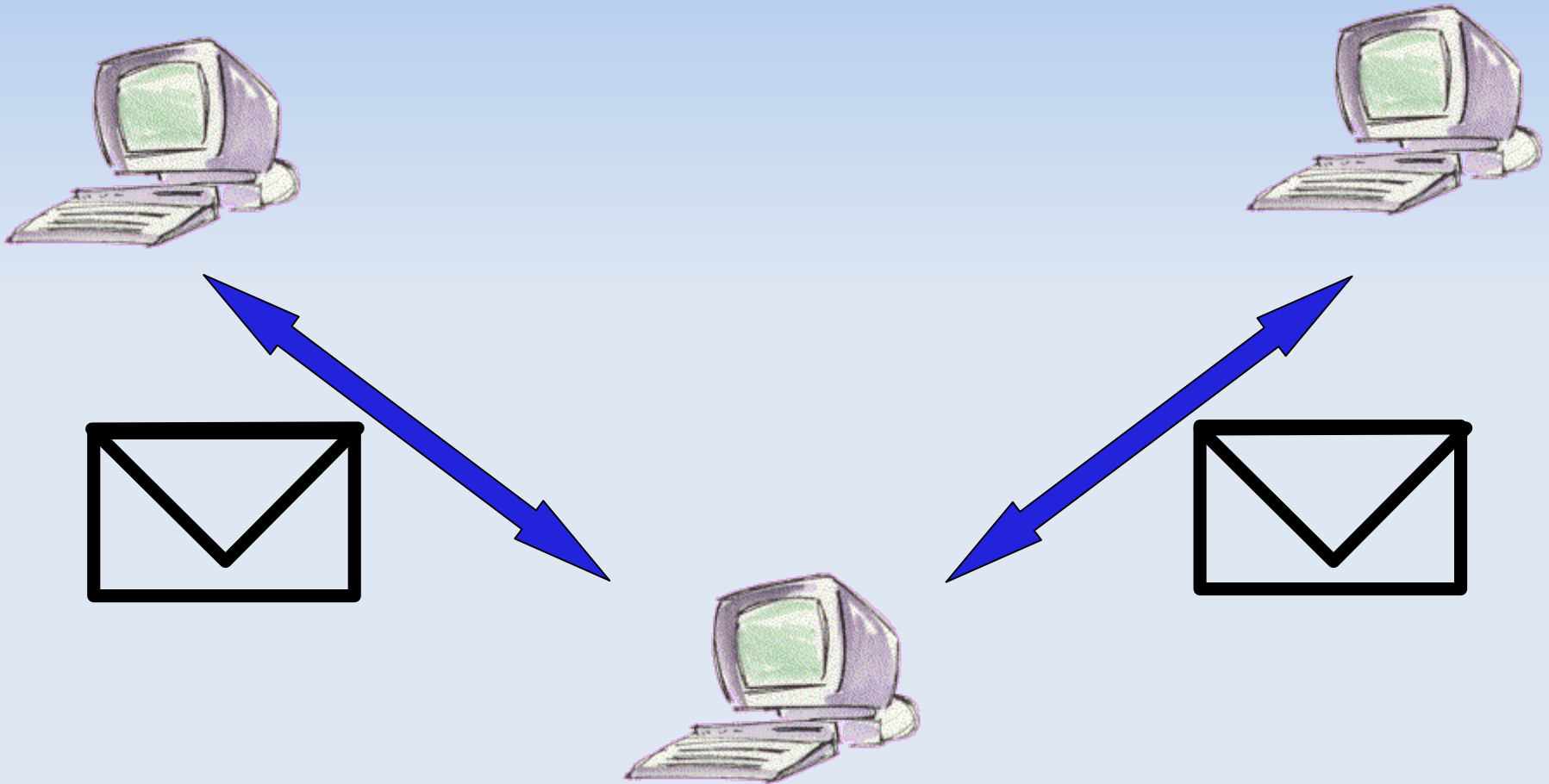
Universität des Saarlandes

# Agenten

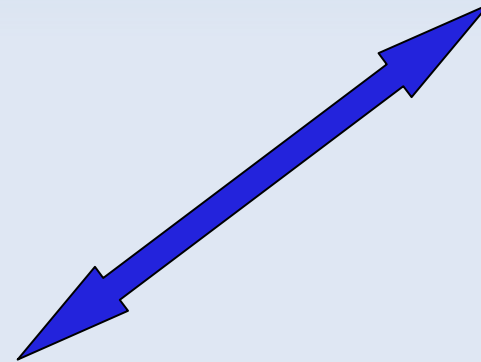
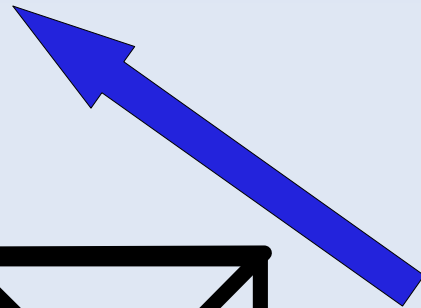
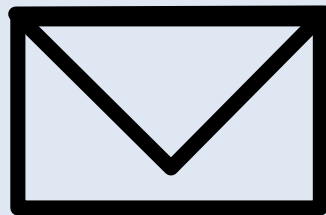
- autonom
- proaktiv
- reaktiv
- zielbasiert
- sozial
- adaptiv
- kognitiv



# Agentensysteme



# Agentensysteme

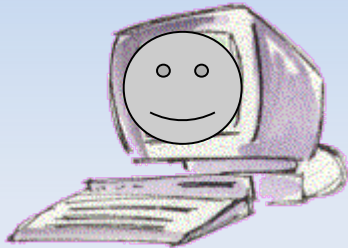


Asynchroner Nachrichtentransfer

# Agentensysteme

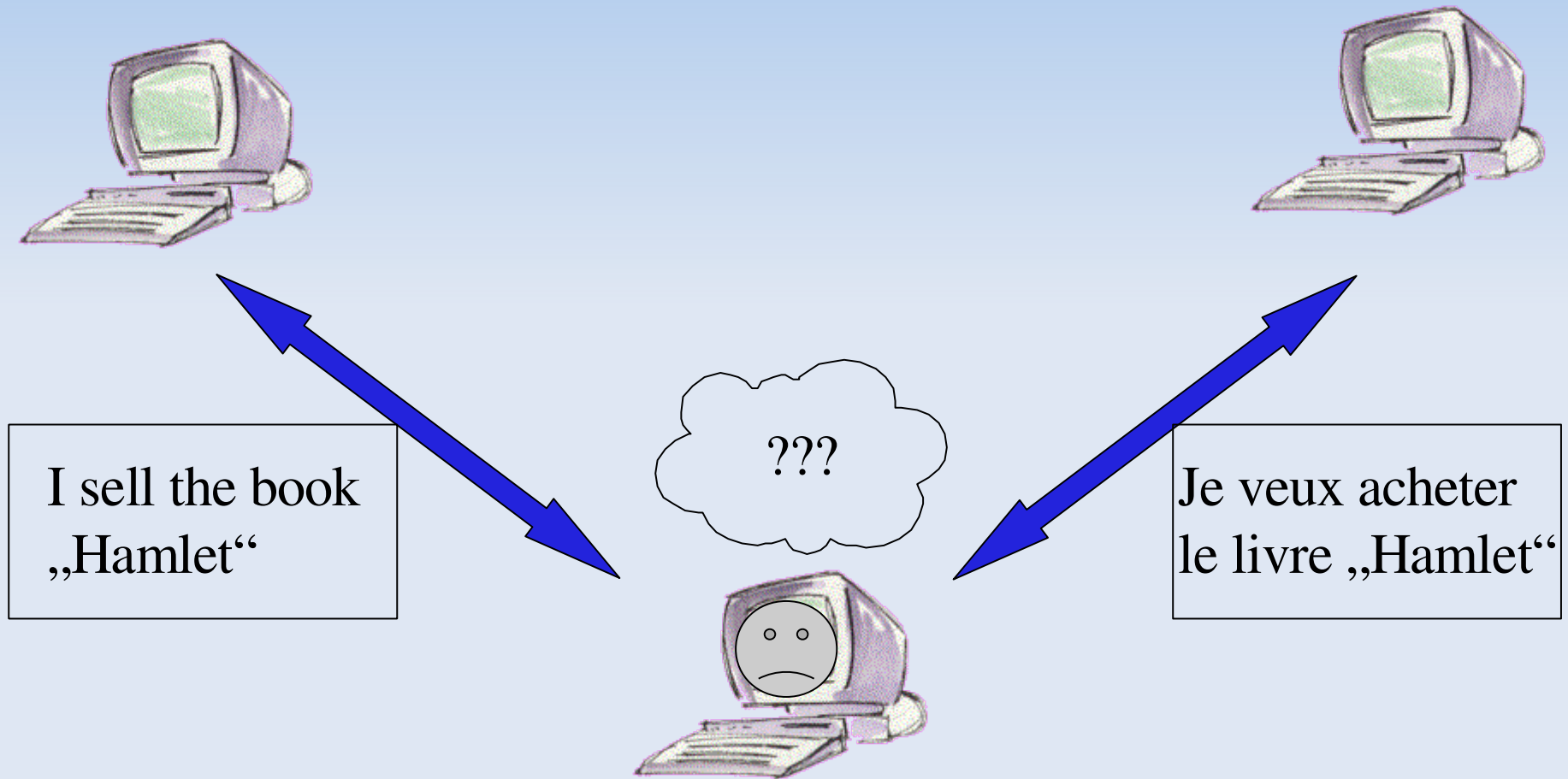
... 2 Tage später...

Ah!! Mail!

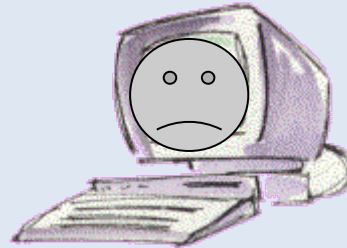
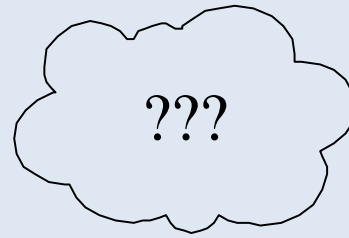
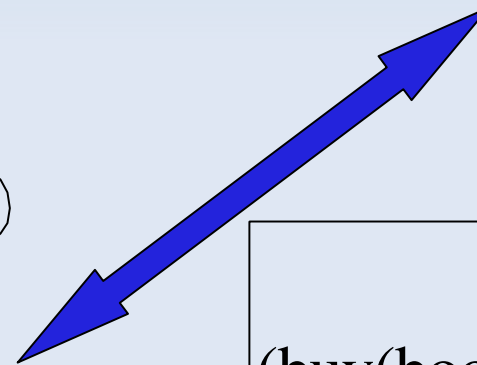
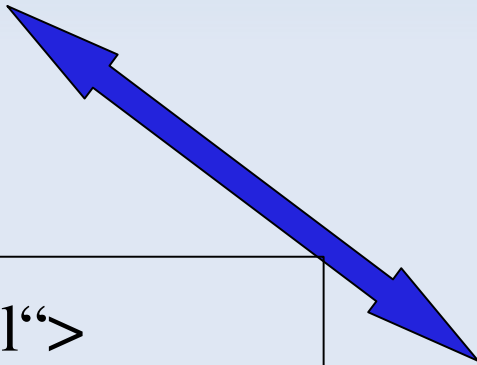


Asynchroner Nachrichtentransfer

# Agentensysteme



# Agentensysteme



```
<action="sell">  
<book title=„Hamlet“/>  
</action>
```

```
(buy(book(„Hamlet“))
```

# Agentensysteme



Standard?

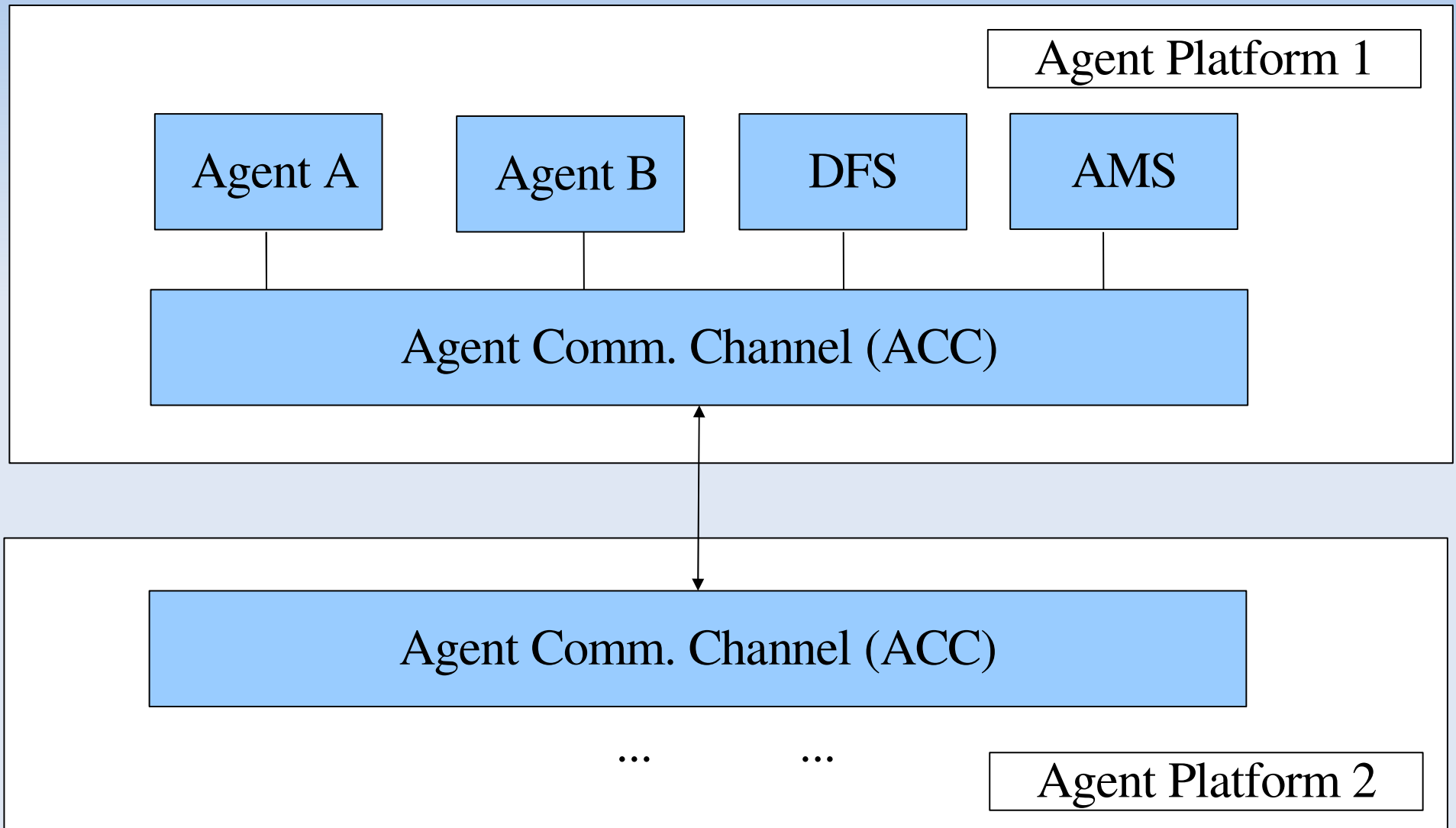
# Agentensysteme

**FIPA!!**



- Foundation for Intelligent Physical Agents
- IEEE Standard seit 2005
- Standards für
  - ACL
  - Agentenarchitektur
- Referenzmodell

# FIPA Modell



- **Agentenplattform (AP):** Physikalische Infrastruktur
- **Agenten:** Aktoren, bieten Dienste an, haben ID
- **Agent Management System (AMS):**  
Erzeugen/Löschen von Agenten,  
Verwaltung von Agenten: „Weiße Seiten“,  
Funktionen: `create`, `execute`, `suspend`, `terminate`

- **Directory Facilitator (DF):**  
Agentenregistrierung, „Gelbe Seiten“,  
Funktionen: `register`, `modify`, `search`
- **Agent Communication Channel (ACC):** Austausch/Propagierung von Nachrichten zw. AP
- **Namensgebung:** Eindeutige ID der Form „`<AgentName>@<AgentAP>.com`“, Transportadresse nach URL-Format

- ACL: Agent Communication Language

Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

**Table 1:** FIPA ACL Message Parameters

Und nun....

Wie baut man ein  
Agentensystem  
nach FIPA?

# Agentensystem nach FIPA

- Lösung 1: FIPA studieren und anwenden
- Lösung 2: JADE

# JADE

- JAVA Middleware
- Effiziente Bearbeitung von P2P Anwendungen
- basiert auf dem Agentenparadigma
- Kabel-Netzwerk & Mobil
- basiert auf FIPA Standard
- API und Ausführungsumgebung

# JADE - Agenten

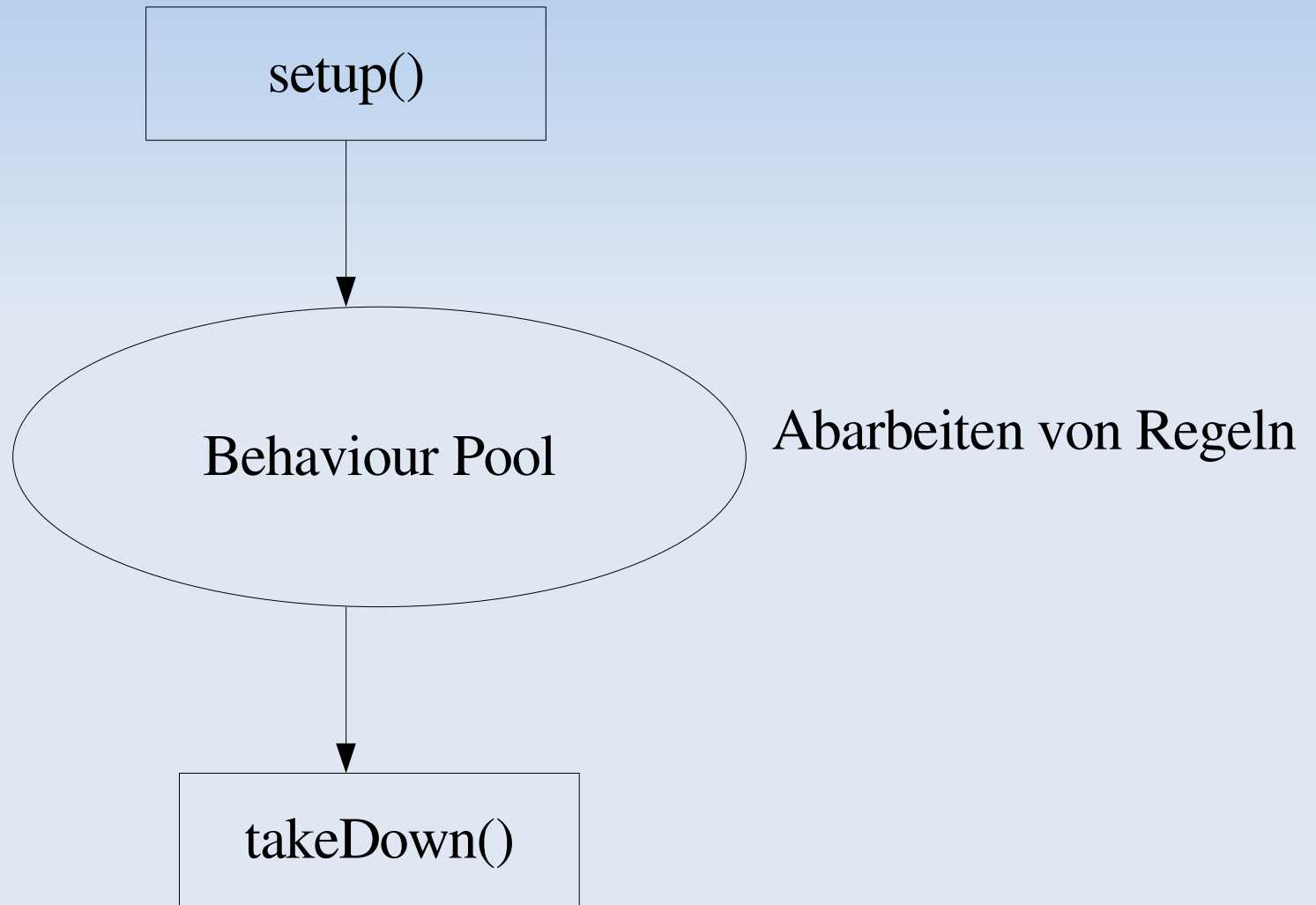
- Starten mit `setup()`
- Behaviours:
  - Methode `addBehaviour()` mit
  - `public void action() // Verhaltensfkt.`
  - `public void done() // Fkt. beendet`
  - sind zu implementieren
- Beenden des Agenten mit
  - `protected void takeDown()`

# JADE - Agenten

- Behaviours
  - One-Shot (einfache Ausführung)
  - Zyklisch (Ausführung solange Agent lebendig)

# JADE - Agenten

## Lifecycle



# JADE – Hallo Welt

```
30 * JADE - Java Agent DEvelopment Framework is a framework to develop
25
26 package examples.hallo;
27 import jade.core.Agent;
28
29
30 public class HalloWorldAgent extends Agent {
31
32     // is executed on Agent startup
33     protected void setup() {
34         System.out.println("Hallo World! My name is "+getLocalName());
35
36         // Make this agent terminate
37         doDelete();
38     }
39 }
40
41
```

# JADE – Hallo Welt

```
30+ * JADE - Java Agent DEvelopment Framework is a framework to develop[]
25
26 package examples.hallo;
27 import jade.core.Agent; → Import der Jade Agentenklasse
28
29
30+ public class HalloWorldAgent extends Agent {
31
32     // is executed on Agent startup
33+ protected void setup() {
34     System.out.println("Hallo World! My name is "+getLocalName());
35
36     // Make this agent terminate
37     doDelete();
38 }
39 }
40
41
```

# JADE – Hallo Welt

```
30 * JADE - Java Agent DEvelopment Framework is a framework to develop
25
26 package examples.hallo;
27 import jade.core.Agent;
28
29
30 public class HalloWorldAgent extends Agent {
31
32     // is executed on Agent startup
33     protected void setup() {
34         System.out.println("Hallo World! My name is "+getLocalName());
35
36         // Make this agent terminate
37         doDelete();
38     }
39 }
40
41
```

→ Definition einer eigenen Agentenklasse

# JADE – Hallo Welt

```
30 * JADE - Java Agent DEvelopment Framework is a framework to develop
25
26 package examples.hallo;
27 import jade.core.Agent;
28
29
30 public class HalloWorldAgent extends Agent {
31
32     // is executed on Agent startup
33     protected void setup() {
34         System.out.println("Hallo World! My name is "+getLocalName());
35
36         // Make this agent terminate
37         doDelete();
38     }
39 }
40
41
```

→ Wird beim StartUp ausgeführt

# JADE – Hallo Welt

```
30 * JADE - Java Agent DEvelopment Framework is a framework to develop
25
26 package examples.hallo;
27 import jade.core.Agent;
28
29
30 public class HalloWorldAgent extends Agent {
31
32     // is executed on Agent startup
33     protected void setup() {
34         System.out.println("Hallo World! My name is "+getLocalName());
35
36         // Make this agent terminate
37         doDelete();
38     }
39 }
40
41
```

 **Agent beenden**

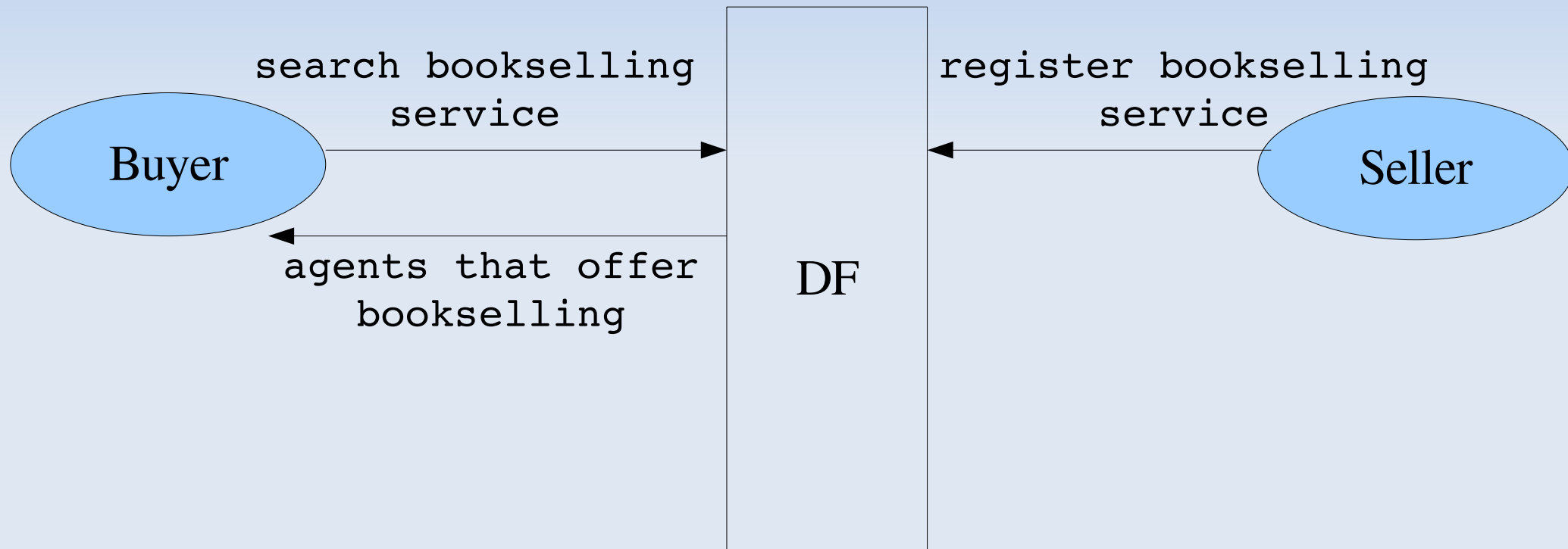
# JADE – Hallo Welt

cathrin@Romeo: /home/cathrin/Desktop/jade/src

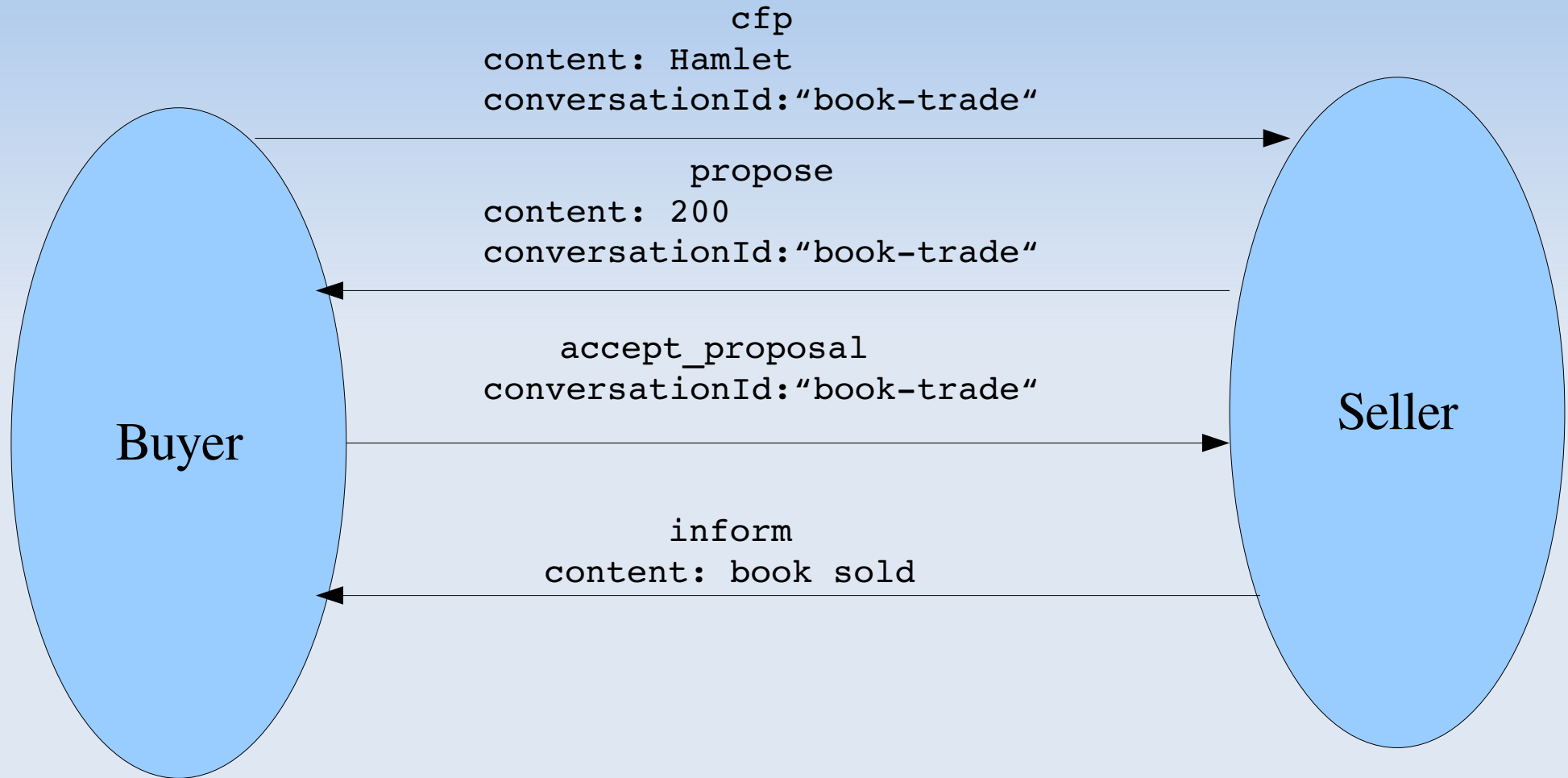
Datei Bearbeiten Ansicht Terminal Reiter Hilfe

```
cathrin@Romeo:~/Desktop/jade/src$ java jade.Boot Karl:examples.hallo.HalloWorldAgent
05.11.2006 13:53:48 jade.core.Runtime beginContainer
INFO: -----
      This is JADE3.4 - revision 5874 of 2006/03/09 14:13:11
      downloaded in Open Source, under LGPL restrictions,
      at http://jade.tilab.com/
-----
05.11.2006 13:53:50 jade.core.BaseService init
INFO: Service jade.core.management.AgentManagement initialized
05.11.2006 13:53:50 jade.core.BaseService init
INFO: Service jade.core.messaging.Messaging initialized
05.11.2006 13:53:50 jade.core.BaseService init
INFO: Service jade.core.mobility.AgentMobility initialized
05.11.2006 13:53:50 jade.core.BaseService init
INFO: Service jade.core.event.Notification initialized
05.11.2006 13:53:51 jade.core.messaging.MessagingService boot
INFO: MTP addresses:
http://Romeo:7778/acc
05.11.2006 13:53:52 jade.core.AgentContainerImpl joinPlatform
INFO: -----
Agent container Main-Container@JADE-IMTP://Romeo is ready.
-----
Hallo World! My name is Karl
█
```

# JADE – Book Trading



# JADE – Book Trading



# JADE – Book Trading

```
42
43 // Put agent initializations here
44 protected void setup() {
45     // Create the catalogue
46     catalogue = new Hashtable();
47
48     // Create and show the GUI
49     myGui = new BookSellerGui(this);
50     myGui.show();
51
52     // Register the book-selling service in the yellow pages
53     DFAgentDescription dfd = new DFAgentDescription();
54     dfd.setName(getAID());
55     ServiceDescription sd = new ServiceDescription();
56     sd.setType("book-selling");
57     sd.setName("JADE-book-trading");
58     dfd.addServices(sd);
59     try {
60         DFService.register(this, dfd);
61     }
62     catch (FIPAException fe) {
63         fe.printStackTrace();
64     }
65
66     // Add the behaviour serving queries from buyer agents
67     addBehaviour(new OfferRequestsServer());
68
69     // Add the behaviour serving purchase orders from buyer agents
70     addBehaviour(new PurchaseOrdersServer());
71 }
72
```

# JADE – Book Trading

```
100  /**
101     Inner class OfferRequestsServer.
102     This is the behaviour used by Book-seller agents to serve incoming requests
103     for offer from buyer agents.
104     If the requested book is in the local catalogue the seller agent replies
105     with a PROPOSE message specifying the price. Otherwise a REFUSE message is
106     sent back.
107     */
108  private class OfferRequestsServer extends CyclicBehaviour {
109      public void action() {
110          MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.CFP);
111          ACLMessage msg = myAgent.receive(mt);
112          if (msg != null) {
113              // CFP Message received. Process it
114              String title = msg.getContent();
115              ACLMessage reply = msg.createReply();
116
117              Integer price = (Integer) catalogue.get(title);
118              if (price != null) {
119                  // The requested book is available for sale. Reply with the price
120                  reply.setPerformative(ACLMessage.PROPOSE);
121                  reply.setContent(String.valueOf(price.intValue()));
122              }
123              else {
124                  // The requested book is NOT available for sale.
125                  reply.setPerformative(ACLMessage.REFUSE);
126                  reply.setContent("not-available");
127              }
128              myAgent.send(reply);
129          }
130          else {
131              block();
132          }
133      }
134  } // End of inner class OfferRequestsServer
135
```

# JADE – Book Trading

```
135
136  /**
137     Inner class PurchaseOrdersServer.
138     This is the behaviour used by Book-seller agents to serve incoming
139     offer acceptances (i.e. purchase orders) from buyer agents.
140     The seller agent removes the purchased book from its catalogue
141     and replies with an INFORM message to notify the buyer that the
142     purchase has been sucesfully completed.
143  */
144  private class PurchaseOrdersServer extends CyclicBehaviour {
145      public void action() {
146          MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.ACCEPT_PROPOSAL);
147          ACLMessage msg = myAgent.receive(mt);
148          if (msg != null) {
149              // ACCEPT_PROPOSAL Message received. Process it
150              String title = msg.getContent();
151              ACLMessage reply = msg.createReply();
152
153              Integer price = (Integer) catalogue.remove(title);
154              if (price != null) {
155                  reply.setPerformative(ACLMessage.INFORM);
156                  System.out.println(title+" sold to agent "+msg.getSender().getName());
157              }
158              else {
159                  // The requested book has been sold to another buyer in the meanwhile .
160                  reply.setPerformative(ACLMessage.FAILURE);
161                  reply.setContent("not-available");
162              }
163              myAgent.send(reply);
164          }
165          else {
166              block();
167          }
168      }
169  } // End of inner class OfferRequestsServer
170
```

# JADE – Book Trading

**DEMO**

# Zusammenfassung

- Um ein MAS zu erstellen, bedarf es einer Standardisierung
- Zur Standardisierung von Architektur und Kommunikation: FIPA
- JADE ist ein JAVA Tool, das zur Erstellung und Ausführung FIPA konformer MAS dienlich ist
- Einbindung von JESS zur Regelerstellung möglich

# Zusammenfassung

- Agentenkommunikation verläuft eingebettet
- Nur wenig Kenntnis vom FIPA Standard notwendig
- Plattformunabhängig

Das war's!



**FRAGEN??**

