AOSE

Agentenorientierte Softwareentwicklung

Multiagentensysteme: Lehre, Praxis und Forschungsaktivitäten

Arbeitskreis Objekttechnologie Norddeutschland
74. Treffen am 15. Oktober 2007

Hochschule für Angewandte Wissenschaften (HAW) Hamburg
Fakultät Technik und Informatik
MMLab

Wolfgang Renz
wr@informatik.haw-hamburg.de

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

MMLab
Labor für multimediale Systeme

Agenda

- **Motivation: Praxistauglichkeit**
  - *Adaptive Systeme bauen*
- **Multiagentensysteme (MAS)**
  - *Open-Source-Projekt Jadex*
  - **Agentenorientierte Programmierung, AOSE**
    - *Tool support, Unreal Tournament 2003*
  - **Forschung: Selbstorganisation in Software**
    - *allgemeines Prinzip unabhängig von MAS*
- **Schlussfolgerungen**
  - *wachsendes Interesse*

Problemstellung

- Allgemeine Problemstellung
  - allgemeine Anforderungen

- *Adaptive* Systeme bauen,
  - *die sich zur Laufzeit an dynamische Veränderungen anpassen,*
  - Komplexität beherrschen, Skalierbarkeit, Modularisierung,
  - Flexibilität im Deployment, Integrierbarkeit durch Serviceorientierung,
  - Deskriptive Programmierung, MDA

Multiagentensysteme (MAS) als Lösungsvorschlag

Motivation sie zu studieren ist ihre Praxistauglichkeit:

Motivation: Praxistauglichkeit

Multiagentenbasierte kommerzielle Lösungen (1):

Agentis

- Agentis *Adaptive Enterprise*™
  - "A goal-oriented modeling environment for IT developers and business analysts"
  - "Goal-Oriented Modeling"

    ....defining a few simple plans, consisting of a few steps...
  - "Goal-Directed Execution"

    Software agents execute services dynamically at run-time based on the immediate goal and current conditions.
  - Anwendungsfelder:

IT-Infrastruktur, Transport&Logistik, Energieversorger, Bank&Versicherung

aus: http://www.agentissoftware.com/en/solutions/features.jsp

Motivation: Praxistauglichkeit

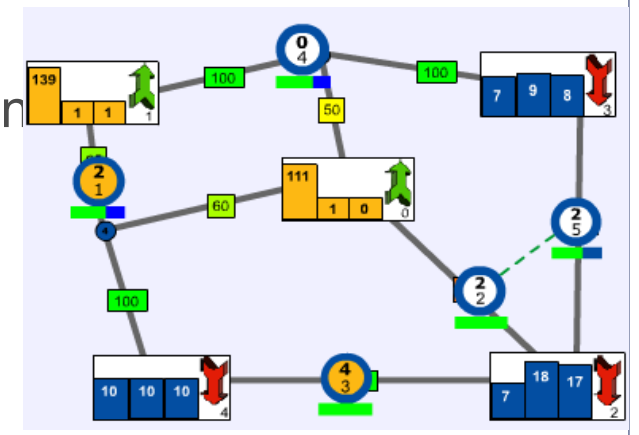
Multiagentenbasierte kommerzielle Lösungen (2):

- Living Systems® von Whitestein Technologies
  - Technology Suite:

    J2SE und J2EE-gestützte Agentenplattform
    Eclipse-basierte Entwicklungswerkzeuge,
    Entwicklungsmethodik UML2.0 mit EPF sowie
    Agentenerweiterung AML mit RUP-Apassung (ADEM)
  - Logistik: Adaptive Network Solutions

    für Transport, Produktion und Supply-Chain
  - Telekommunikation:

    Network Access Management
  - Banken und Versicherungen



Dynamik und Komplexität händeln durch *Adaptivität zur Laufzeit*

aus: http://www.whitestein.com/pages/solutions/overview.html

Was ist ein Agent ?

- Russell/Norvig:

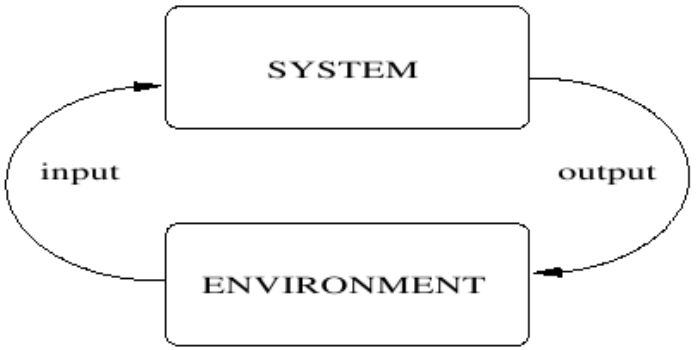
  An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

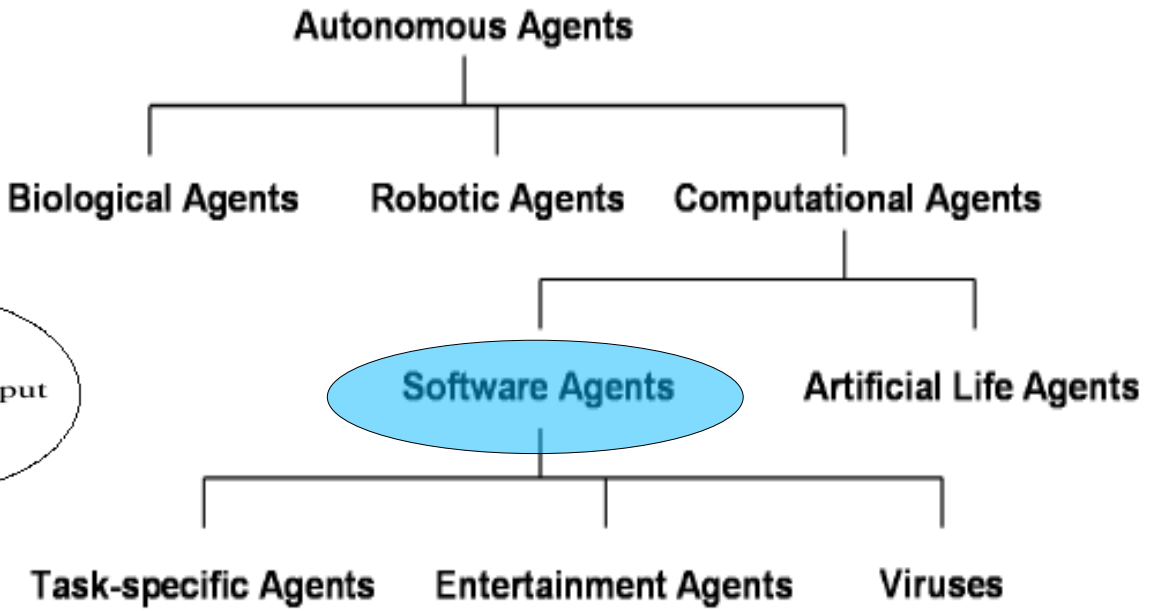
  Autonome Einheit, situiert in einer Umgebung,

  hat Aufträge zu erfüllen.

- Taxonomie (1996)

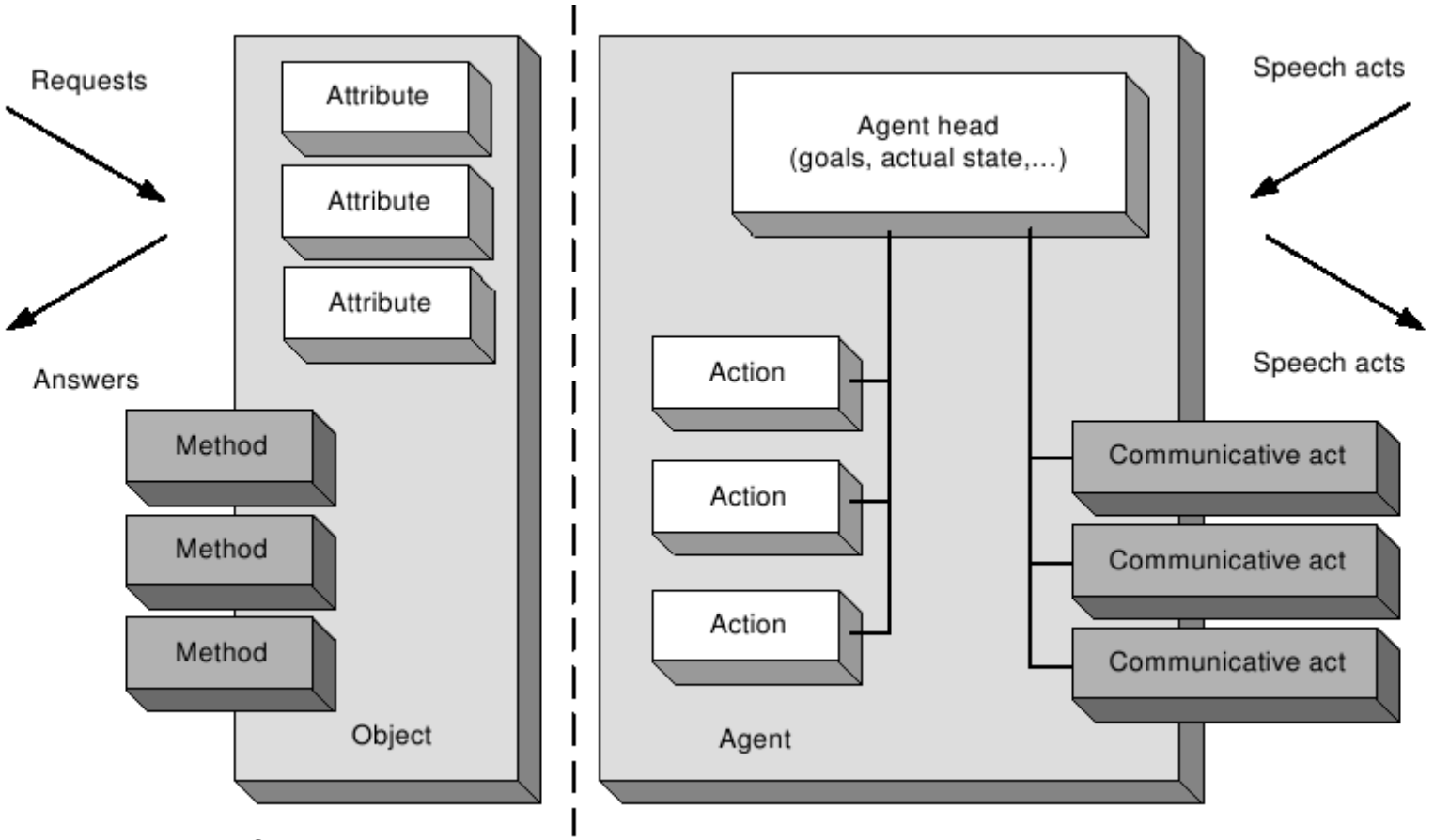
  Franklin & Grasser





Objekt vs. Agent

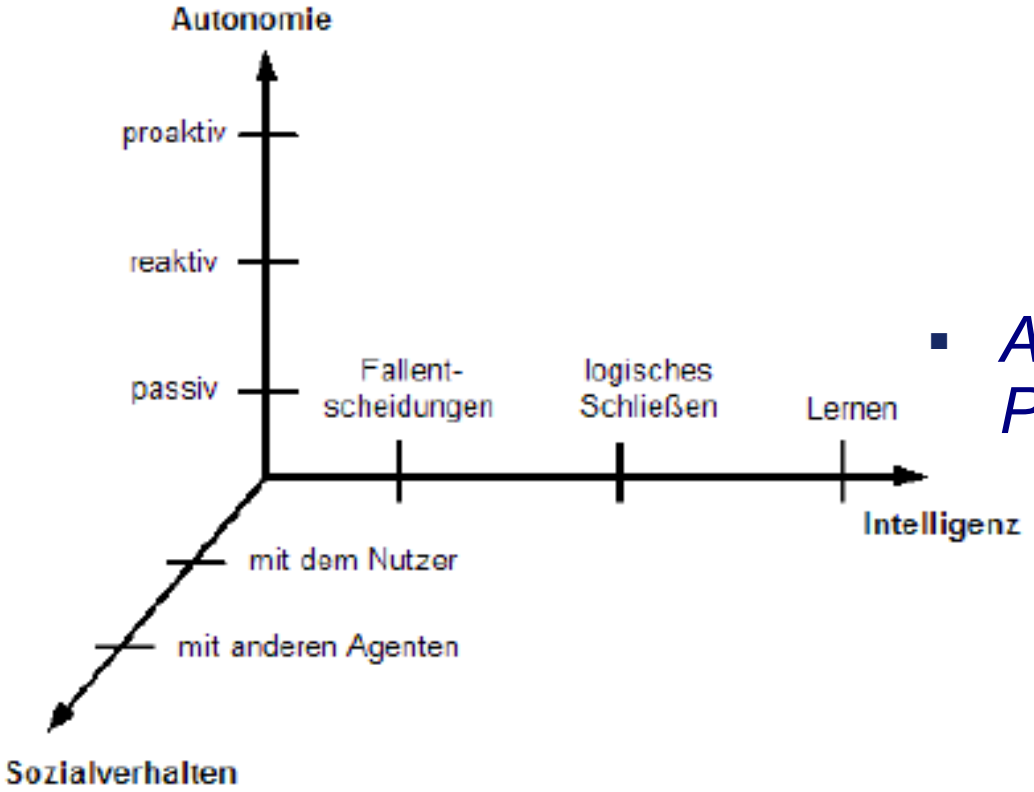
- Agents have the right to say “go” or “no” !
- Objects do it for free – Agents do it for Money (or: because they want to)



Charakteristika von Agenten

- Rationalität:

  Ein Agent ist auf die Erfüllung seiner Ziele aus und wird diese optimieren.



- Agenten als Paradigma für das Software Engineering:

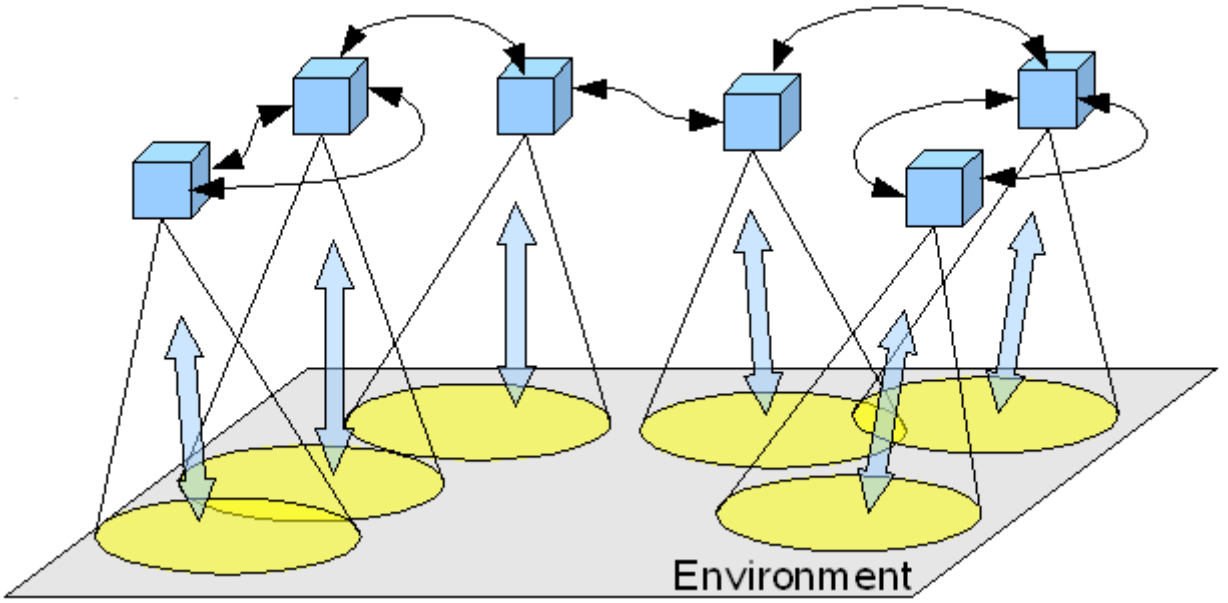
  Zentrales Ziel zur Beherrschung von Komplexität ist die Modularisierung, d.h. die Strukturierung in lose gekoppelte Module.

- *Agent-oriented Programming* (AOP)

  Yoav Shoham (1993)
  Neues Programmierparadigma für MAS!

Multi-Agenten Systeme (MAS)

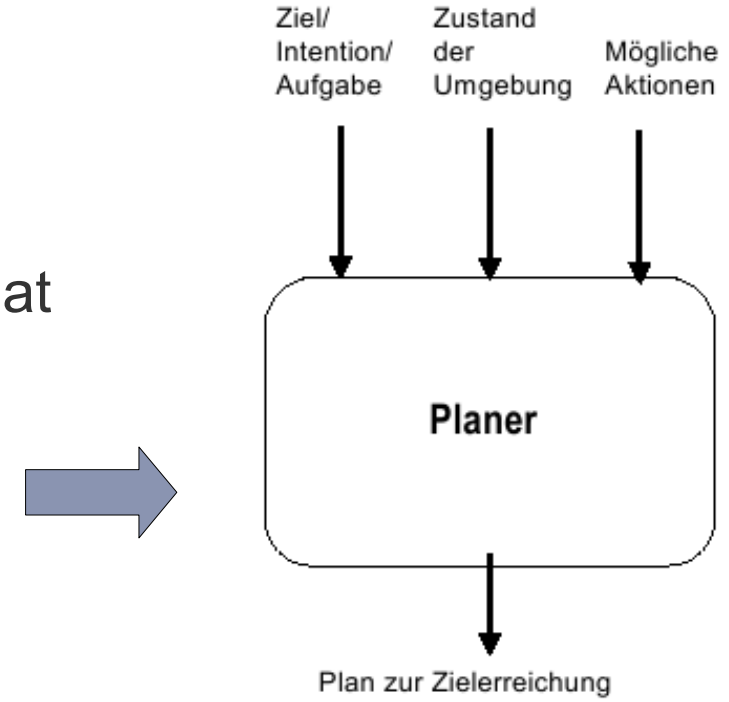
- Ein MAS besteht aus mehreren Agenten, die miteinander interagieren
  - über direkte Kommunikation (*Sprechakte*) oder
  - indirekt über die Umgebung („*Environment Mediated*“)
- Zusammenwirken (Kooperation) mehrerer Agenten:
  Organisationen interagierender Elemente (Koordination)
- Agent hat lokale Wahrnehmung seiner Umgebung
- Umgebung
  - Dynamisch
  - i.a. Nicht-Deterministisch



Intentionale Agenten

BDI-Modell (Rao&Georgeff 1995): Agenten werden direkt auf der Basis intentionaler Begriffe programmiert:

- **B**eliefs: Weltwissen des Agenten
- **D**esires / Goals:
  Ziele repräsentieren die Aufträge
- **I**ntentions:
  Optionen, die ein Agent gewählt hat

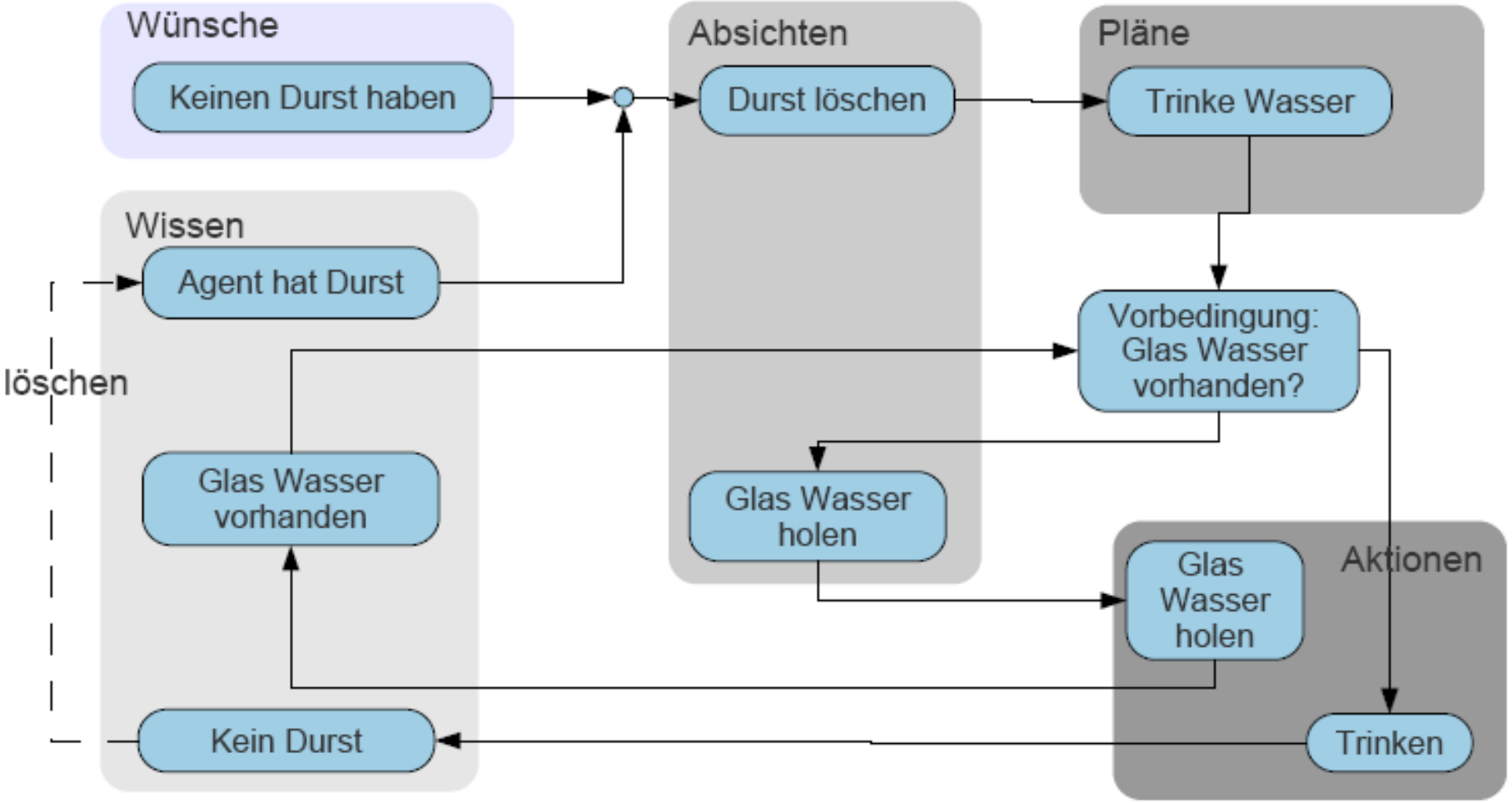


Reaktives Planen (in Main Loop):

- "Means-End Reasoning"
  erzeugt ausführbare Mittel (Pläne)
- Deliberation:
  - Wähle aus den Zielen Optionen für mögliche Intentionen,
  - Erzeuge eine Auswahl aus diesen Optionen,
  - Treffe Commitment (Festlegung) auf eine oder mehrere davon.

Beispiel

- **Durstiger Agent in BDI**



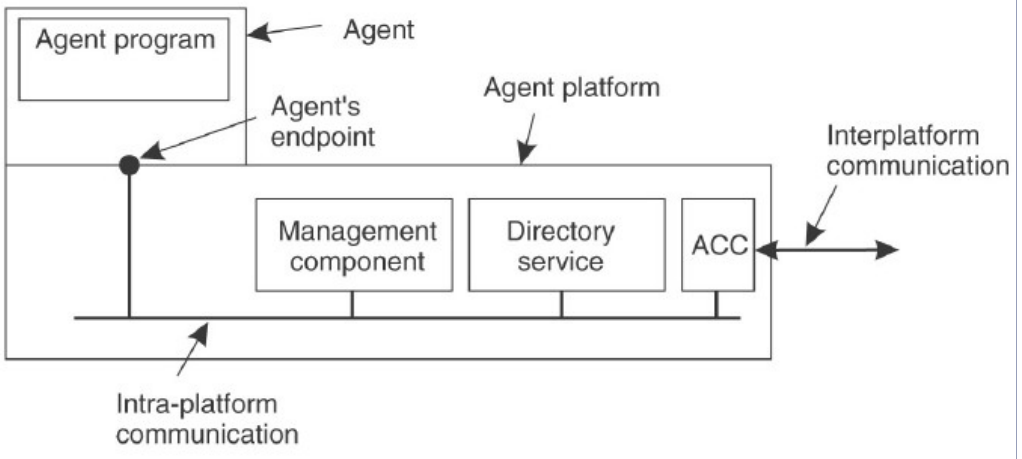
© Zapf, Universität Dortmund, Computer Science Dept.

Jadex Plattform

BDI Agent System

- Software-Engineering-Umgebung zur Konstruktion von MAS
  - BDI Architektur mit Reasoning Engine
    - Java (ausführbare Pläne)
    - XML (deskriptive Programmierung der Goals und Beliefs)
  - Laufzeitumgebung mit Middlewarefunktionalität
    - Kommunikation, Gelber-Seiten-Service, Plattform-Management
    - FIPA-Standard-konform (JADE-basiert) oder Standalone

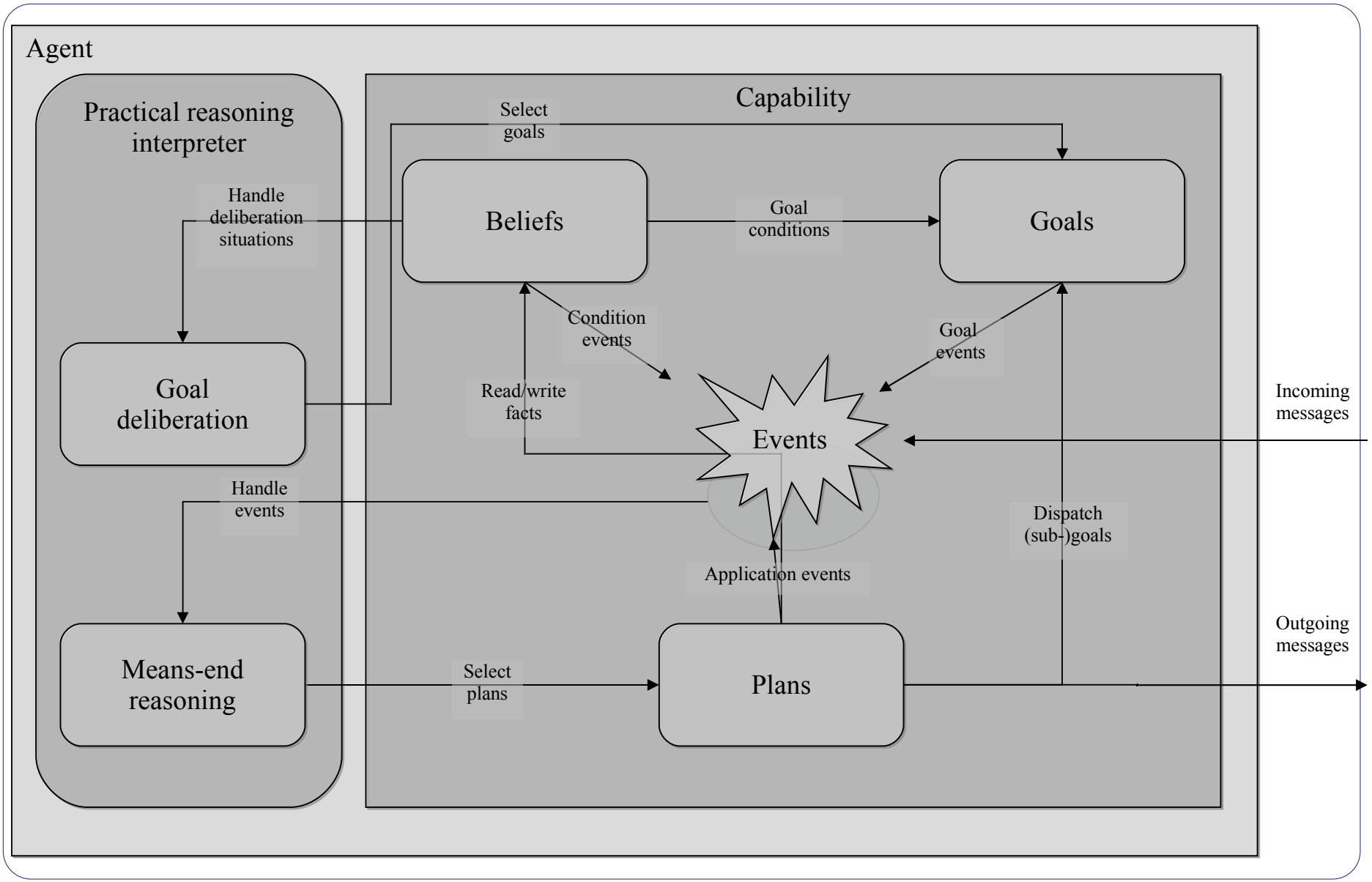
http://www.fipa.org
http://sharon.cselt.it/projects/jade/papers/PAAM.pdf



  - Entwicklungswerkzeuge
  - Open Source (LGPL License):

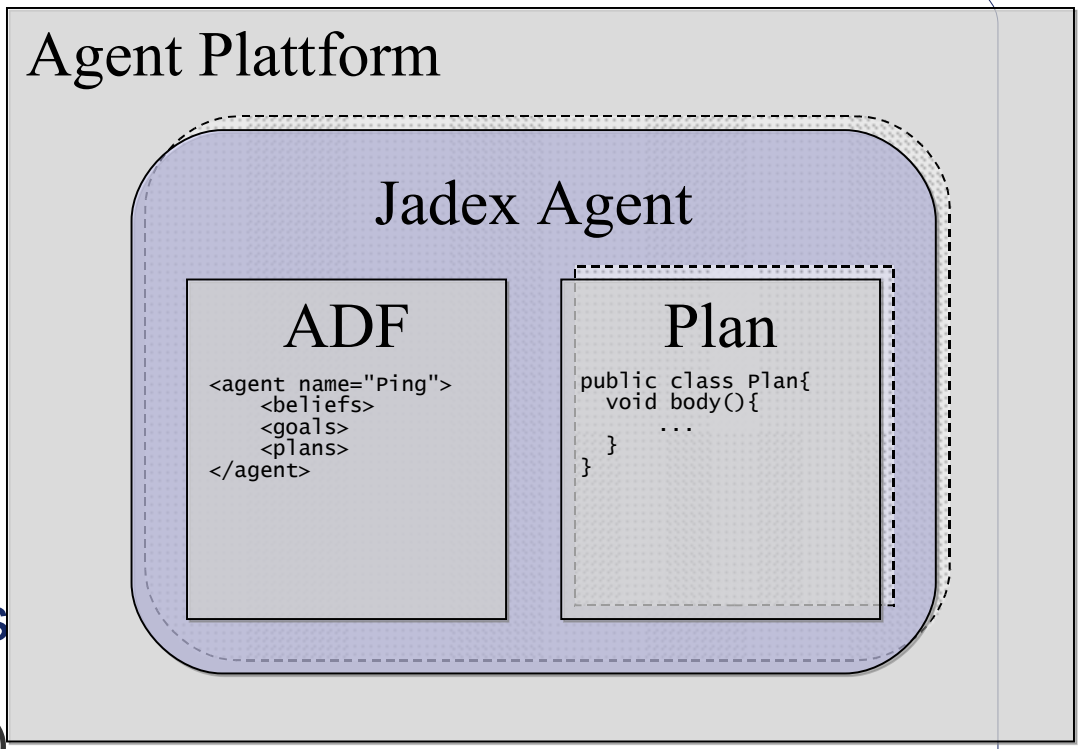
http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

Jadex BDI Agent System — Agenten-Architektur



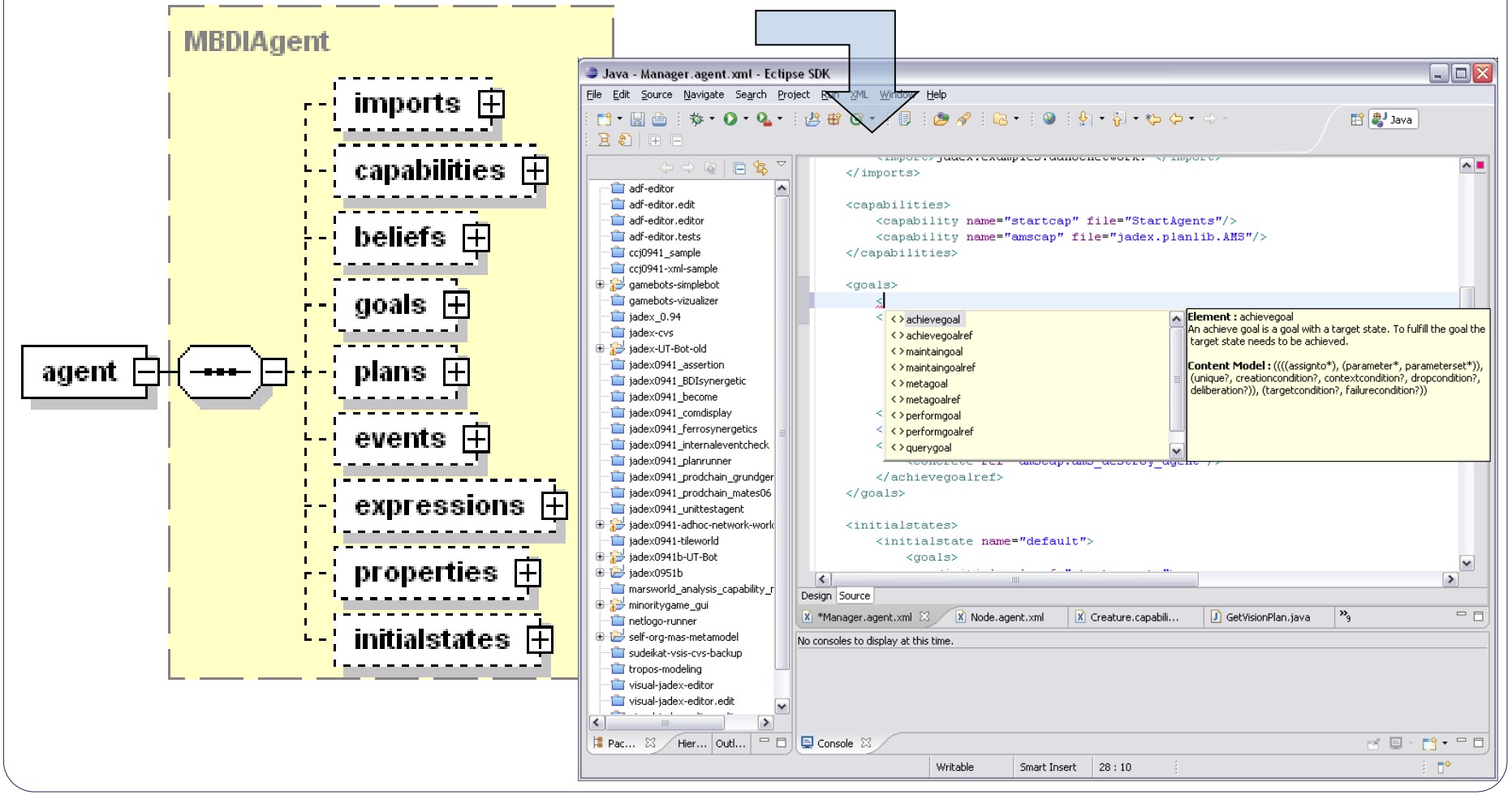
Agentenprogrammierung

- Basiert nicht auf einer agenten-orientierten Programmiersprache
- Stattdessen: Hybrider Ansatz:
  - Deklaration statischer Eigenschaften
    - Agent Definition Files (ADF)
    - XML (BDI-Meta-Model)
  - Ausführbare Mittel in Java Programmiersprache
    - API
      - Manipulation von ADF-Elementen zur Laufzeit
      - Benutzung von Plattform Services zur Laufzeit
    - Benutzung von Java Techniken möglich (Third-Party APIs)



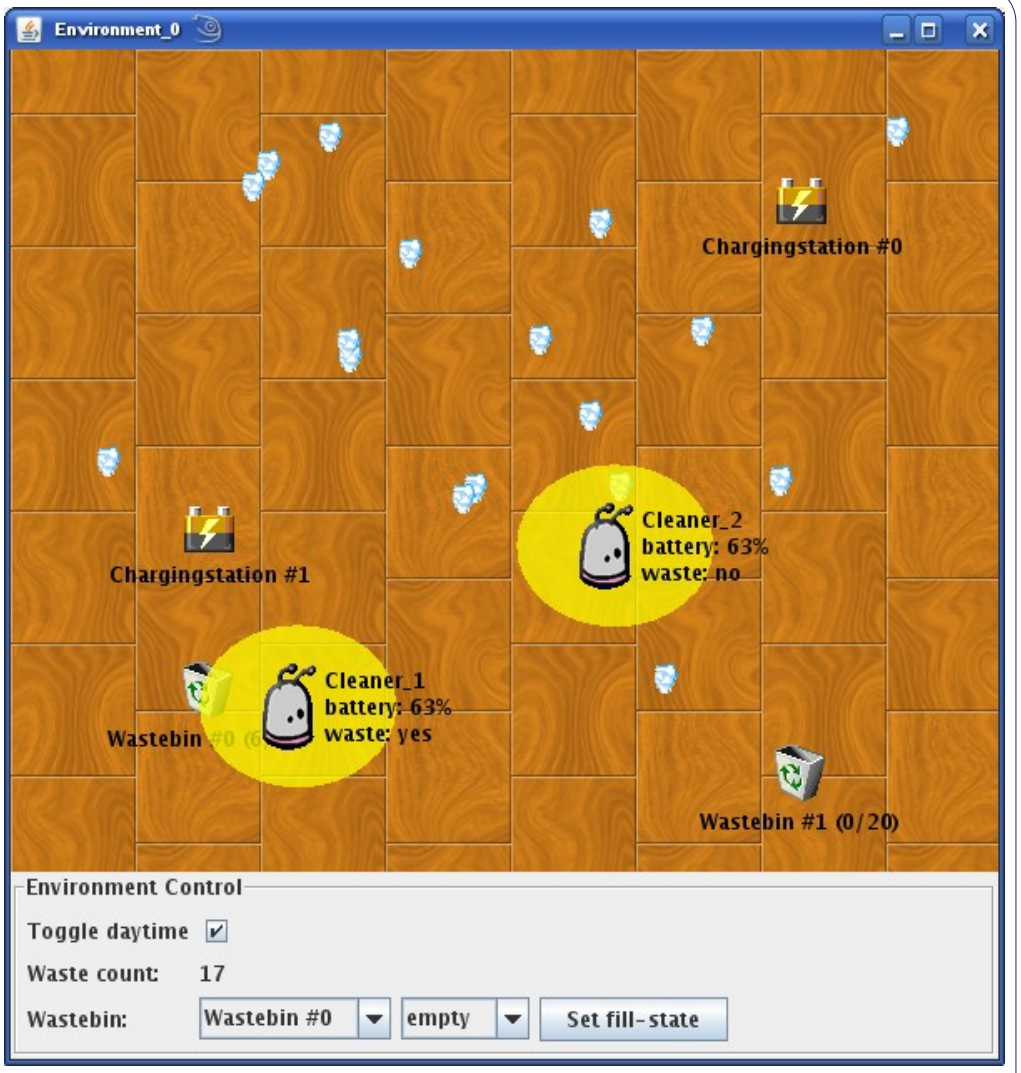
Das Jadex Meta-Model

- Beschrieben in XML-Schema
- Assistenz bei der Erstellung von ADF



Beispiel: Cleanerworld

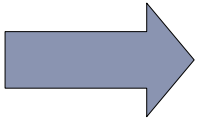
- Agenten sammeln “Müll” auf und transportieren ihn in die Mülleimer
  - Batterie-Ladung
  - Mülleimer-Füllgrad
- Nachts:
  - Routen abfahren
- Eingeschränktes MAS
  - (+) Dynamische Umgebung
  - (-) keine Kooperation
  - (-) keine Koordination



Jadex Agent Specification (1): Beliefs

- Die Wissensbasis (“Beliefbase”) beinhaltet das lokale Wissen eines Agenten
  - Java Objekte
  - key / value Paare

Name
Typ
Wert

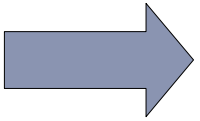


```
<!-- The current cleaner location. -->
<belief name="my_location"  class="Location">
    <fact>new Location(0.2, 0.2)</fact>
</belief>

<!-- The speed. -->
<belief name="my_speed" class="double">
    <fact>3</fact>
</belief>

<!-- The radius describing how far the agent can see. -->
<belief name="my_vision" class="double">
    <fact>0.1</fact>
</belief>
```

Jadex Agent Specification (1): Beliefs

- Die Wissensbasis (“Beliefbase”) beinhaltet das lokale Wissen eines Agenten
  - Java Objekte
  - key / value Paare
  - Beliefsets (Listen von Fakten einer Klasse)

Name
Typ
Werte



```
<!-- The points used for patrolling at night. -->
<beliefset name="patrolpoints" class="Location">
    <fact>new Location(0.1, 0.1)</fact>
    <fact>new Location(0.1, 0.9)</fact>
    <fact>new Location(0.3, 0.9)</fact>
    <fact>new Location(0.3, 0.1)</fact>
    <fact>new Location(0.5, 0.1)</fact>
    <fact>new Location(0.5, 0.9)</fact>
    <fact>new Location(0.7, 0.9)</fact>
    <fact>new Location(0.7, 0.1)</fact>
    <fact>new Location(0.9, 0.1)</fact>
    <fact>new Location(0.9, 0.9)</fact>
</beliefset>
```

Jadex Agent Specification (1): Beliefs

- Die Wissensbasis (“Beliefbase”) beinhaltet das lokale Wissen eines Agenten
  - Java Objekte
  - key / value Paare
  - Beliefsets (Listen von Fakten einer Klasse)
  - dynamische evaluation von Ausdrücken

```
<!-- A belief holding the current time (re-evaluated on every access). -->
<belief name="time" class="long">
    <fact evaluationmode="dynamic">
        System.currentTimeMillis()
    </fact>
</belief>

<!-- A belief continuously updated every 10 seconds. -->
<belief name="timer" class="long" updaterate="10000">
    <fact> System.currentTimeMillis() </fact>
</belief>
```

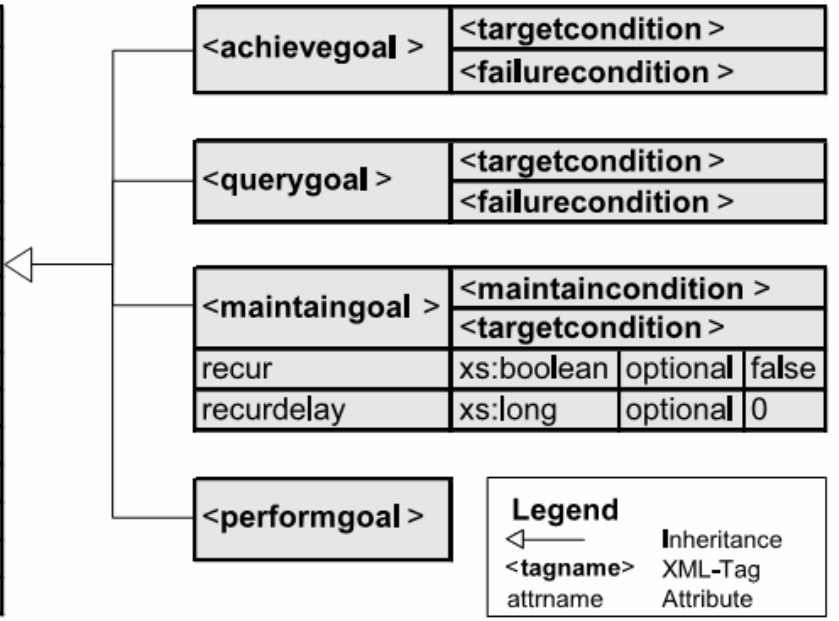
bei jedem Zugriff

kontinuierlich

Jadex Agent Specification (2): Goals

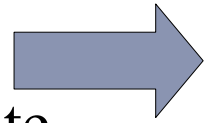
- Ziele sind typisiert, es gibt 4 Arten:
  - Achieve: Herbeiführung eines bestimmten Zustands
  - Maintain: Beibehaltung eines gewünschten Zustands
  - Perform: Ausführung eines Planes
  - Query: Beschaffung eines (Wissens-)Wertes

goal (abstract)	<parameter > (0..n)		
	<parameterset > (0..n)		
	<bindings >		
	<creationcondition >		
	<contextcondition >		
	<dropcondition >		
Attribute	Type	Use	Default
name	xs:string		
description	xs:string	optional	
exported	xs:boolean	optional	false
retry	xs:boolean	optional	true
retrydelay	xs:long	optional	0
mlreasoning	xs:boolean	optional	true
randomselection	xs:boolean	optional	false
exclude	xs:boolean	optional	when_tried
posttoall	xs:boolean	optional	false



XML-Struktur

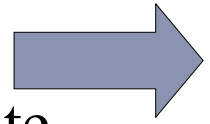
Performgoal



Invariante

```
<!-- Look out for waste when nothing better to do, what means that
    the agent is not cleaning, not loading and it is daytime. -->
<performgoal name="performlookforwaste" retry="true" exclude="never">
    <contextcondition>
        $beliefbase.daytime
    </contextcondition>
</performgoal>
```

erneute Ausführung

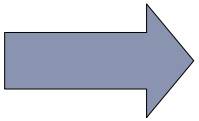


Invariante

```
<!-- Perform patrols at night when the agent is not loading. -->
<performgoal name="performpatrol" retry="true" exclude="never">
    <contextcondition>
        !$beliefbase.daytime
    </contextcondition>
</performgoal>
```

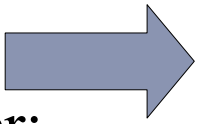
Achieve Goal

Ziel-Bedingung:



```
<!-- Try to move to the specified location. -->
<achievegoal name="achievemoveto">
    <parameter name="location" class="Location"/>
    <!-- The goal has been reached when the agent's location is
        near the target position as specified in the parameter. -->
    <targetcondition>
        $beliefbase.my_location.isNear($goal.location)
    </targetcondition>
</achievegoal>
```

Parameter:
- Name
- Typ



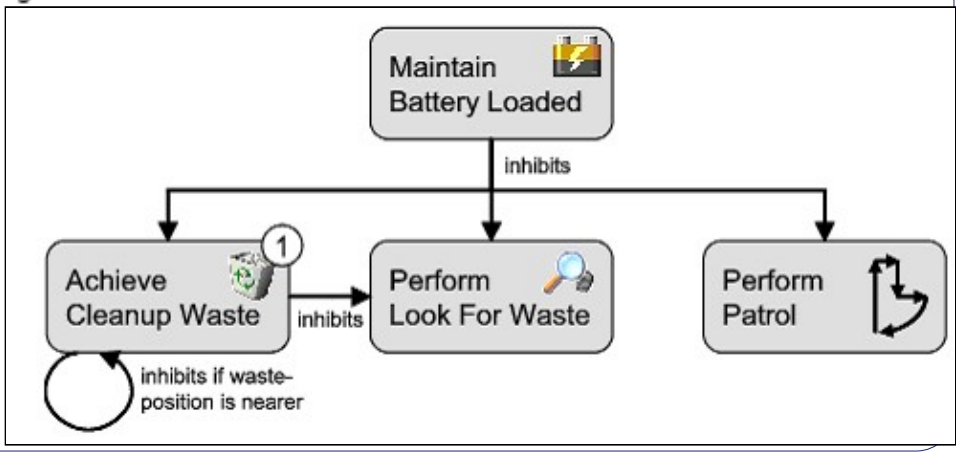
```
<!-- Drop a piece of waste. -->
<achievegoal name="drop_waste_action">
    <parameter name="waste" class="Waste"/>
    <parameter name="wastebin" class="Wastebin"/>
</achievegoal>
```

Maintaingoal

```
<!-- Observe the battery state. -->
<maintaingoal name="maintainbatteryloaded">
    <deliberation>
        <inhibits ref="performlookforwaste" inhibit="when_in_process"/>
        <inhibits ref="achievecleanup" inhibit="when_in_process"/>
        <inhibits ref="performpatrol" inhibit="when_in_process"/>
    </deliberation>
    <!-- Engage in actions when the state is below 0.2. -->
    <maintaincondition>
        $beliefbase.my_chargestate > 0.2
    </maintaincondition>
    <!-- The goal is satisfied when the charge state is 1.0. -->
    <targetcondition>
        $beliefbase.my_chargestate >= 1.0
    </targetcondition>
</maintaingoal>
```

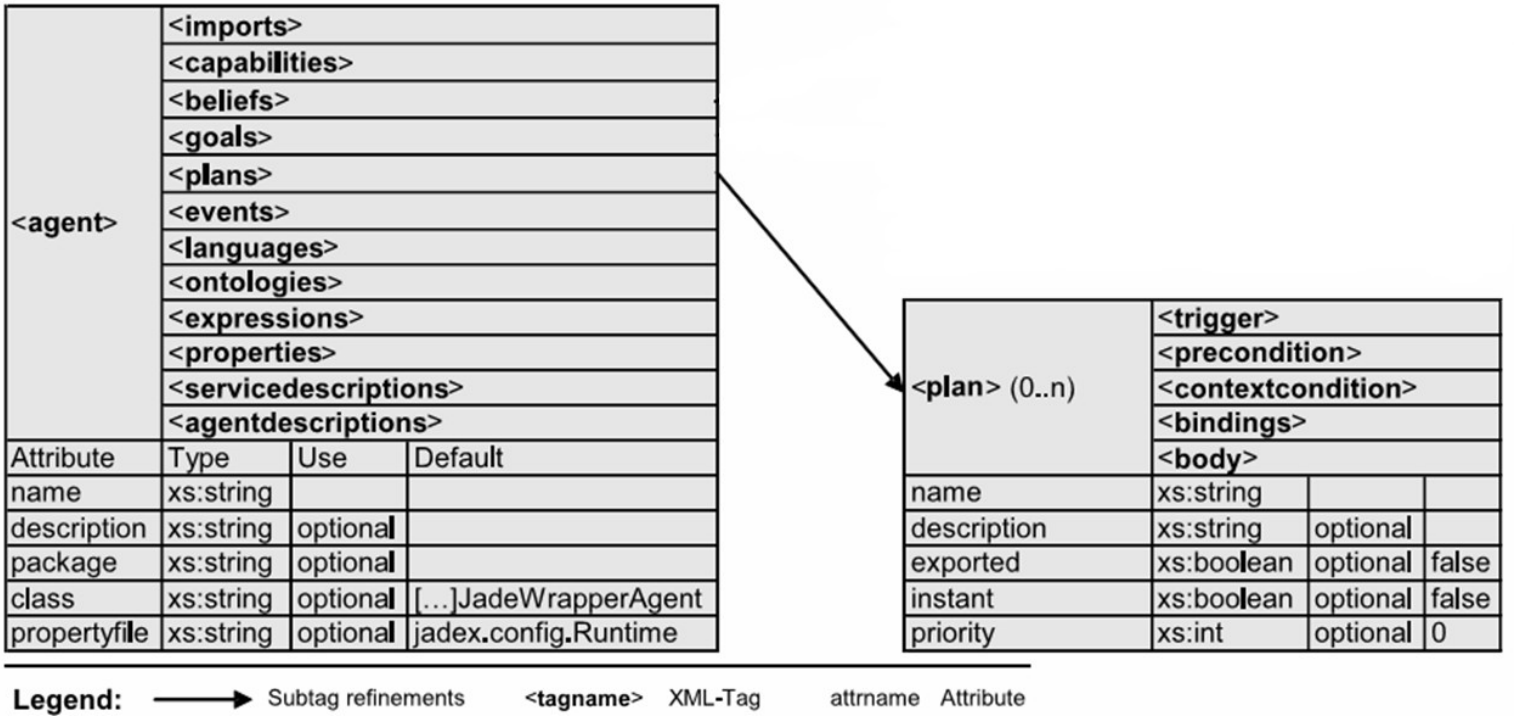
"Verletzte" Bedingung:

Ziel-Bedingung:



Jadex Agent Specification (3): Plans

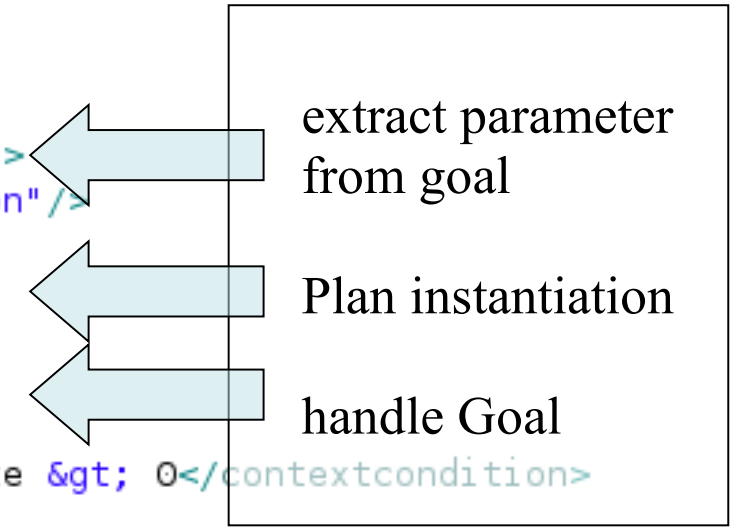
- Pläne beschreiben prozedurales Wissen
  - Mittel um:
    - Ziele zu erfüllen
    - Auf Ereignisse zu reagieren
- Plan Realisierung
  - Plan head im ADF
  - Plan body in Java



Jadex Agent Specification (3): Plans

- Pläne beschreiben prozedurales Wissen
  - Mittel um:
    - Ziele zu erfüllen
    - Auf Ereignisse zu reagieren
- Plan Realisierung
  - Plan head im ADF
  - Plan body in Java

```
<!-- Move to a location. -->
<plan name="moveto">
    <parameter name="location" class="Location">
        <goalmapping ref="achievemoveto.location"/>
    </parameter>
    <body>new MoveToLocationPlan()</body>
    <trigger>
        <goal ref="achievemoveto"/>
    </trigger>
    <contextcondition>$beliefbase.my_chargestate &gt; 0</contextcondition>
</plan>
```



Jadex Agent Specification (3): Plan Body

- Einfache Java Klassen
- API
  - Agenten-Konzepte
  - Plattform Services

E. g.:
Belief access or dispatching Sub-goals

```
/**
 *  The plan body.
 */
public void body()
{
  Target target = (Target) getBeliefbase().getBelief("target").getValue();

  // Move to the target.
  IGoal go_target = createGoal("move_dest");
  go_target.getParameter("destination").setValue(target.getLocation());
  dispatchSubgoalAndWait(go_target);
```

```
import jadex.runtime.Plan;

/** Plan skeleton.  */
public class SomePlan extends Plan {

    public void body() {
        // Plan code...
    }

    public void passed(){
        // Optional clean-up code in case of a plan success.
    }

    public void failed(){
        // Optional clean-up code in case of a plan failure.
    }

    public void aborted(){
        // Optional clean-up code in case the plan is aborted.
    }
}
```

MAS @ MMLab

- Beispiele für Projekte mit Agentenbezug im MMLab

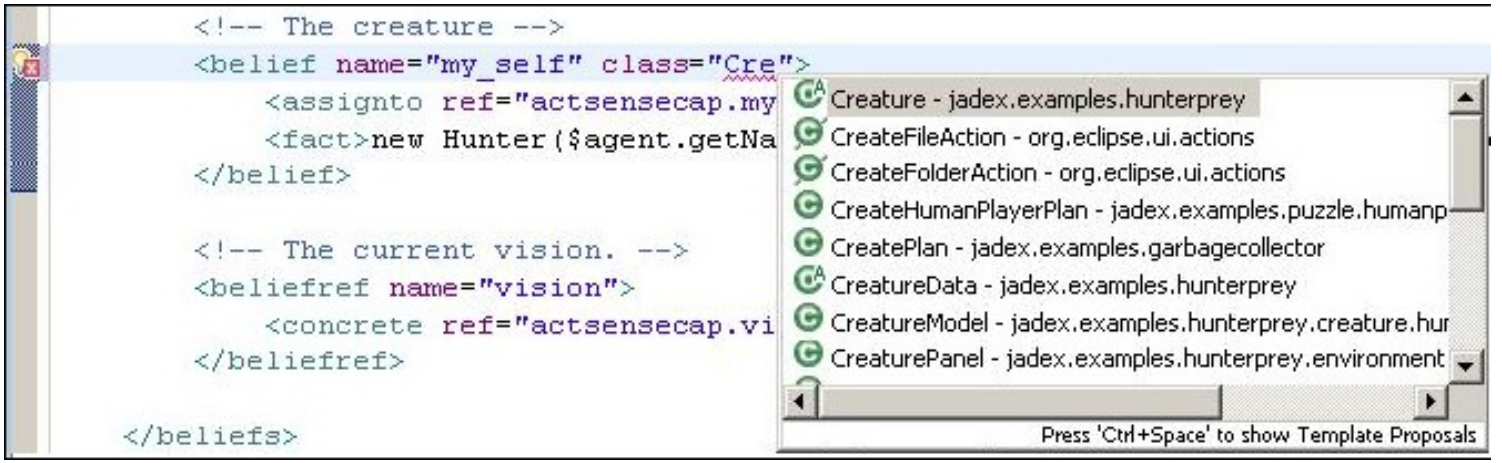
- Bachelorarbeit Konstantin Rollhäuser, Alexander Seizeller:
  Implementationsunterstützung für Webapplikationen und Multiagentensysteme am Beispiel von Struts, X3D und Jadex
- Bachelorarbeit: Wladimir Ruwinski und Dennis Timotin
  Entwicklung eines Multiagentensystem-basierten Frameworks zur Simulation logistischer Prozesse
- Vorlesung WP Sommersem. 2007:
  Einführung in die Agentenorientierte Softwareentwicklung
- Promotion: Jan Sudeikat in Kooperation mit Uni Hamburg
  Entwicklung selbstorganisierender verteilter Systeme am Beispiel von Multiagentensystemen

Eclipse-Codeassist

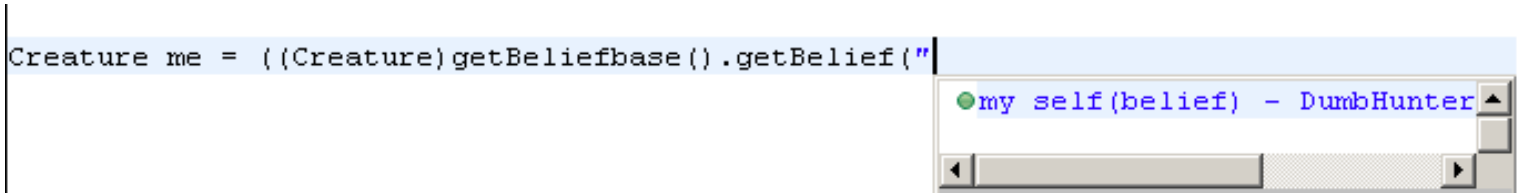
- Bachelorarbeit:

  Implementationsunterstützung für Webapplikationen und Multiagentensysteme am Beispiel von Struts, X3D und Jadex

  - XML-Editor unterstützt Referenzen auf Javaklassen und -Methoden



  - Java-Editor unterstützt Referenzen auf XML-Elemente



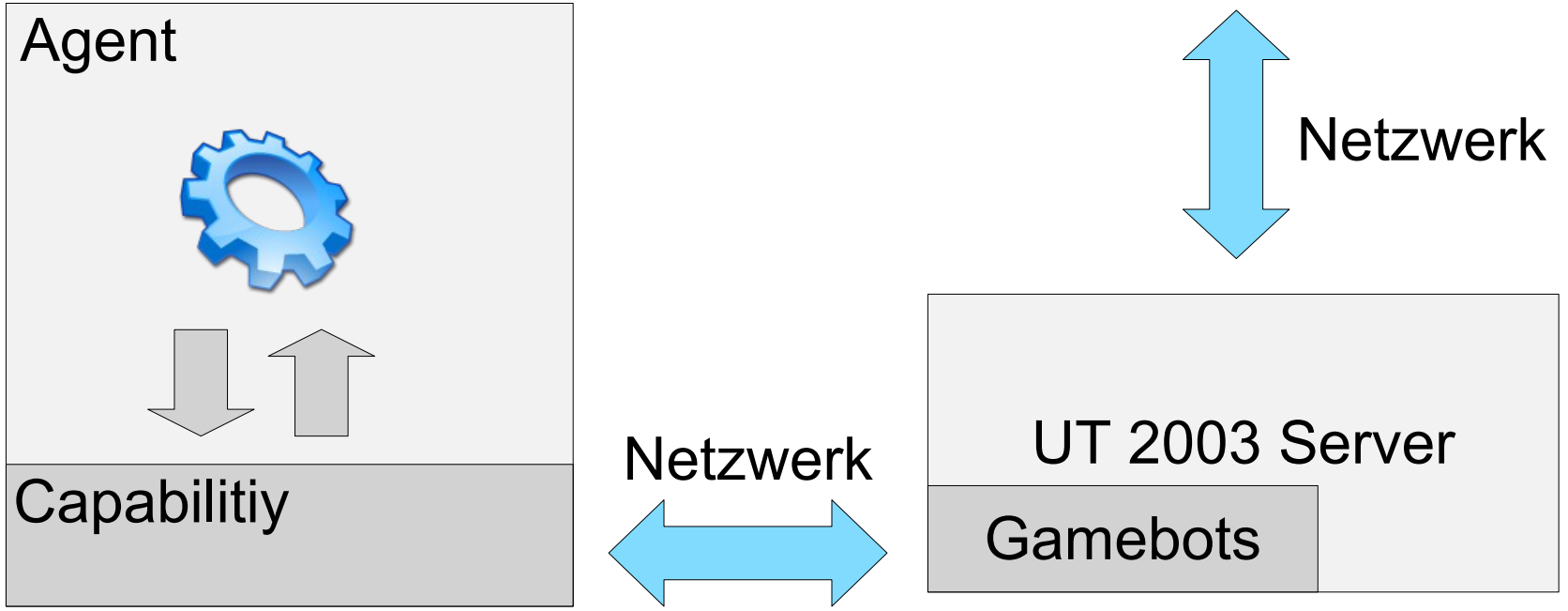
WP-Praktikum 2007: AOSE

- 4 Termine:
- Jadex Tutorial
- kleine Programmieraufgaben
- Aufgabe:
- Non-Playable Character (NPC) Intelligence in UT 2003
- Fragestellung: intelligentes Verhalten von NPCs
  - 1: Verhaltenmodule (Bewegung, Angreifen, Munition suchen, ...)
  - 2: Integration mittels Deliberation

Jadex plays UT

- Architektur:
  - Open Source system Gamebots (RIT)
  - Jadex Client Funktionalität: Capability (MMLab)

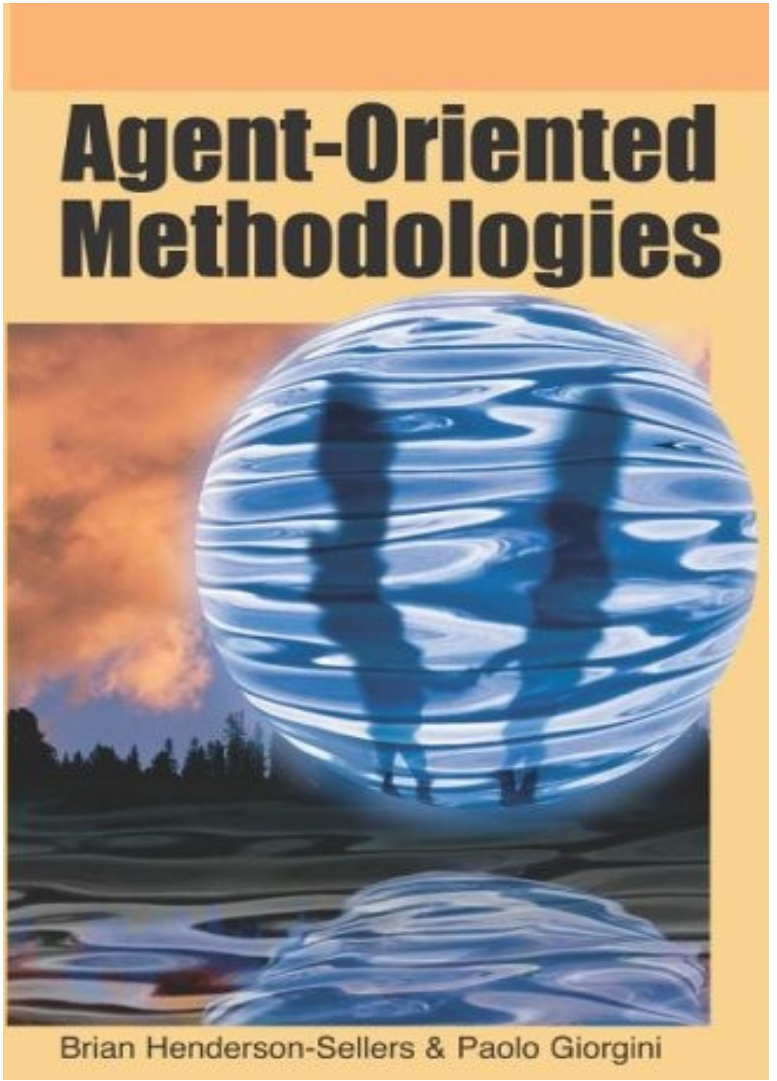
UT 2003 Client Anwendung



Selforganisation in AOSE ?

- High-Level Characteristics:
  - Adaptiveness
  - Flexibility
  - Scalability
  - Maintainability
  - ***Emergent behavior***
- *„…it is perhaps the last of this list that causes most concern...“*
- *„…emergent behavior has to be considered and planned for at the systems level using top-down analysis and design techniques…“*

little consideration
in AOSE Methodologies !

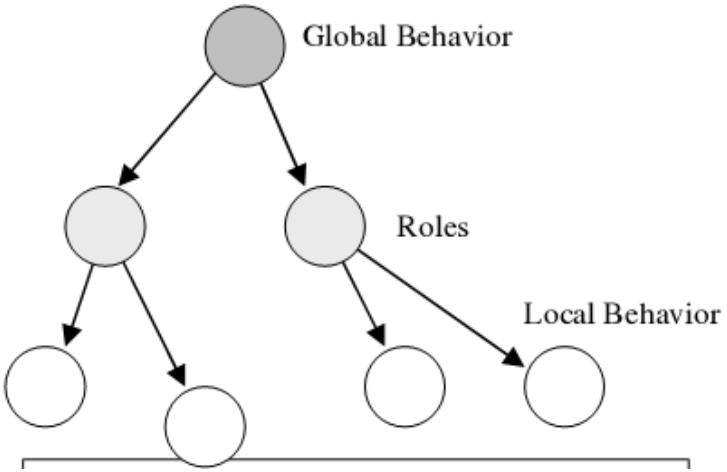


IDEA Group Publishing, 2005

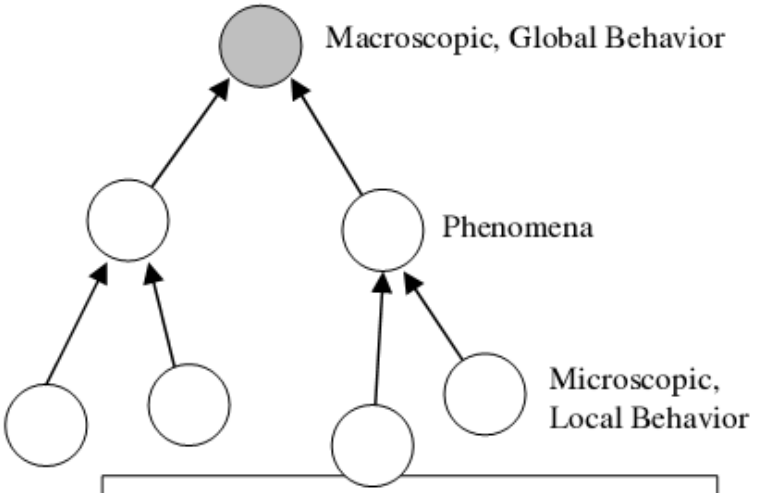
Complex Systems Engineering Paradoxon

- MAS sind komplexe adaptive Systeme:
- Organisations von Elementen
- Flexible Interaktions

Top-Down



Bottom-Up

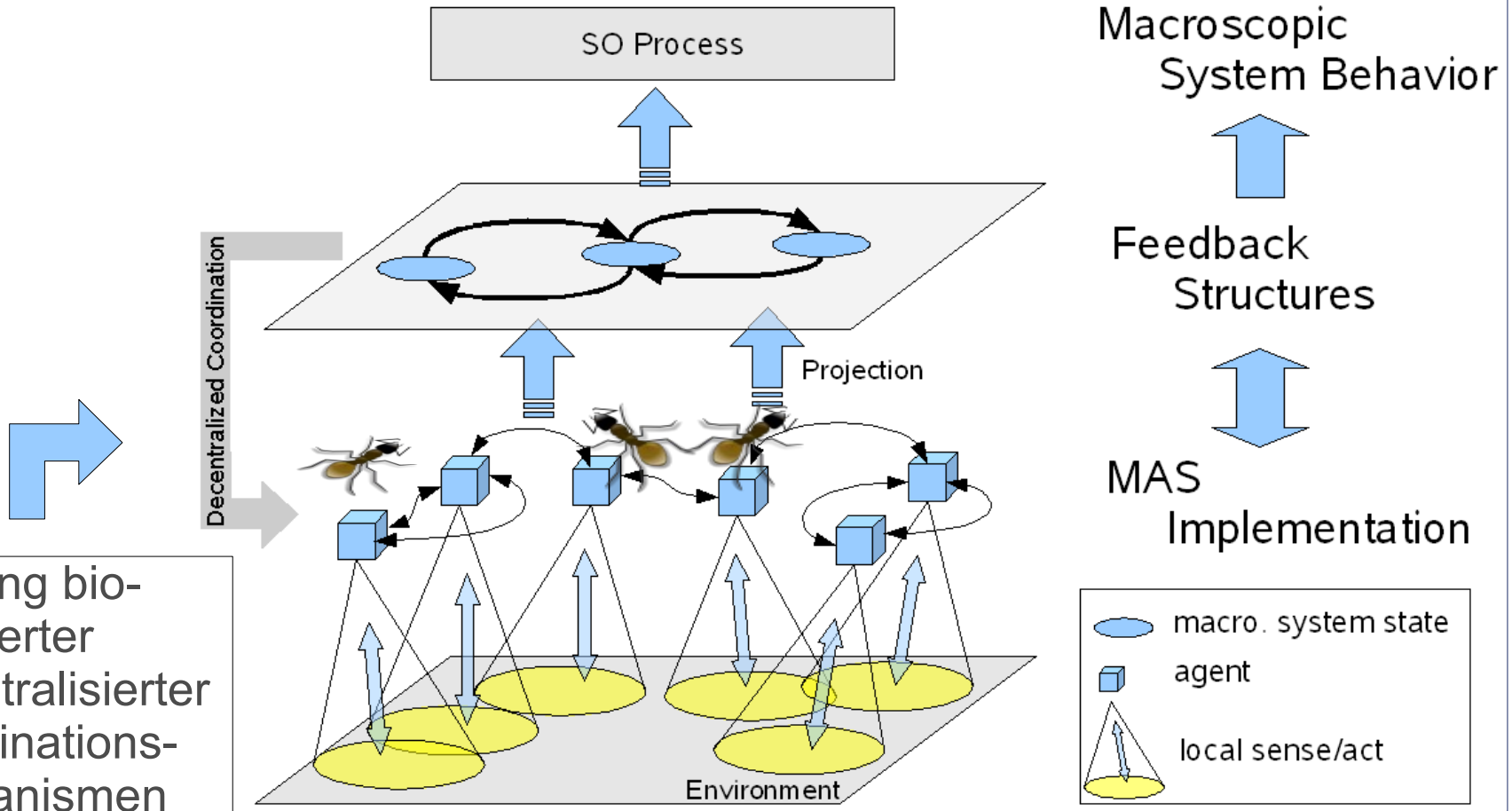


Paradoxon: mikroskopisches Agentendesign vs. makroskopische Anforderungen

Stand der Technik der SO-MAS-Entwicklung

- Nutzung dezentralisierter Koordinationsmechanismen

  Übertragung von biologischen, physikalischen oder soziologischen Mechanismen der Selbstorganisation



Nutzung bio-inspirierter dezentralisierter Koordinations-mechanismen

Selbstorganisations-Community

SO-MAS-basierte kommerzielle Lösungen:

- Schwärme unbemannter Flugobjekte
- Dokumentenclustering für Millionen
  verteilt anfallender Textdokumente
- Ad-hoc- und Sensornetze
- Logistik, Supply Chains

aus: http://www.newvectors.net/

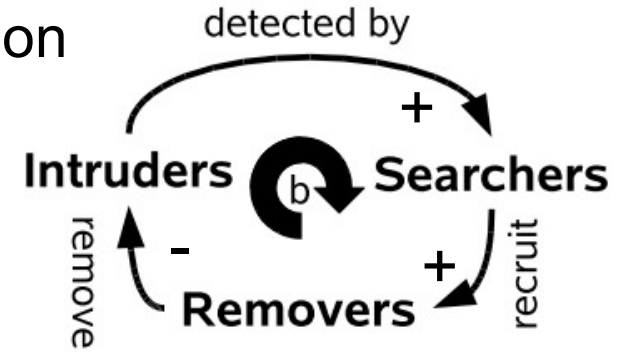
Verbundforschung:

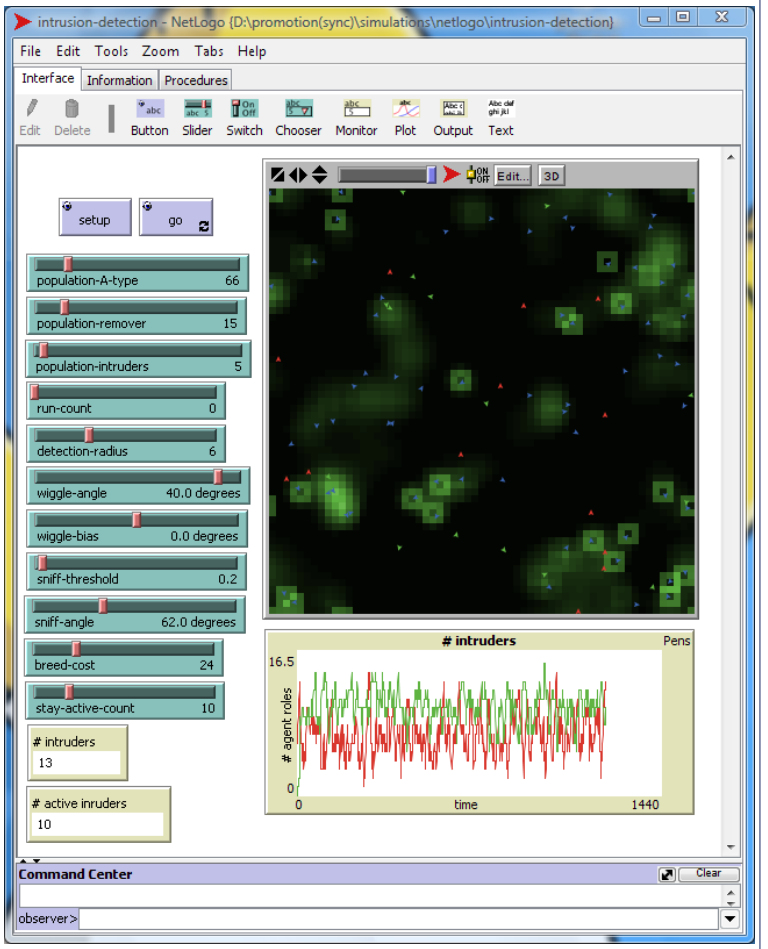
- Organic Computing Initiative (Schmeck, Uni Karlsruhe,
  Müller-Schloer, Uni Hannover)
  DFG-Schwerpunkt-Programm 1183
- Europäische Projekte an Universitäten und Forschungslabors

Fallstudie: Stigmergy-basierte Koordination

- Einfaches Intrusion Detection System
  - Immune System inspiriert
  - Agentenrollen
    - bösartige Intruders sollen
    - von Sentry Agenten erkannt werden, die Entfernung durch
    - Remover Agenten veranlassen.
- Stigmergy-basierte Koordination
    - Sentryagenten kommunizieren, wo Intruder sind.
    - Remover werden rekrutiert.
- CLD-Notation

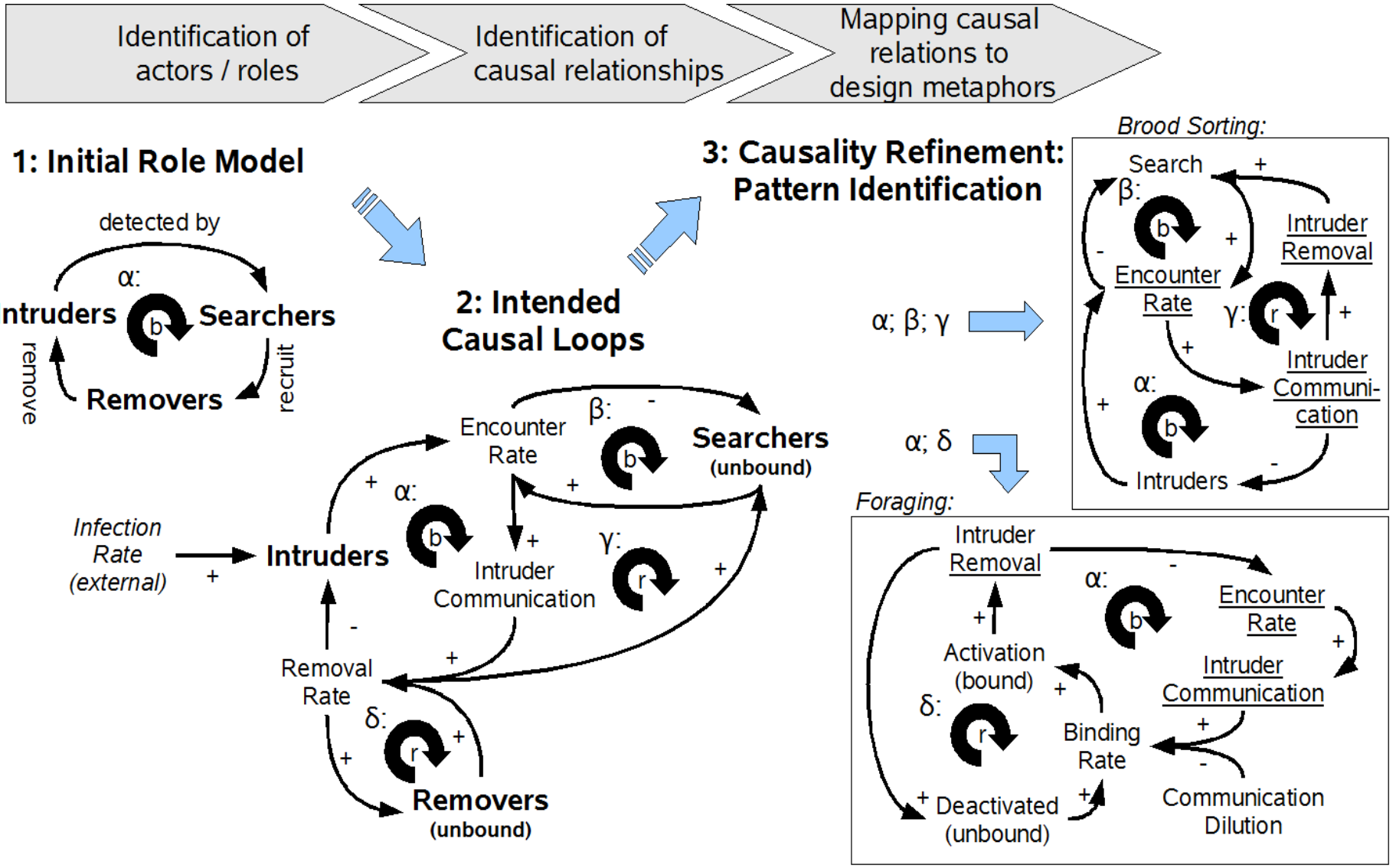
Sterman 2000:





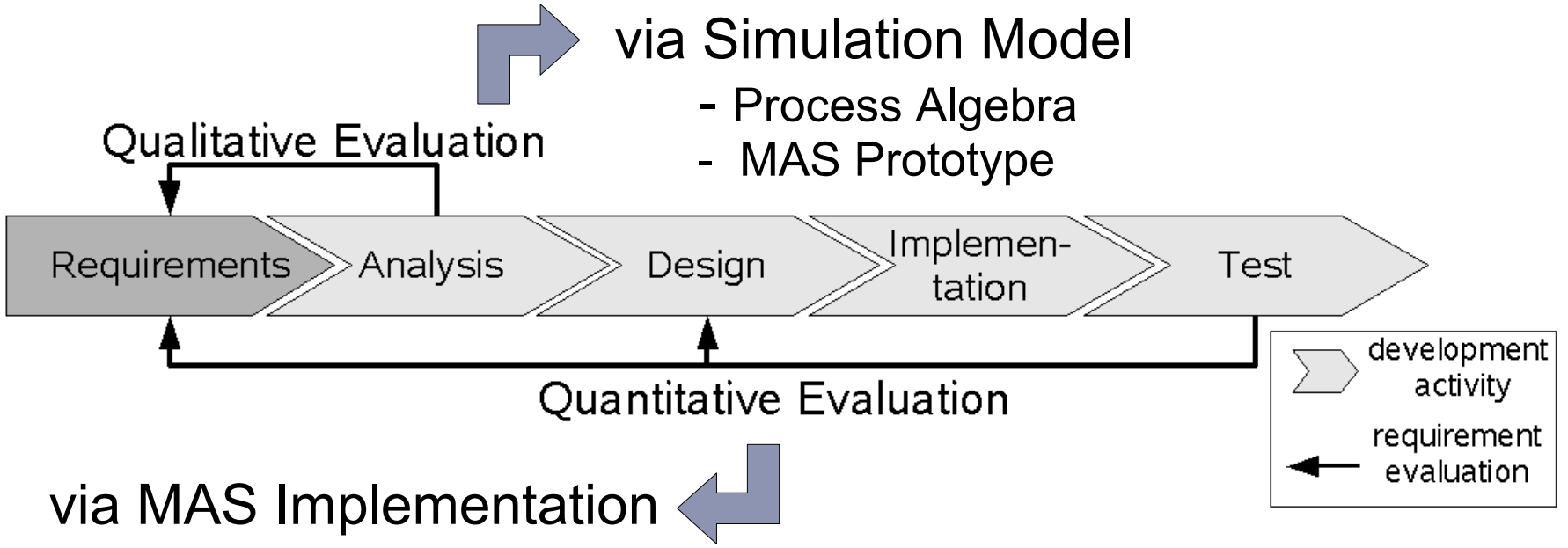
http://ccl.northwestern.edu/netlogo/

Fallstudie: empirische Analyse



Selbstorganisierende Systeme

1. Konstruktionprozess :



2. Diagrammatische Spezifikation der Dynamischen Pattern durch Causal Loop Diagramme:

- Katalog dynamischer Pattern *(Sudeikat, Renz: EEMMAS07)*
- Werkzeugunterstützung möglich (to be done)

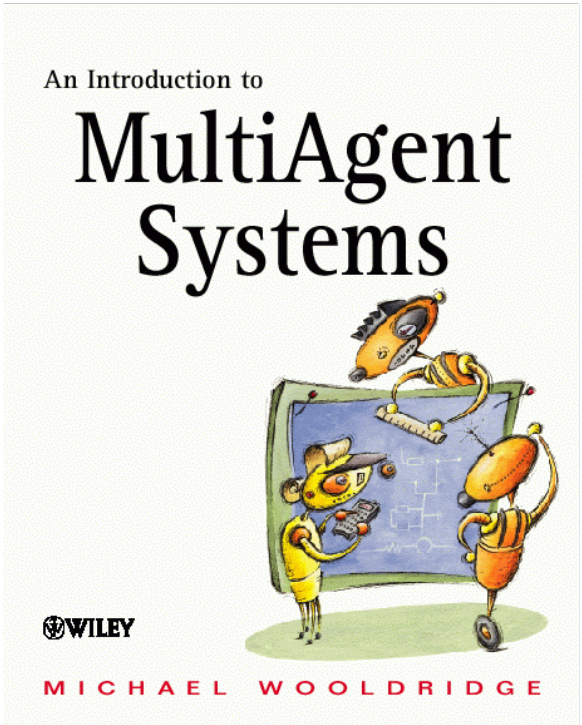
MAS-Bücher

Michael Wooldridge

An Introduction to Multiagent Systems

2002, John Wiley & Sons

ISBN 0 47149691X

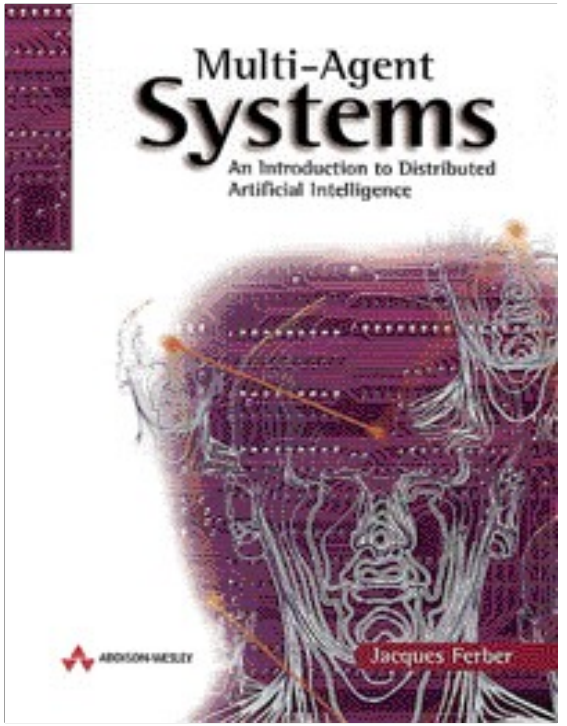


Jacques Ferber

Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence

1999, Addison Wesley Longman

ISBN 0-201-36048-9

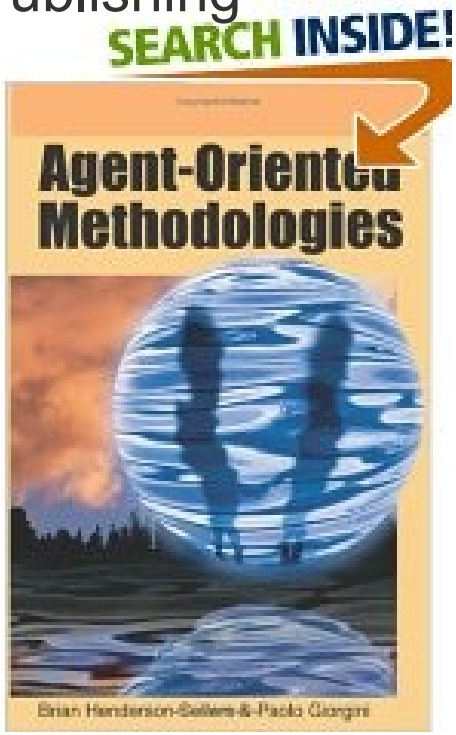


AOSE-Bücher

B. Henderson-Sellers & Paolo Giorgini

Agent-Oriented Methodologies

2006 IDEA Publishing



Ralf Jakob & Gerhard Weiss

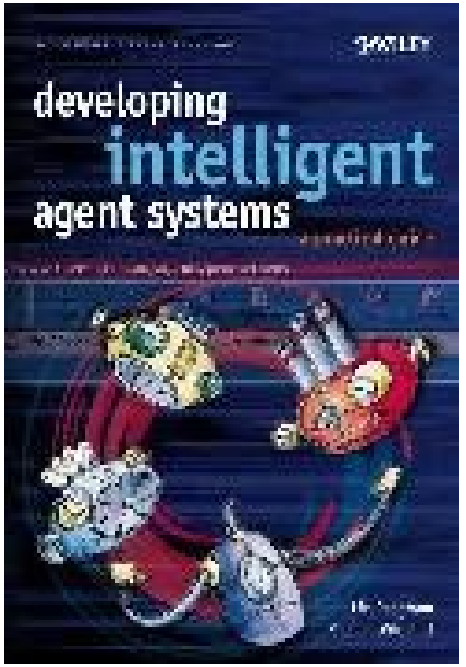
Agentenorientierte Softwareentwicklung

2005 Springer

Lin Padgham & Michael Winikoff

Developing Intelligent Agent Systems: A Practical Guide

2004, John Wiley



Webseiten

Web-Portale für Agenten

- Multiagent Systems: http://www.multiagent.com/
- AgentLink: http://www.agentlink.org/ [seit 2006 nicht mehr gepflegt]
- Agents Portal: http://aose.ift.ulaval.ca/
- http://www.aaai.org/AITopics/html/multi.html
- Journal-Webseite http://www.inderscience.com/browse/index.php?journalCODE=ijaose
- Jadex Homepage: http://vsis-www.informatik.uni-hamburg.de/projects/jadex/
- Netlogo Homepage: http://ccl.northwestern.edu/netlogo/
- API für UT-Engine: http://www.cs.rit.edu/~jdb/gamebots/

Schlusswort

- **Motivation: Praxistauglichkeit**
  - *Adaptive Systeme bauen*
- **Multiagentensysteme (MAS)**
  - *Open-Source-Projekt Jadex*
  - **Agentenorientierte Programmierung**
    - *Tool support, Unreal Tournament 2003*
  - **Forschung: Selbstorganisation in Software**
    - *allgemeines Prinzip unabhängig von MAS*
- **Schlussfolgerungen**
  - *wachsendes Interesse*