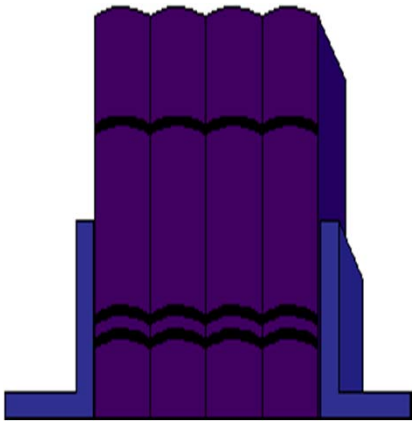
E Agentenbasierte Systemtechnologie

Ein Software-Agent ist ein Programm, das in einer vernetzten Umgebung selbständig Aufgaben durchführen kann.

Wie baut man Software auf, die nicht nur Aufträge ausführen kann, sondern auch „eigene“ Entscheidungen trifft (inkl. Ortsveränderung) ?



W. Brenner, R. Zarnekow, H. Wittig: „Intelligente Software-Agenten“, Springer, 1998

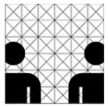
N.R. Jennings, M.J. Wooldridge (Hrsg.): „Agent Technology“, Springer 1998

P. Maes: „Designing Autonomous Agents“, MIT/Elsevier, 1994

J. Ferber (dt. St. Kirn): „Multiagentensysteme“, Addison-Wesley, 2001

Übersicht

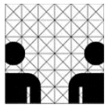
- **E1: Motivation und Begriffe**
  - **IT Trends und (mögliche) Folgen**
  - **Definitionen von Agent & Multiagentensystem**
  - **Eigenschaften von Agenten & MAS**
- E2: Wozu Agenten?
  - Taxonomien und Arten von Agenten
  - Anwendungsgebiete für MAS
- E3: Wie werden Agenten konstruiert?
  - BDI-Agentenarchitektur
  - Jadex-Agentenplattform



Motivation & Begriffe

Hintergrund: Fünf aktuelleTrends kennzeichnen die Entwicklung der I(K)T:

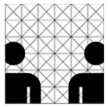
- ***Benutzerzentrierung*,**
- ***Vernetzung*,**
- ***Ubiquität*,**
- ***Intelligenz* und**
- ***Delegation***



Wohin führt uns das?

- **Delegation und Intelligenz erfordern, dass wir in der Lage sind Computersysteme zu bauen, die effektiv für uns / in unserem Namen handeln können**

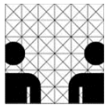
- **Daraus folgt:**
  - Die Computersysteme müssen unabhängig agieren
  - Die Computersysteme müssen unsere Interessen wahrnehmen können, während sie mit anderen Systemen oder Menschen interagieren



Historisch

- **Ursprung der Agententechnologie**
  - Informatik: Künstliche Intelligenz, verteilte Systeme, Software Engineering u.a.
  - andere Disziplinen: Philosophie, Psychologie, Biologie, Wirtschaftswissenschaften, …
  - eigenes Forschungsgebiet seit Anfang bis Mitte 1990er

- **Unterschiedliche Sichtweisen**
  - Multiagentensysteme als Problem Solver (VKI)
  - Agenten als Entwurfsmetapher („Computing as Interaction“)
  - Agenten als Assistenten
  - …



Beispiel: Agent für Büroeinkauf

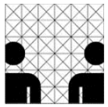
Der Einkäufer eines Unternehmens überträgt Teilaufgaben beim Einkauf von Büromaterial auf einen Software-Agenten:

- Anbieter von Büromaterial finden
- Sortiment von Anbietern auf gewünschte Materialien hin überprüfen
- Angebote beschaffen
- Einkäufer über Angebote informieren

Erweiterte Aufgaben:

- Markt beobachten, Preisbewegungen registrieren
- Kaufzeitpunkte vorschlagen
- Lagerbestände im Unternehmen verfolgen
- selbständig für Nachschub sorgen

„*Principal-Agent-Theorie*“ (dt. Agenturtheorie), BWL 1976



Definition: Agent (1)

Definition

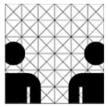
*"Agents are software entities that **assist** people and act on their behalf."*
(P. Maes, 1994)

Charakteristik

*Agents are situated in an environment, act **autonomously**, and are able to **sense** and to **react** to changes.*

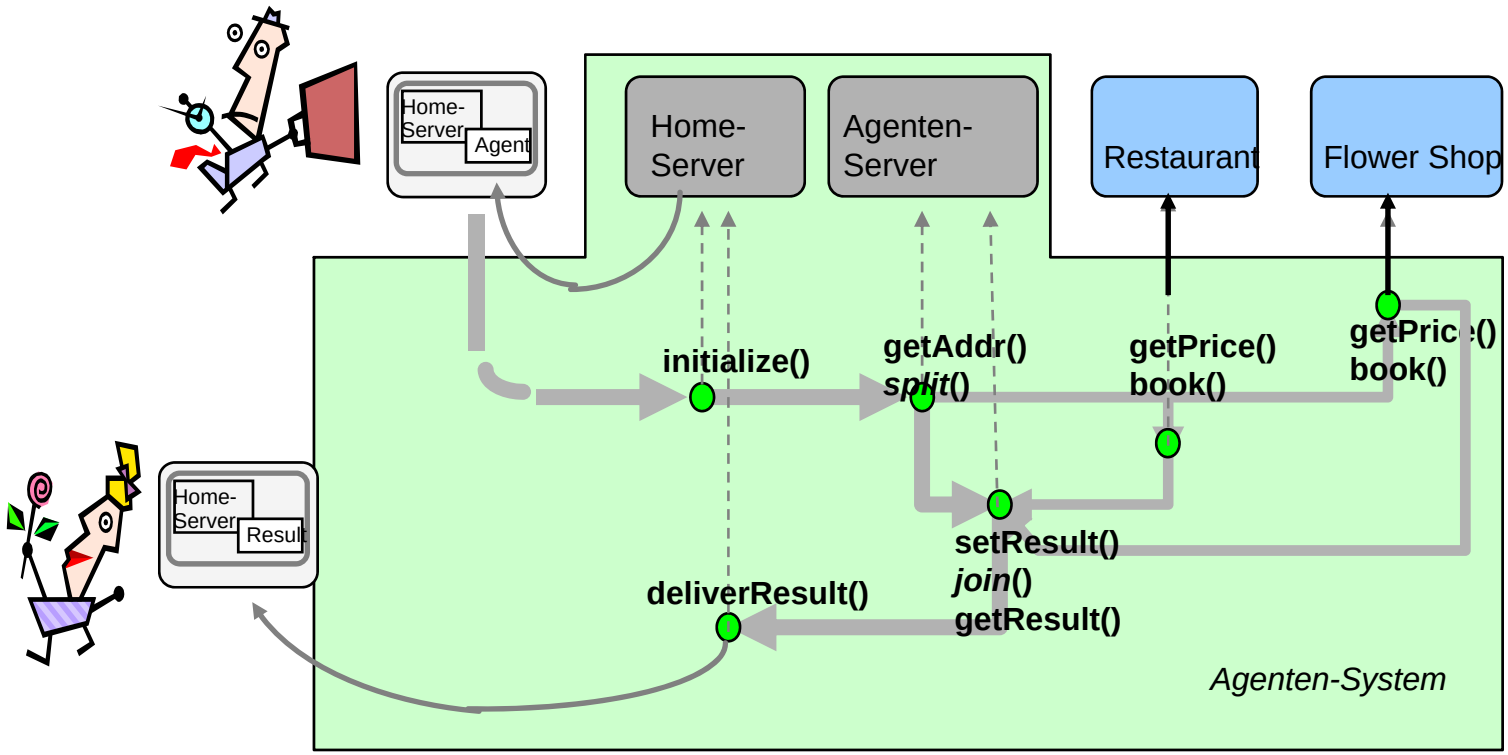
Anwendungsgebiet: *Mobile Agenten für „E-Commerce"*

- E-Commerce als Anwendungsgebiete für Agententechnologie
- besonders interessantes Teilgebiet: elektronische *Verhandlungsführung*
- Vorteile der Verhandlungsführung ergeben sich insbesondere bei deren *Automatisierung* (z.B. kein Zeitverlust, "Embarrassment"...)



„Autonome mobile Agenten“: Spezifikation und Ausführung nebenläufiger verteilter Aktivitäten

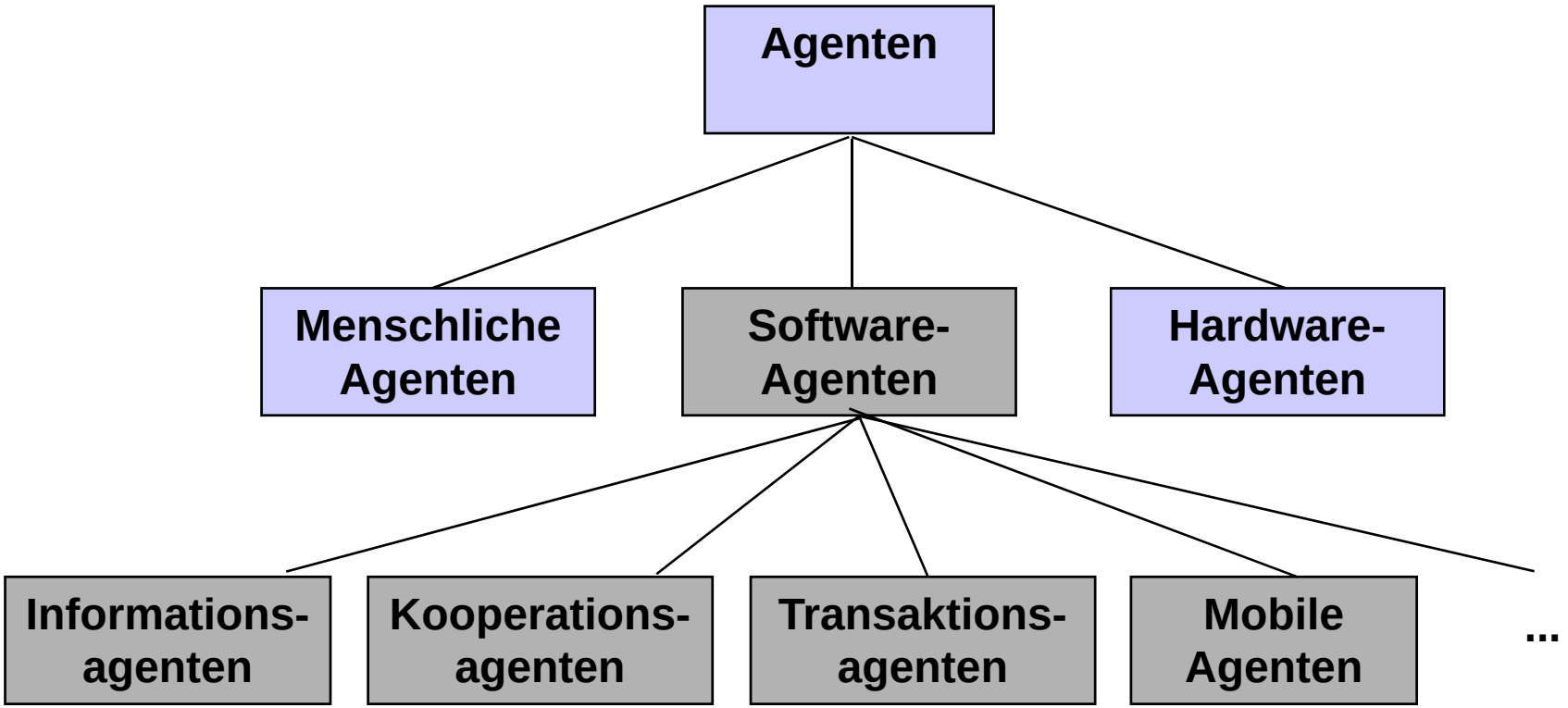
Beispiel: *Abendessen im Restaurant mit Blumen*



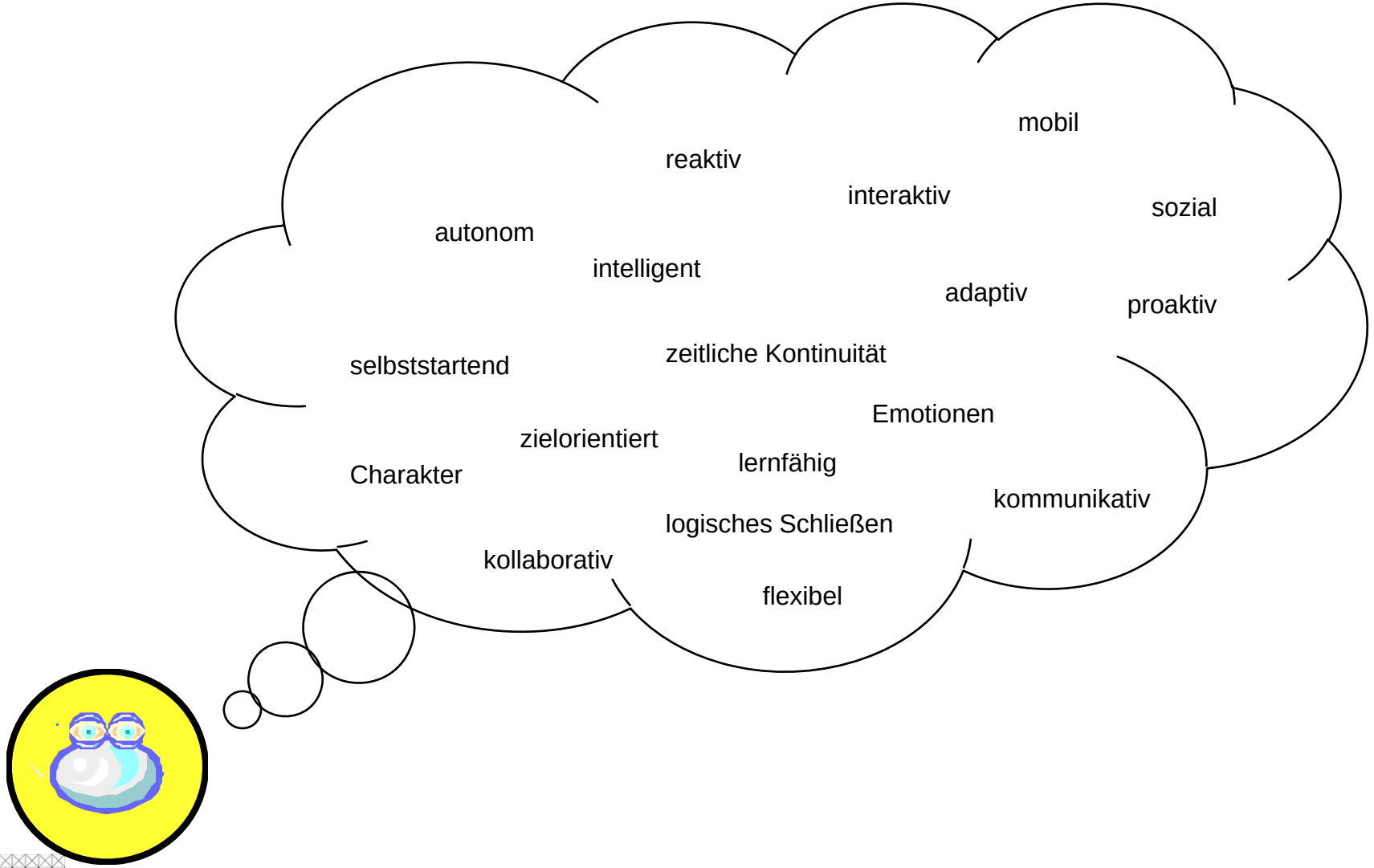
Kategorien von Agenten

Hier von Interesse: Software-Agenten

„Agent“ ist Metapher für Eigenschaften menschlicher Agenten



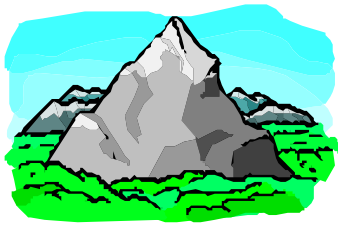
Eigenschaften von Agenten (1)



Eigenschaften von Agenten (2)

- **Autonomie**
  - Der Agent entscheidet „selbstständig“ anhand bestimmter Kriterien über seine nächste Aktion

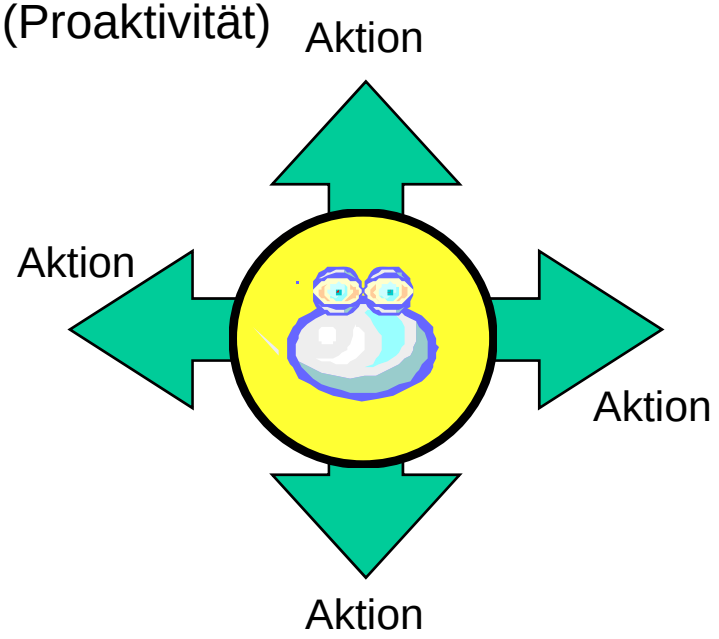
- **Beispiele:**
  - Orientierung auf einer Landkarte:
    Der Agent entscheidet anhand von bestimmten Kriterien (Zeit, Position, Helligkeit,..), welchen Weg er nutzt



  - Buchung einer Reise:
    Der Agent entscheidet anhand der Präferenzen seines Prinzipals (Reiseziel, Transportmittel, ...), welche Reise er bucht

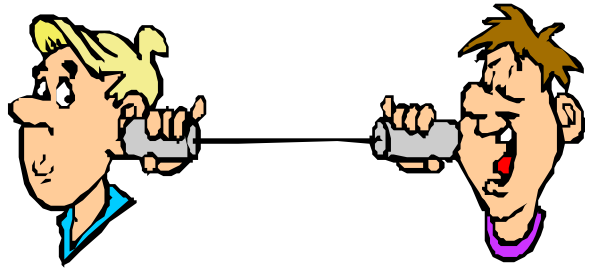
Eigenschaften von Agenten (3)

- **Reaktivität / Proaktivität**
  - **Der Agent reagiert auf Änderungen seiner Umgebung**
  - **Der Agent verändert seine Umgebung aufgrund von internen Parametern oder seines aktuellen Zustandes**
  - **Voraussetzungen:**
    - Existenz von Sensoren und Regeln (Reaktivität)
    - Existenz eines Zieles für den Agenten (Proaktivität)

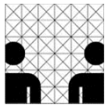


Eigenschaften von Agenten (4)

- **Kommunikation**
  - Der Agent kommuniziert mit seiner Umgebung (Dienste, andere Agenten, Agentensysteme)

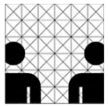


  - **Voraussetzungen:**
    - gemeinsame Sprache (ACL)
    - gemeinsame Interaktionsprotokolle
    - gemeinsame Weltsicht (Ontologie)



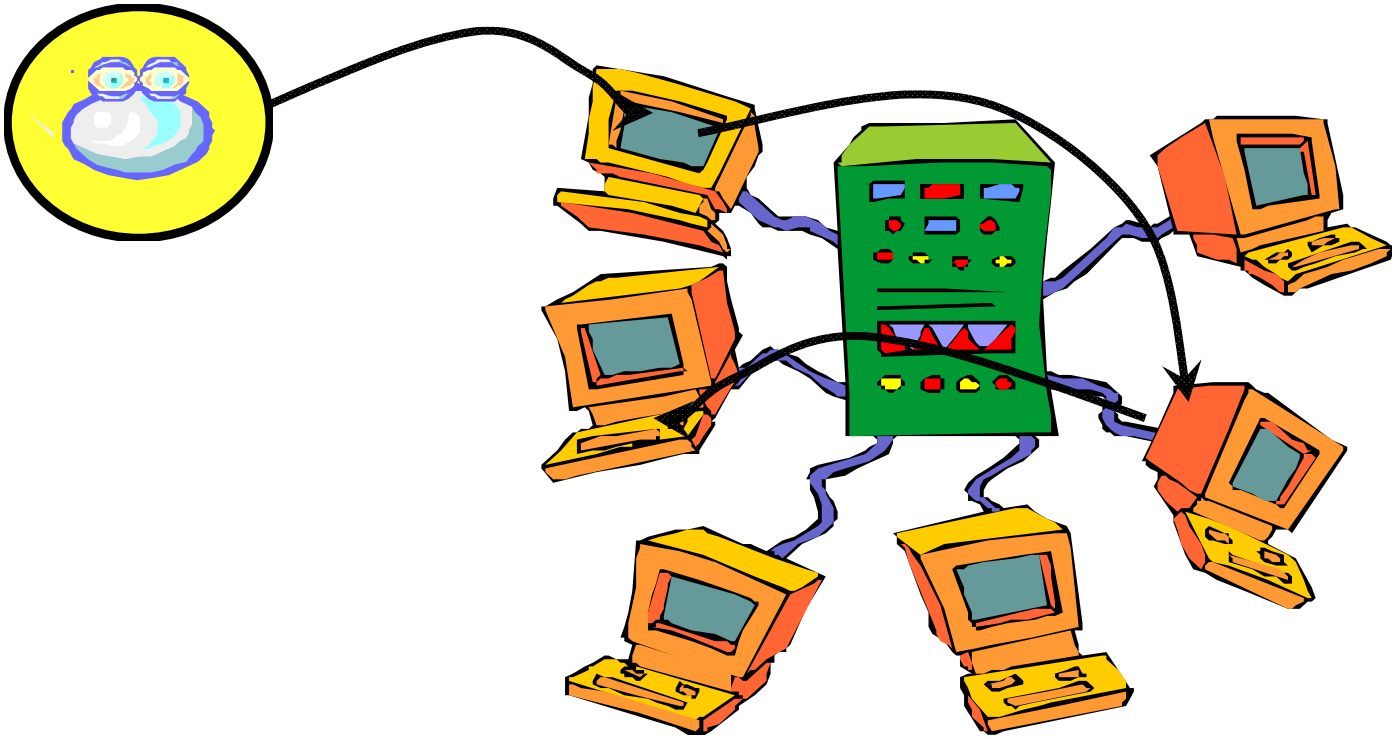
Eigenschaften von Agenten (5)

- **Lernfähigkeit / Intelligenz**
  - **Der Agent kann auf verschieden Weisen lernen:**
    - Übernahme von Wissen
    - Anweisungen, Beispiele
    - Bewertung des Erfolges / der Güte einer ausgeführten Aktion
      - intern - durch Regeln / Funktionen
      - extern - durch einen Benutzer / System
  - **Voraussetzung:**
    - Existenz eines Bausteins im Agenten zur Wissensrepräsentation und Auswertung
- **Interaktivität**
  - mit einem Benutzer durch Benutzerschnittstelle
  - Beispiele: SMS, WAP, eMail, GUI, Spracherkennung usw.



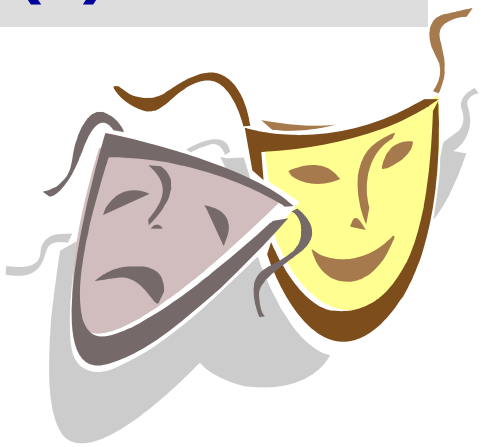
Eigenschaften von Agenten (6)

- **Mobilität**
  - **Der Agent kann von einer Plattform auf eine andere Plattform migrieren**
  - **Voraussetzungen:**
    - Serialisierbarkeit des Agentencodes
    - Die Zielplattform kann den Code des Agenten interpretieren



Eigenschaften von Agenten (7)

- **Emotionen / Charakter**

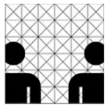


  - **Der Agent kann verschiedene „emotionale“ Zustände einnehmen,** z.B.
    - Erstaunen, Furcht, Freude, Enttäuschung, …
    - beeinflusst durch die Umgebung und z.B. interne Erwartungen
    - Vorteil: Anpassen von Verhalten, Glaubwürdigkeit
  - **Je nach Charakter unterschiedliche Verhaltensweisen**
    - „mutiger“ Agent vs. „vorsichtiger“ Agent
    - z.B. über emotionale Zustände gesteuert

Definition: Agent (2)

- **Allgemeine Beschreibung als Ausgangspunkt**
  - **“An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.”** (Jennings & Wooldridge, 1998, S. 4)

- **Konkrete Definition notwendiger / hinreichender Eigenschaften** (Wooldridge & Jennings 1995)
  - „**Weak Notion of Agency“:** Autonomie, Proaktivität, Reaktivität, soziale Fähigkeiten
  - „**Strong Notion of Agency**“: zusätzlich mentalistische Konzepte zur Beschreibung und Realisierung
  - sollte als graduelles Maß verstanden werden

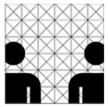


Umgebung

- **Definition: “An environment provides the conditions under which an entity (agent or object) exists”** (Odell et al. 2003)

- **Eigenschaften von Umgebungen** (nach Russel & Norvig)
  - Beobachtbarkeit (partiell vs. vollständig)
  - Determinismus (deterministisch vs. stochastisch vs. strategisch)
  - zeitliche (Un)abhängigkeit (episodisch vs. sequentiell)
  - Dynamik (dynamisch vs. statisch)
  - Kontinuität (kontinuierlich vs. diskret, räumlich vs. zeitlich)
  - Multiagenteneigenschaft

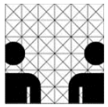
- **Agentenumgebung als Teil der Anwendung**
  - Umgebung übernimmt Aufgaben (Weyns & Holvoet 2006)
  - Ressourcenverwaltung, Kommunikationsverwaltung, aktive Umgebungsprozesse, …



Multiagentensysteme

- **Definition: “A multi-agent system is one that consists of a number of agents, which interact with one another [...].”** (Wooldridge 2001, S. 3)

- **Charakteristische Eigenschaften**
  (Jennings et al. 1998)
  - eingeschränkte Weltsicht der einzelnen Agenten
  - keine globale Systemkontrolle
  - dezentrale Verteilung der Daten im System
  - asynchrone Berechnung



Mensch-Agenten-Kommunikation

Bisher:

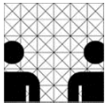
Mensch-Rechner-Kommunikation ist meist eher an *Rechner* angepasst (z.B. Kommandosprache, Parametereingabe unter Rechnerkontrolle)

Ziel:

Mensch-Agenten-Kommunikation mit *menschlichen Kommunikationsformen*:

- **Begriffssystem der natürlichen Sprache**
- **flexibler Satzbau**
- **Dialogkontext**
- **Hintergrundwissen für Sprachverständnis**
- **akustische Eingabe**
- **Gestik**

} **Sprachtechnologie**

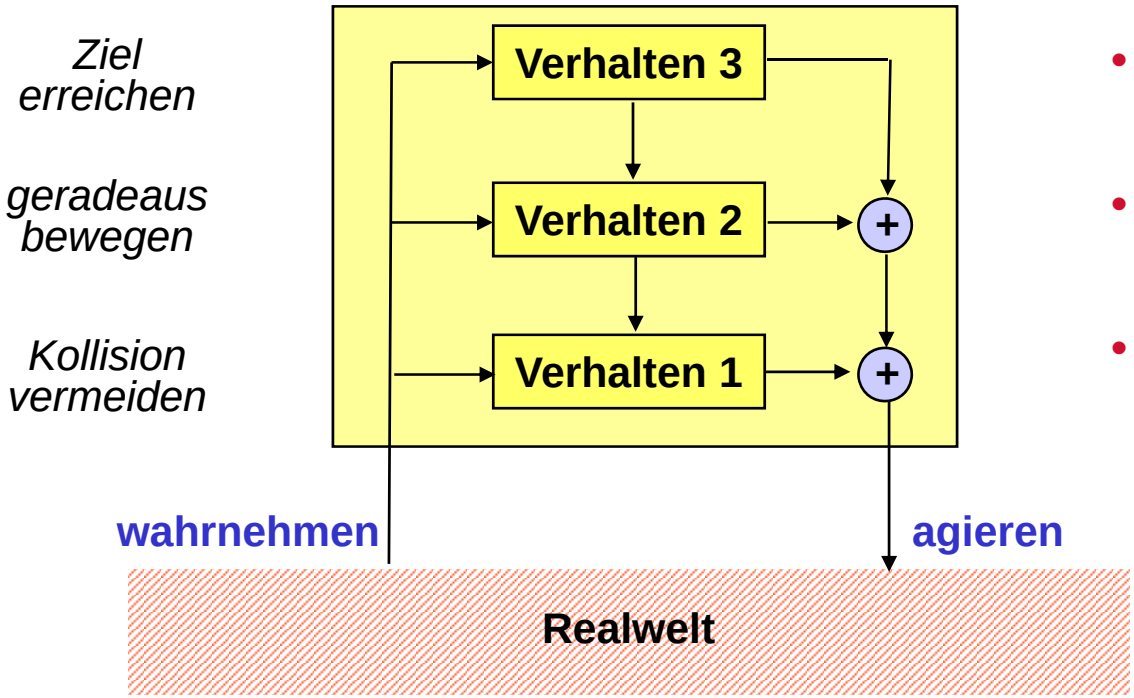


Reaktive Agenten

- **Agenten reagieren durch Verhaltensmodule unmittelbar auf ihre Umgebung**
- **kein internes Weltmodell**

Subsumptionsarchitektur nach Brooks (1986):

Beispiel:



- **Jedes Verhalten ist (fast) eigenständiger Prozess**
- **Verhalten i „subsumiert“ (umfasst) Verhalten i-1**
- **Realwelt übernimmt Rolle eines „Weltmodells“**

Kooperierende Agenten

Wie können individuell motivierte, unabhängig agierende Software-Agenten zur Erreichung von Zielen *kooperieren*?

1. Agenten kooperieren durch komplementäre Aufgabenverteilung

Gemeinsames Ziel wird (implizit) erreicht, wenn jeder Agent seine Aufgabe ausführt.

Beispiele: Fußball spielen, Krankenhauslogistik, Blackboard-Systeme etc.

2. Agenten verwenden Kooperationsmethoden

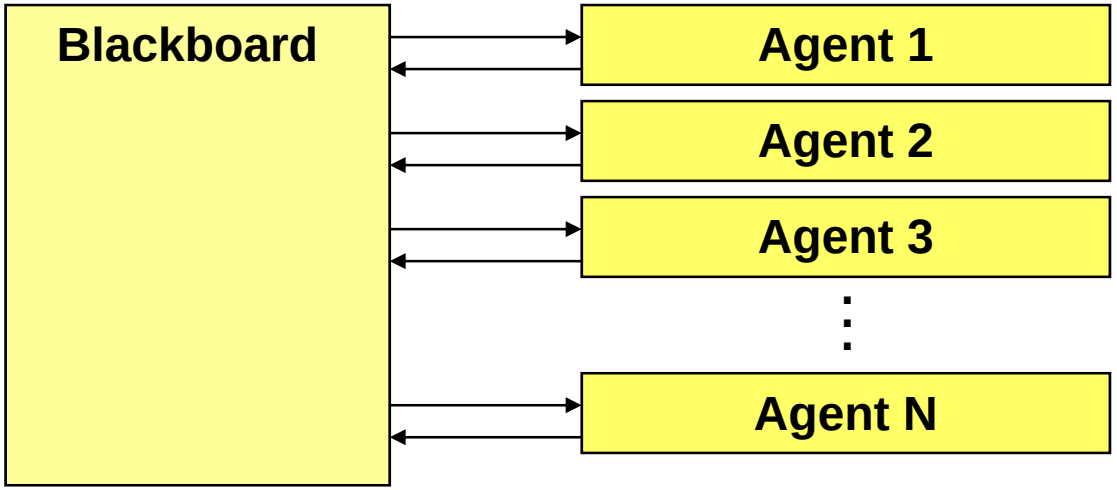
(Explizite) Kooperationsmethoden erlauben es, Multi-Agentenpläne zu schaffen und durchzuführen.

Kooperationsmethoden sind abstrakte, für eine breite Klasse von Anwendungen geeignete Schemata.

Beispiel: Kontraktnetze (s.u.)

Blackboard-Systeme

- **Kommunikation und Koordination über eine gemeinsame Kommunikationstafel (*Blackboard*)**
- **Agent liest Eingangsdaten vom Blackboard und schreibt Ergebnisse auf das Blackboard**
- **Agent wird tätig, wenn Eingangsdaten bereitstehen**



Aufbau eines Blackboard-Agenten

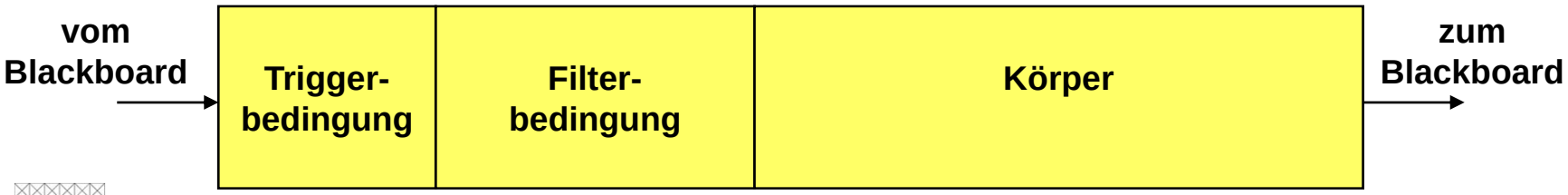
Ein Blackboard-Agent hat meist die Rolle eines Spezialisten:

- **spezielles Wissen**
- **spezielle Fähigkeit**
- **ein einziges Ziel**
- **nur an seinen Eingabedaten interessiert**

Ein Blackboard-Agent wird aktiv, wenn der Inhalt des Blackboards die Triggerbedingung des Agenten erfüllt (z.B. *Sind neue Daten in einer bestimmten Blackboard-Abteilung?*).

Ein getriggerter Agent prüft das Blackboard mit seiner Filterbedingung (z.B. *Sind die neuen Daten für seine Aufgabe geeignet?*) und liest seine Eingabedaten.

Im Körper des Agenten werden die Eingabedaten bearbeitet. Das Ergebnis wird auf dem Blackboard abgelegt.



Nachrichtenaustausch zwischen Agenten

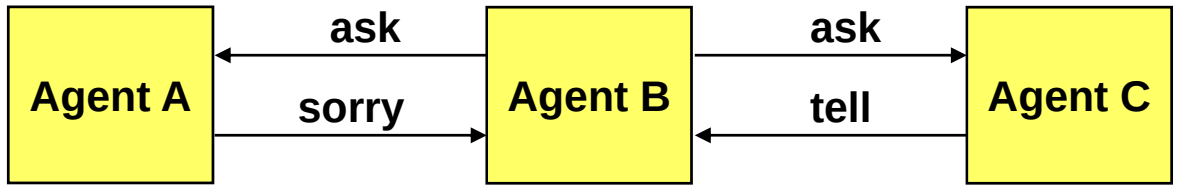
Kommunikationstheorie („Sprechakttheorie“) bietet nützliche Abstraktionen für Nachrichtenaustausch:

Nachrichtenaustausch = zielgerichtete Aktion

„Sprechakt“	**Akt des Nachrichtenaustausches**
„Sprechakttyp“	**Unterscheidung von Sprechakten nach Art des Ziels (z.B. fragen, antworten, anweisen, informieren, ...)**

Ein Kommunikationsprotokoll legt Regeln fest, nach denen Sprechakte in einem Dialog aufeinander folgen dürfen.

Beispiel:



Agentenkommunikation mit KQML

KQML (‚Knowledge Query and Manipulation Language') ist verbreitete Sprache für Nachrichtenaustausch in Multi-Agentensystemen.

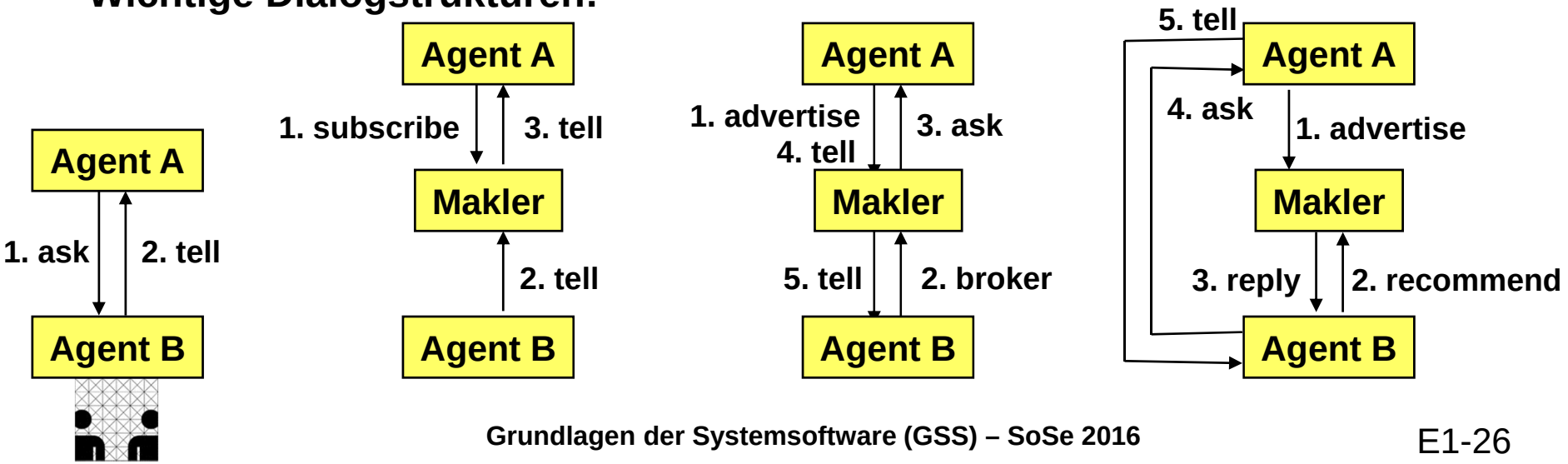
Struktur einer KQML-Nachricht:

```
(<Sprechakttyp>
:content <Aussage/Sprechakt>
:sender <Name>
:receiver <Name>
:language <text>
:ontology <text>)
```

Beispiel:

```
(ask-one
:content (PRICE IBM ?price)
:sender news-server
:receiver stock-server
:language LPROLOG
:ontology NYSE-TICKS)
```

Wichtige Dialogstrukturen:



Kooperation zwischen Agenten

Sprechakte **können die Kooperation zwischen Agenten zum Lösen von gemeinsamen Aufgaben strukturieren.**

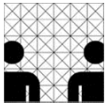
Sprechakttypen für Kooperationen:

- ***propose***
- ***accept***
- ***reject***
- ***refine***
- ***modify***
- ***inform***
- ***query***

Sprechaktobjekte für Kooperationen:

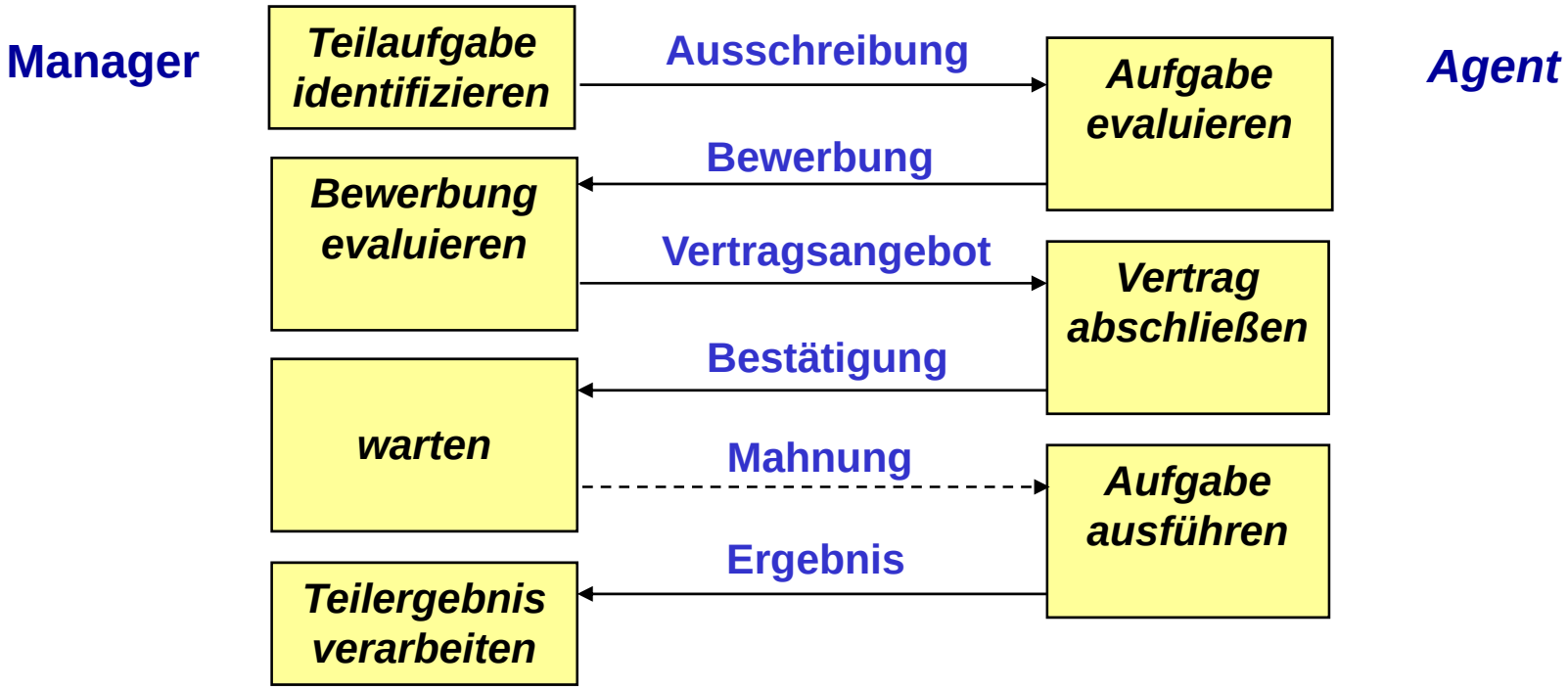
- ***goal***
- ***goal breakdown***
- ***plan***
- ***commitment***
- ***task assignment***
- ***resource allocation***
- ***schedule***

Kooperationsprotokolle **legen geeignete Abläufe für den Austausch von (Kooperations-) Sprechakten fest.**



Kontraktnetz-Protokoll

Schematisierter Verhandlungsablauf zwischen Manager und Agenten zur Vergabe von Teilaufgaben in Kontraktnetzen:

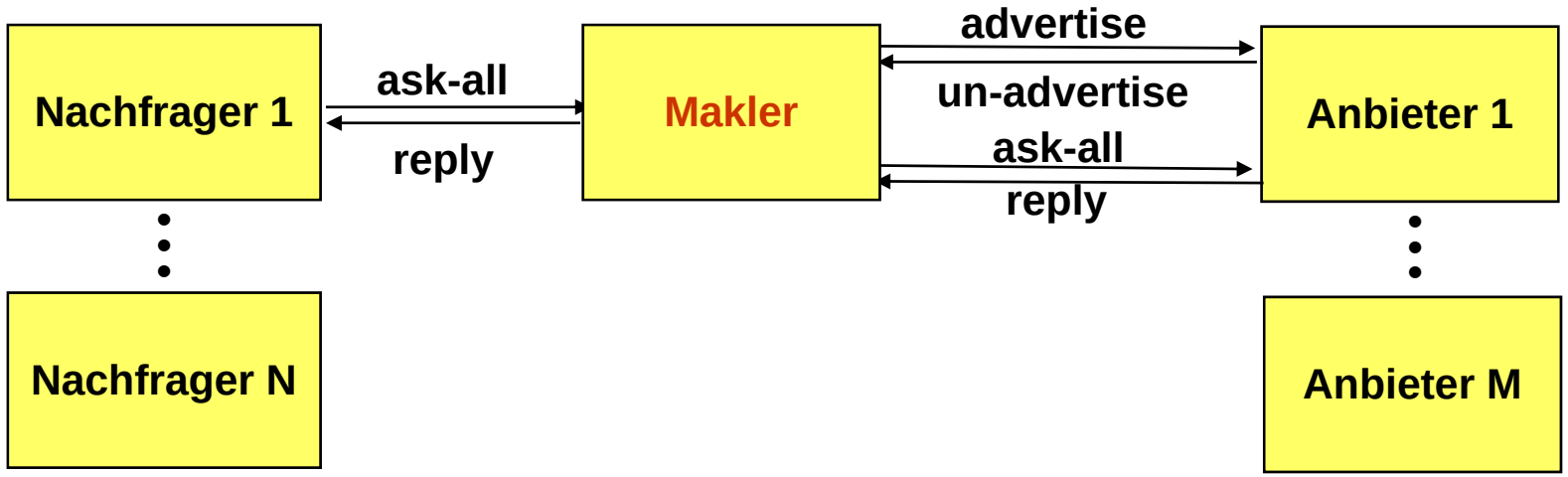


<u>Anm.:</u> Jeder Agent kann seinerseits Teilaufgaben als Manager vergeben

Makler-Protokolle

Ein Makler (Broker) ist Vermittler zwischen Nachfragern und Anbietern von Diensten oder Problemlösungen.

- **Nachfrager (*Requester*)** behandelt Makler wie Problemlöser
- **Anbieter (*Server*)** behandelt Makler wie Problemsteller



Lernende Agenten

Beispiel: Maxims (Maes et al. 94)

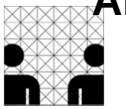
- **Agent lernt, wie ein Benutzer mit seiner Email umgeht**
- **Agent macht Vorschläge: Nachricht lesen, löschen, priorisieren, kategorisieren, archivieren, automatisch beantworten, ...**

Agent kann mehrere Lernstrategien verfolgen:

- **Lernen durch Imitieren des Benutzers**
- **Lernen durch Lehrbeispiele des Benutzers**
- **Lernen durch Benutzerkritik**
- **Lernen von anderen Agenten**

Grundidee:

- **eMail-Aktionen des Benutzers anhand von Situations-Aktions-Beschreibungen speichern**
- **Situationen durch zahlreiche Merkmale beschreiben**
  (Sender, Empfänger, Kurzüberschrift, Länge, Schlüsselworte, ...)
- **Relevanz von Merkmalen lernen und gewichten**
- **Aktionen für neue Situationen durch Vergleich mit gespeicherten Situations-Aktions-Paaren ermitteln (→ fallbasiertes Schließen)**



Email-Aktionen lernen

Verstärkungslernen:

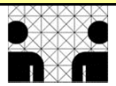
Relevanz von Merkmalswerten erhöhen, die zu derselben Aktion führen

Fallbasiertes Schließen:

Aktion desjenigen gespeicherten Falles F´ vorschlagen, dessen Merkmale mit dem aktuellen Fall F am besten übereinstimmen:

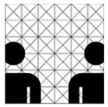
Abstandsmaß $D(F,F') = \sum_i r_i \cdot d(m_i, m_i')$

Merkmal m_1 (Sender)	Relevanz r_1	Merkmal m_2 (Überschrift)	Relevanz r_2	Aktion
Cutter Corporation	7	E-Commerce Investment Strategies	3	delete unread
FB-Leiterin	3	Vorstands-Sitzung 14.12.	7	read
Cutter Corporation	7	Your Best Buy Ever	3	delete unread
FB-Leiterin	3	Vorstands-Sitzung 2.2.	7	read
FB-Leiterin	4	FB-Frauentreffen 9.1.	6	archive
Ausschreibungsdienst	4	Mitteilung Nr. 42	6	archive
Planer	3	Vorstands-Sitzung 11.3.	7	read
Planer	3	Vorstands-Sitzung 11.3. fällt aus	7	read



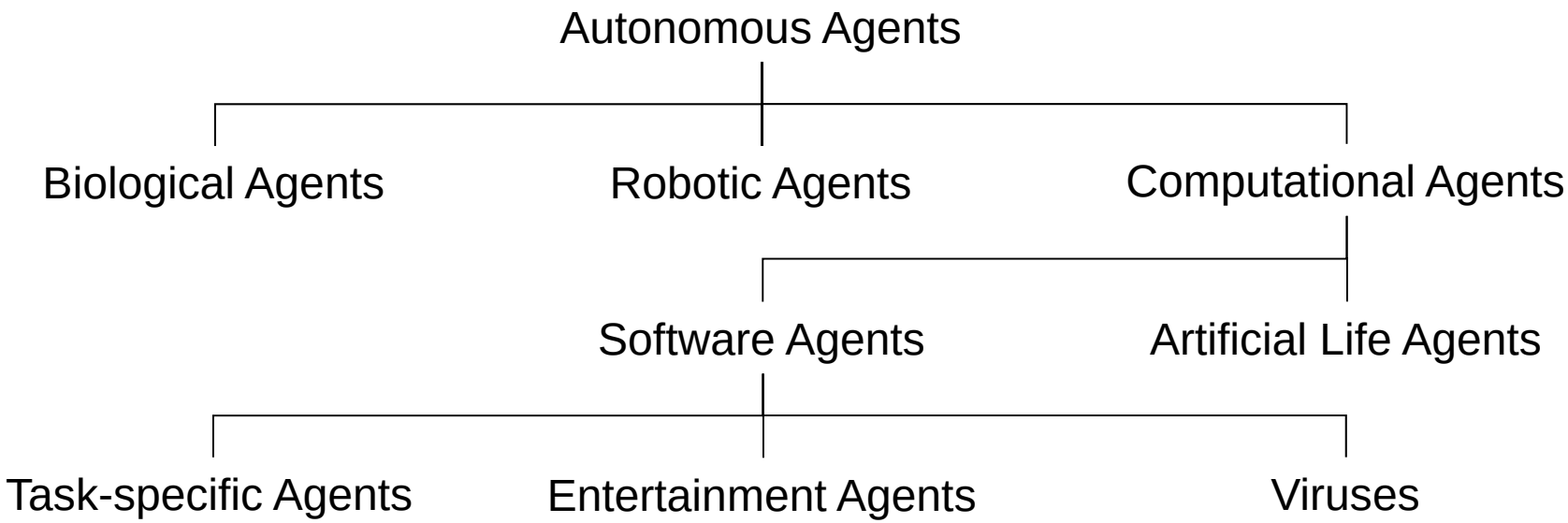
Übersicht

- E1: Motivation und Begriffe
  - IT Trends und (mögliche) Folgen
  - Definitionen von Agent & Multiagentensystem
  - Eigenschaften von Agenten & MAS
- **E2: Wozu Agenten?**
  - **Taxonomien und Arten von Agenten**
  - **Anwendungsgebiete für MAS**
- E3: Wie werden Agenten konstruiert?
  - BDI-Agentenarchitektur
  - Jadex-Agentenplattform



Taxonomien von Agenten (1)

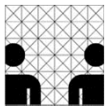
- **Taxonomie nach Franklin & Graesser**



Taxonomien von Agenten (2)

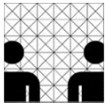
Einordnung nach Aufgaben

- **Informationsagenten**
  - **Informationssuche, -filterung, -aufbereitung**
- **Interface Agenten**
  - **Benutzer bei der Arbeit unterstützen**
  - **Vom Benutzer lernen**
- **Unterstützungsagenten**
  - **Automatisierung bzw. Unterstützung von Geschäftsaufgaben**
  - **Geschäftszweck steht im Vordergrund**
- **Mobile Agenten**
  - **Verarbeitung auf andere Rechner übertragen**
- **Problemlöseagenten**
  - **Dekomposition des Problemraums**
  - **Repräsentation unterschiedlicher Zielkriterien**
- **Simulation / Robotersteuerung**
  - **Agenten als abstraktes Modellierungskonzept**
  - **Kontrolle unabhängiger Einheiten**



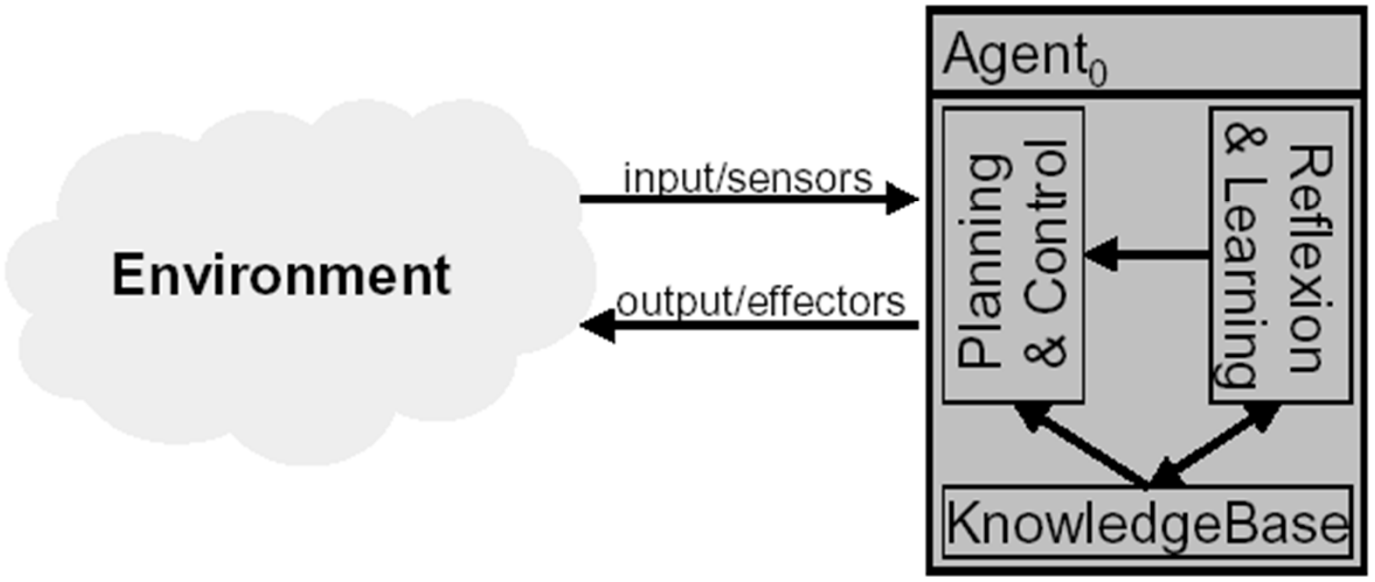
Einige Agenten-Anwendungen

- **Industrielle Anwendungen**
  - Prozesssteuerung
  - Produktion
  - Luftverkehrskontrolle
- **Kommerzielle Anwendungen**
  - Informations-Management
  - E-Commerce
  - Business Process Management, Logistik etc.
- **Medizinische Anwendungen**
  - Patientenüberwachung
  - Gesundheitsfürsorge
- **Unterhaltung**
  - Spiele
  - Interaktives Theater und Kino

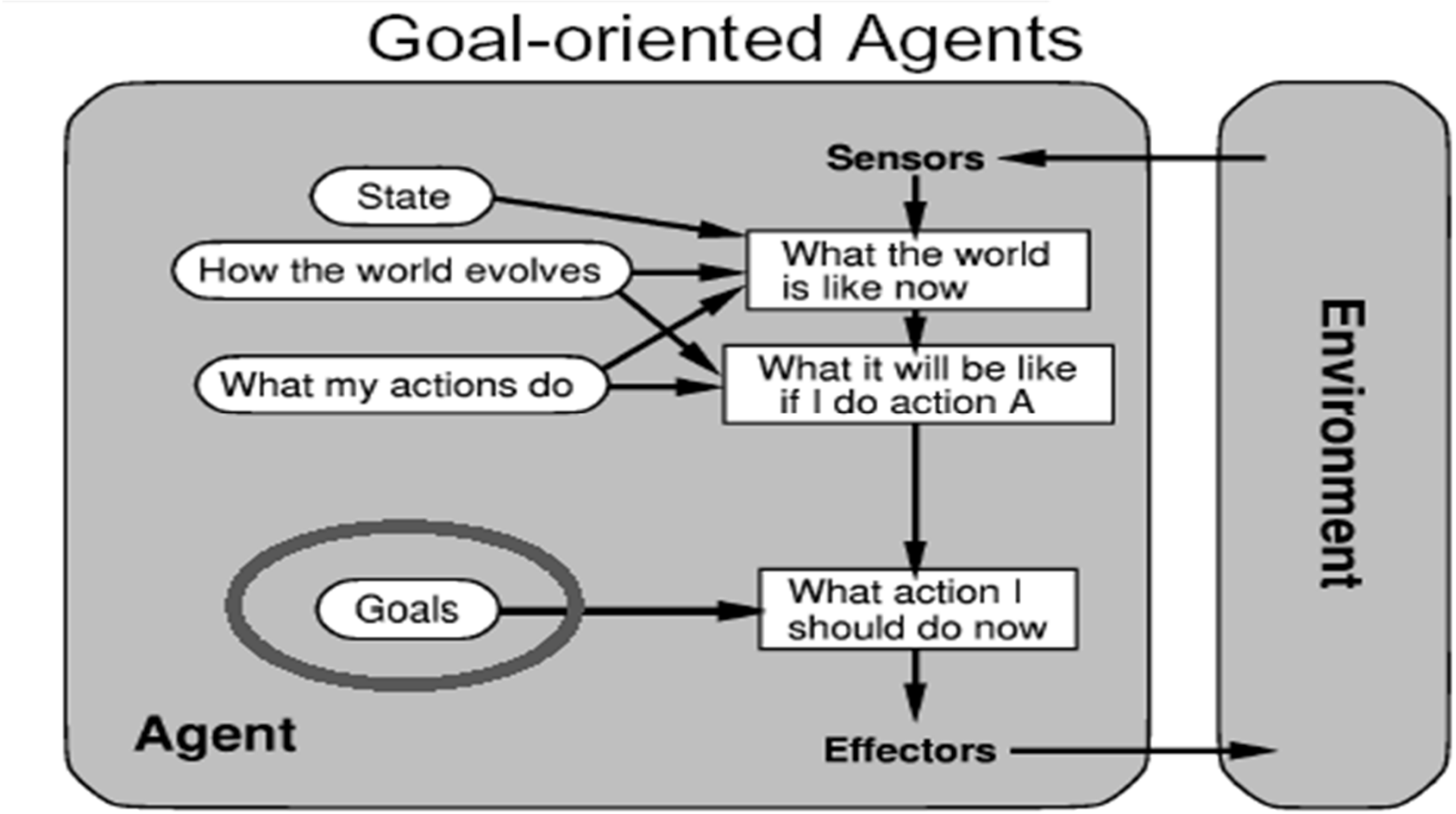


Intelligente, lernende Agenten: Systemarchitektur

Definition of an Intelligent Learning Agent



Interne Struktur eines zielorientiert arbeitenden Agenten

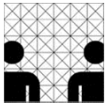


[I. Timm, nach Russell/Norvig]

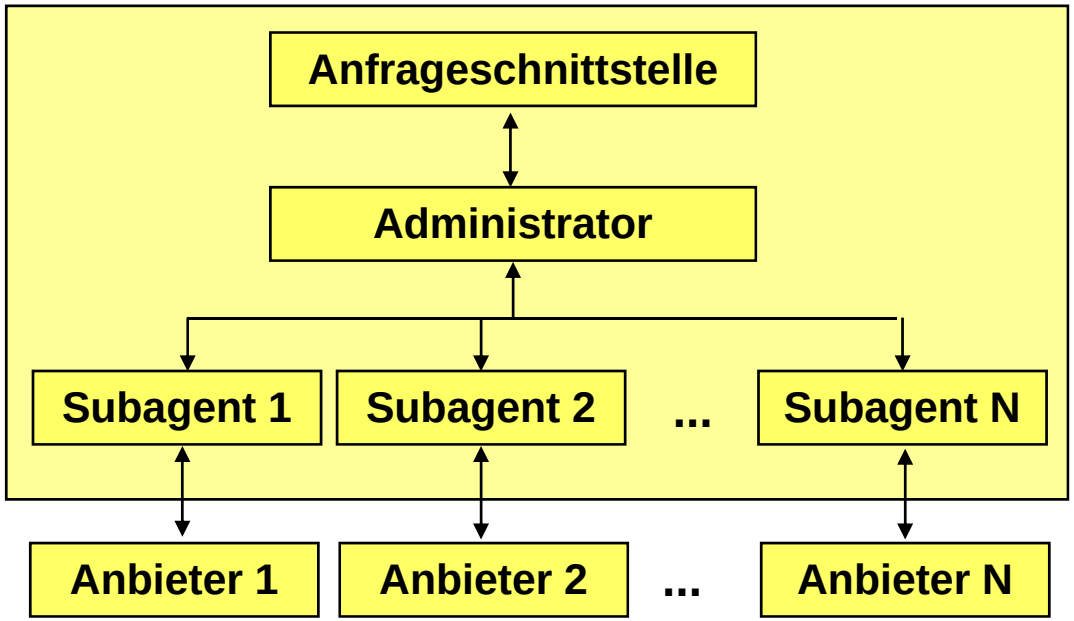
Electronic Commerce

„Software-Agenten“ übernehmen die Rolle von Anbietern und Nachfragern

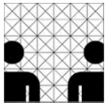
- ***Einfache Kaufagenten*** **verschaffen Produktinformationen**
  - Suche
  - Preisvergleich
- ***Komplexe Kaufagenten*** **unterstützen gesamten Kaufvorgang**
  - Suche
  - Preisvergleich
  - Zahlung
  - Lieferung
- ***Agentenbasierte Marktplätze*** **umfassen Kauf- und Verkaufsagenten, Kreditagenten, Zahlungsagenten, Werbeagenten etc.**
  - Suche
  - Werbung
  - Preisvergleich
  - Verhandlung
  - Kreditvergabe
  - Zahlung
  - Lieferung



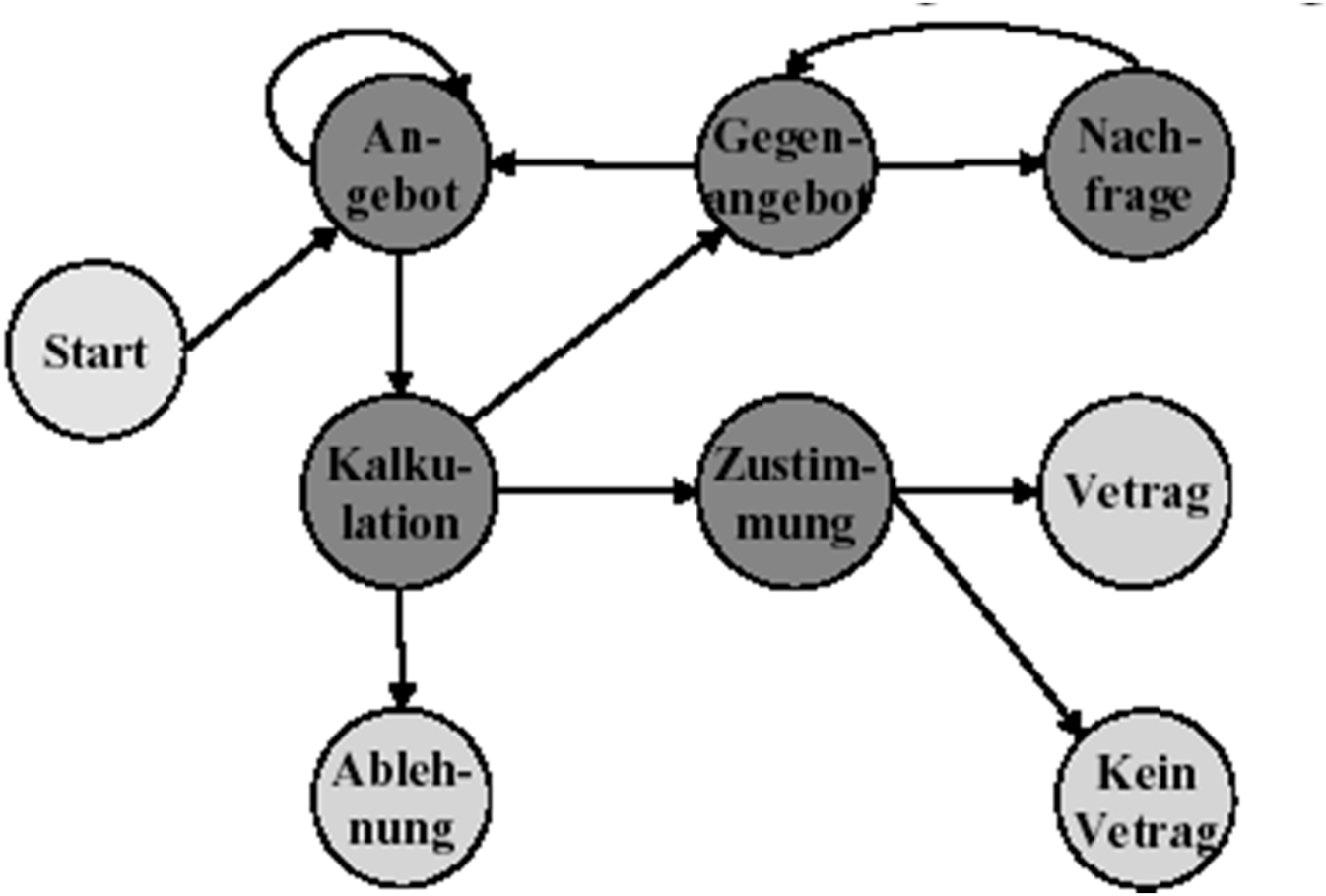
Beispiel: Einfache Kaufagenten



- **Anfrageschnittstelle**
  Definition eines Produktwunsches und Präsentation der Ergebnisse
- **Administrator**
  Kennt Anbieter. Beauftragt Subagenten und aggregiert deren Ergebnisse
- **Subagent**
  Führt Suchanfrage bei Anbieter durch und übermittelt Ergebnisse
- **Anbieter**
  Stellt seine Produktdatenbestände zur Verfügung



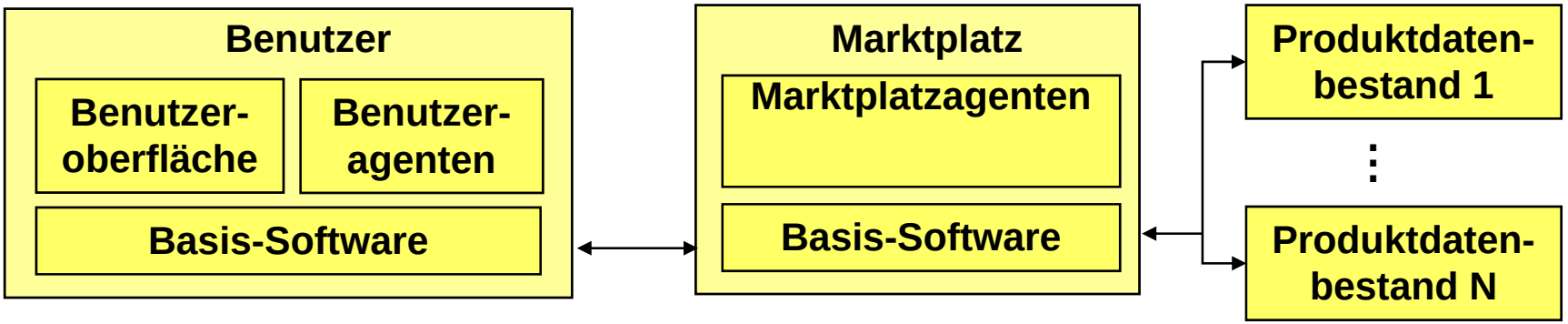
Allgemein: Automatische Verhandlungssteuerung



[Parson et al., 1998]

Agentenbasierte Marktplatz

Kooperation unabhängig agierender Agenten mit Verhandlungsfähigkeiten



Benutzeroberfläche

Zur Verständigung zwischen einem Benutzer und seinem Agenten.

Eingabe notwendiger Parameter

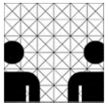
Benutzeragent

Unterstützung eines Benutzers beim Kauf oder Verkauf von Produkten.

Kontakt mit dem Marktplatz über Basis-Software

Marktplatzagent

Unterstützung von Anbietern beim Verkauf ihrer Produkte. Zugriff auf Produktdaten-bestände. In der Regel ein Marktagent pro Anbieter.



Wann ist Agenten-Technologie nützlich?

Komplexe Systeme

Beispiele: Roboter, Produktionsautomatisierung, Katastrophenplanung

- Dekomposition in Agenten bietet Modularisierung und Abstraktion
- Metapher einer Gesellschaft kooperierender Agenten

Offene verteilte Systeme

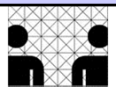
Beispiele: Internet, Reservierungssystem, Auskunftssystem, E-Commerce

- Struktur eines offenen Systems ändert sich dynamisch
- Systemkomponenten sind heterogen
- Kooperation durch Aushandeln (Negotiation) statt Auftragsvergabe

Partnerschaftliche Kooperationssysteme

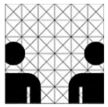
Beispiele: Beratungssysteme, Handelspartner, Manager

- vage, unpräzise Auftragsbeschreibung
- proaktives Verhalten, eigene Vorschläge
- Agent passt sich an veränderte Aufgaben und veränderte Umgebung an



Übersicht

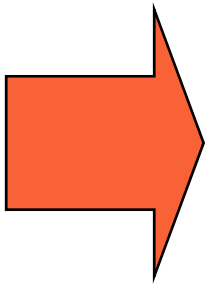
- E1: Motivation und Begriffe
  - IT Trends und (mögliche) Folgen
  - Definitionen von Agent & Multiagentensystem
  - Eigenschaften von Agenten & MAS
- E2: Wozu Agenten?
  - Taxonomien und Arten von Agenten
  - Anwendungsgebiete für MAS
- **E3: Wie werden Agenten konstruiert?**
  - **BDI-Agentenarchitektur**
  - **Jadex-Agentenplattform**



Grundelemente einer wissensbasierten Agentenarchitektur

Agent muss über sein Umfeld Bescheid wissen

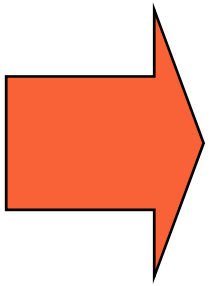
"Weltwissen", "Domänenwissen", "Weltmodell"



BELIEFS

Agent muss seine Möglichkeiten kennen

"Ressourcen", "Fähigkeiten", "Skills"



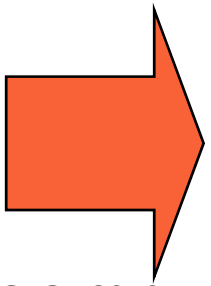
DESIRES

Agent muss seinen Zweck kennen

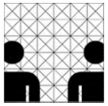
"Ziele", "Wünsche"

Agent muss Aktivitäten verfolgen

"Pläne", "Absichten"

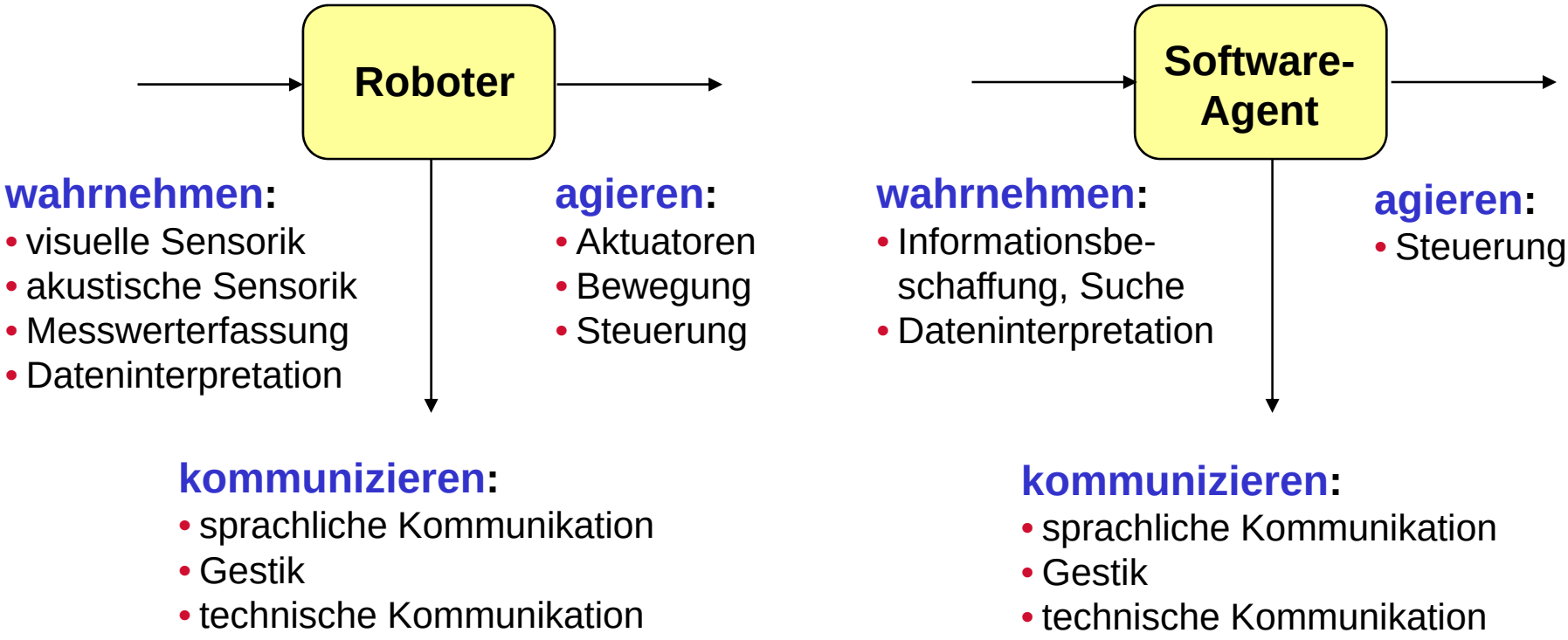


INTENTIONS



Roboter vs. Software-Agenten

- **BDI-Architektur ist Metapher für Robotik und Agenten-Technologie**
- **Hauptunterschiede entstehen durch unterschiedliche Welten und unterschiedliche Interaktionsmöglichkeiten**



Anforderungen an BDI-Agenten

Komponenten eines BDI-Agenten müssen (je nach Anwendung) komplexe Funktionalität unterstützen. Dafür gibt es teilweise umfangreiche Methodenschätze aus eigenständigen Forschungsbereichen.

BELIEFS - Verwaltung von heterogenem, veränderlichen Wissen über das Tätigkeitsfeld des Agenten

DESIRES - heterogene Zielvorgaben mit veränderlicher Priorität

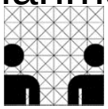
INTENTIONS - Aktionen planen, die zur Erfüllung von Zielen dienen

KOMMUNIZIEREN - mit Menschen und kooperierenden Agenten in jeweils geeigneter Kommunikationsform

WAHRNEHMEN - multimodale Informationen erfassen & interpretieren, eigenes Wissen ändern oder erweitern

AGIEREN - in das Umfeld eingreifen, zielorientierte Aktionen planmäßig durchführen

- **Objekt-orientierte Datenmodelle**
- **Wissens-repräsentation**
- **Operations Research**
- **KI-Planungs-techniken**
- **Multiagenten-systeme**
- **Bild- und Sprachverstehen**
- **Robotik**

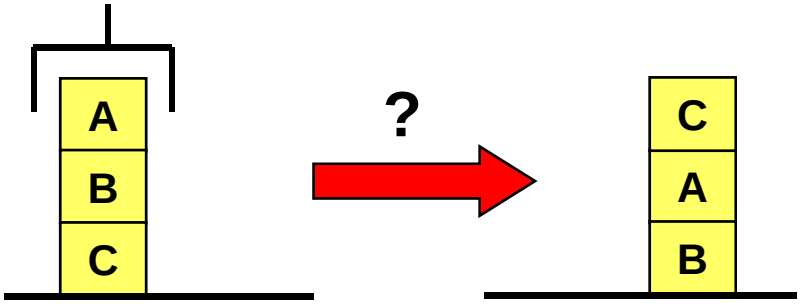


Planen

Planen ist zentrale Kompetenz eines BDI-Agenten:

Planen ist das Entwerfen einer Aktionsfolge, mit der eine Startsituation in eine gewünschte Zielsituation überführt werden kann

Veranschaulichung von Planungsproblemen anhand der Blockswelt:

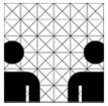


Startsituation:
on (A, B), on (B,C), on (C, Table)

Elementare Aktion:
move (x, y)

***Planungsverfahren* sind Forschungsgegenstand in mehreren Disziplinen:**

- **Künstliche Intelligenz (KI)**
- **Verteilten Systemen**
- **Operations Research**
- **Automatisierungstechnik etc.**

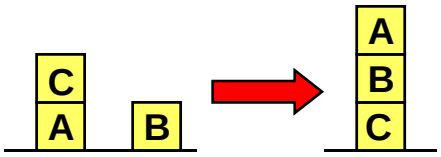


Planen als Suchen

Suchproblem wird definiert durch

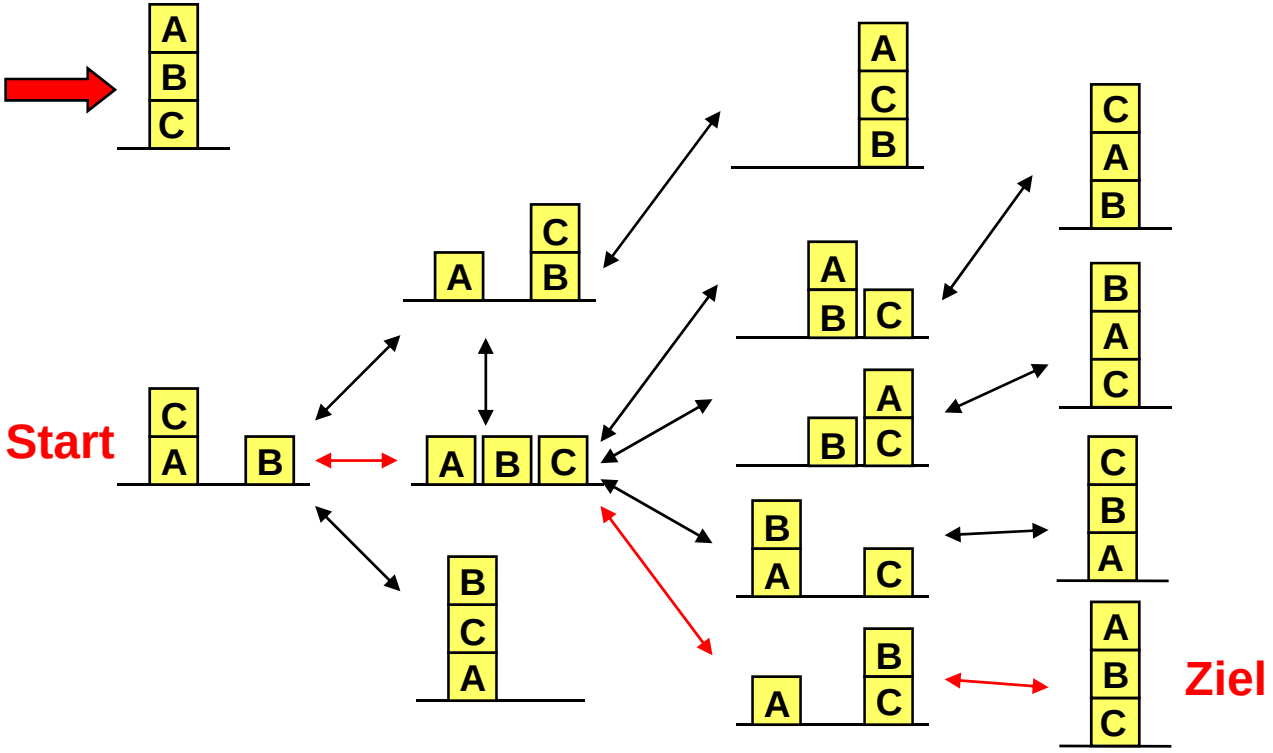
- **Zustände (speziell: Startzustand, Zielzustände)**
- **Operatoren zur Zustandstransformation**
- **Suchstrategie**

Beispiel:



Suche ergibt Plan:

move (C, Table)
move (B, C)
move (A, B)



Probleme beim Planen

Qualifikationsproblem:

Unter welchen Vorbedingungen kann eine Aktion ausgeführt werden?

Rahmenproblem (frame problem):

Was sind die Effekte einer Aktion?

Beispiel: Reiseplanung

Operator "fahre_mit_eigenem_auto" ist *anwendbar* wenn

- *man eigenes Auto besitzt*
- *Auto nicht defekt ist*
- *Auto nicht gestohlen ist*
- *Autoschlüssel zur hand ist*
- *man autofahren kann*
- *Autofahren nicht verboten ist*
- *kein Erdbeben stattfindet*
- *das Auto nicht explodiert*
- *...*

Operator "fahre_mit_eigenem_auto" hat *Effekte*:

- *Ortsveränderung von Auto und Ladung*
- *Verbrauch von Benzin*
- *Ausstoß von Abgasen*
- *Abnutzung des Autos*
- *Abnutzung der Straße*
- *Behinderung von Verkehrsteilnehmern*
- *Lärmbelästigung*
- *Auslösen eines Erdrutsches*
- *...*

Beispiel

Ziel: Gelange zur Uni

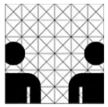
Plan: Gehe zu Fuß zur Uni
Für Ziel: Gelange zur Uni
Kontext: Wenn es nicht regnet
Body: Gehe zu Fuß zur Uni

Plan: Fahre mit dem Zug zur Uni
Für Ziel: Gelange zur Uni
Kontext: Wenn es regnet
Body: Sub-goal: Nehme ein öffentliches Verkehrsmittel = Zug

Plan: Fahre mit der Straßenbahn zur Uni
Für Ziel: Gelange zur Uni
Kontext: immer anwendbar
Body: Sub-goal: Nehme ein öffentliches Verkehrsmittel = Straßenbahn

Plan: Nehme ein öffentliches Verkehrsmittel
Für Ziel: Nehme ein X
Kontext: immer anwendbar
Body: Gehe zur Haltestelle, Überprüfe Fahrplan, Scheitere wenn lange Wartezeit, Nimm X

[Quelle: Winikoff, Padgham: Developing Intelligent Agent Systems]



Beispiel für Multi-Agentensystemplattform „Jadex“ Reasoning Engine (Uni HH, FBI, VSIS)

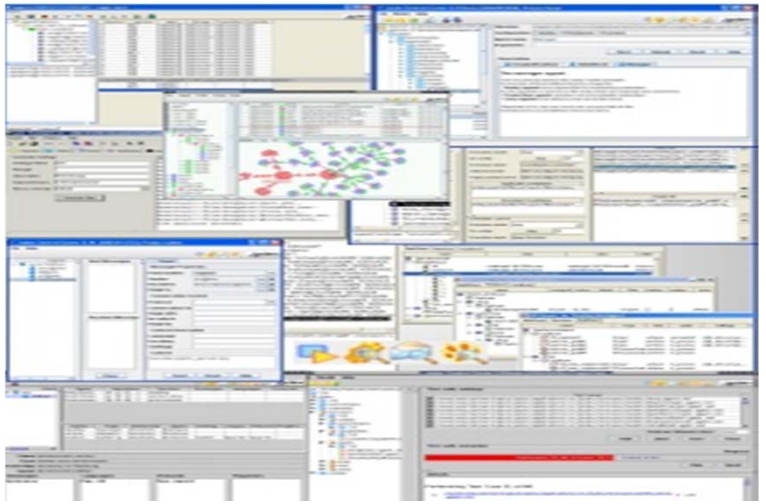
- **Forschungsprojekt am Arbeitsbereich *VSIS***
- **Jadex *Reasoning Engine* für Agentensysteme**
  - folgt dem *Belief-Desire-Intention* Modell
  - unterstützt unterschiedliche Agentensysteme (*JADE*, *DIET,...)* läuft aber auch "*stand alone"*

➔ **erleichtert die Programmierung (mit Java+XML) von intelligenten Agenten**

➔ **unterstützt den Programmierer durch diverse Tools**

➔ **Ziel: Simulation & AOSE**
(Agent-oriented Software Engineeering)

➔ **enthält viele anschauliche Beispiele intelligenter Agenten**

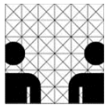


Wissen (Beliefs) in Jadex (1)

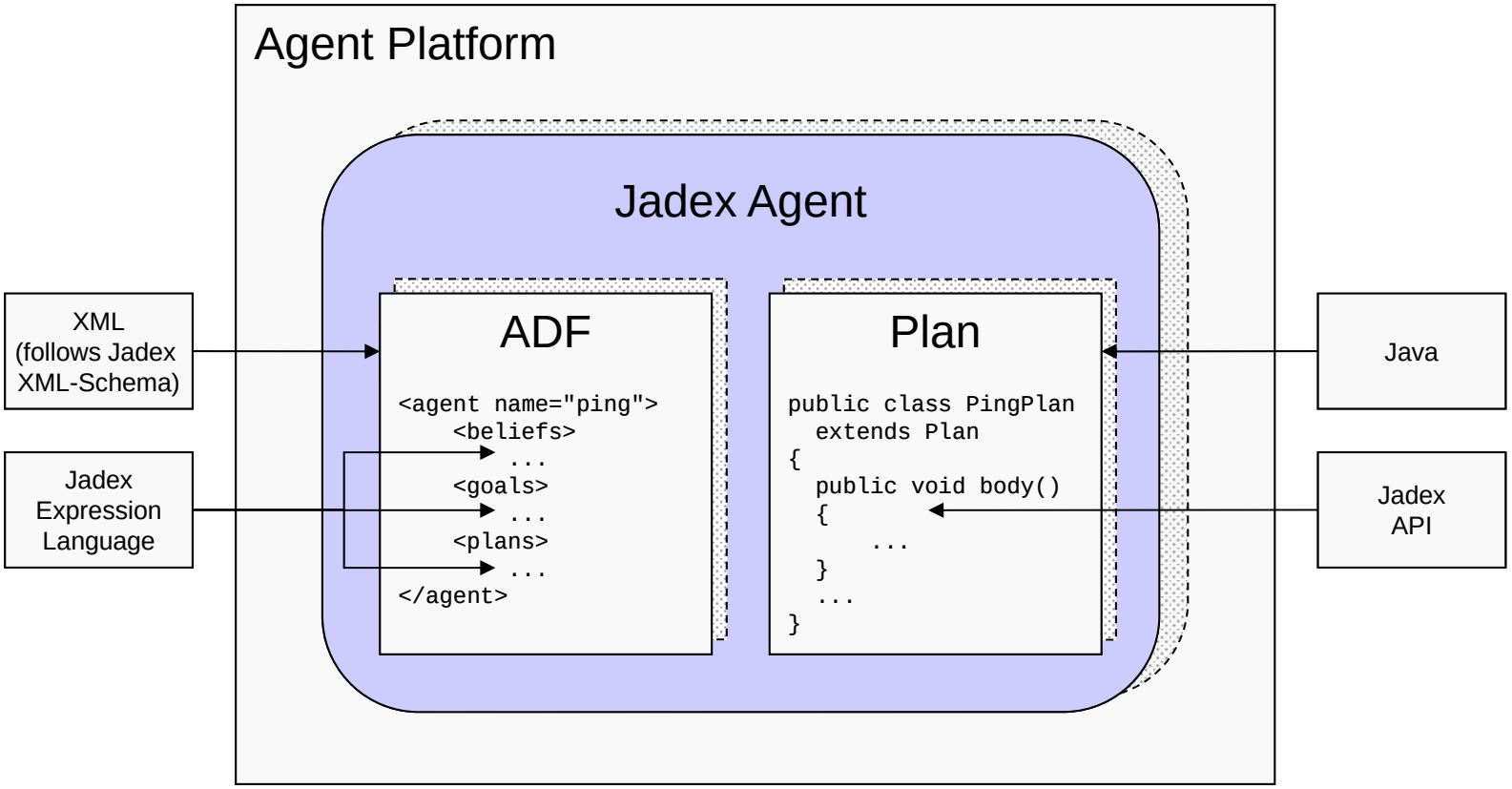
- **Beliefs dienen der Speicherung von Wissen eines Agenten**
  - **objektorientierte Beschreibung (anstelle der üblichen Logik-basierten)**
  - **Speicherung von beliebigen Objekten als Fakten innerhalb von über Namen zugreifbaren**
    - Beliefs (ein Fakt)
    - Belief Sets (eine Menge von Fakten)
  - **Zugriff über Methoden der Belief Base, oder über eine mengenorientierte, deskriptive Anfragesprache (an OQL orientiert)**

```
<beliefs>
  <!-- The battery charging state (1.0 full -> 0.0). -->
  <belief name="my_chargestate" class="double">
    <fact>0.85</fact>
  </belief>
</beliefs>
```

```
<beliefs>
  <!-- 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>
  </beliefset>
</beliefs>
```



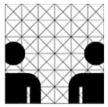
Wissen (Beliefs) in Jadex (2)



Ziele (Goals) in Jadex

- **Ziel dienen dazu, Wünsche des Agenten darzustellen**
  - **Ziele führen dazu, dass der Agent Aktionen unternimmt, ein Ziel zu erreichen**
  - **Ziele müssen nicht konsistent sein (entsprechen Desires)**
  - **Ziele lassen sich nach ihrem Typ verfeinern**
    - Perform Goals
    - Achieve Goals
    - Query Goals
    - Maintain Goals
    - (Meta-level Goals)

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

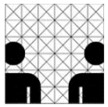


Pläne (Plans) in Jadex

- **Pläne repräsentieren prozedurales Wissen des Agenten zum Erreichen seiner Ziele**

- **Pläne bestehen aus Plan-Head und -Body**
  - ***Head* definiert wann ein Plan anwendbar ist**
    - Trigger, precondition, context condition, body Erzeugung
  - ***Body* enthält den auszuführenden Code**
    - Von sehr abstrakt (nur Subgoals), bis sehr konkret (nu

```
<!-- Perform patrols. -->
<plan name="patrol">
   <body>new PatrolPlan()</body>
      <trigger>
         <goal ref="performpatrol"/>
      </trigger>
</plan>
```



Jadex-Beispiele

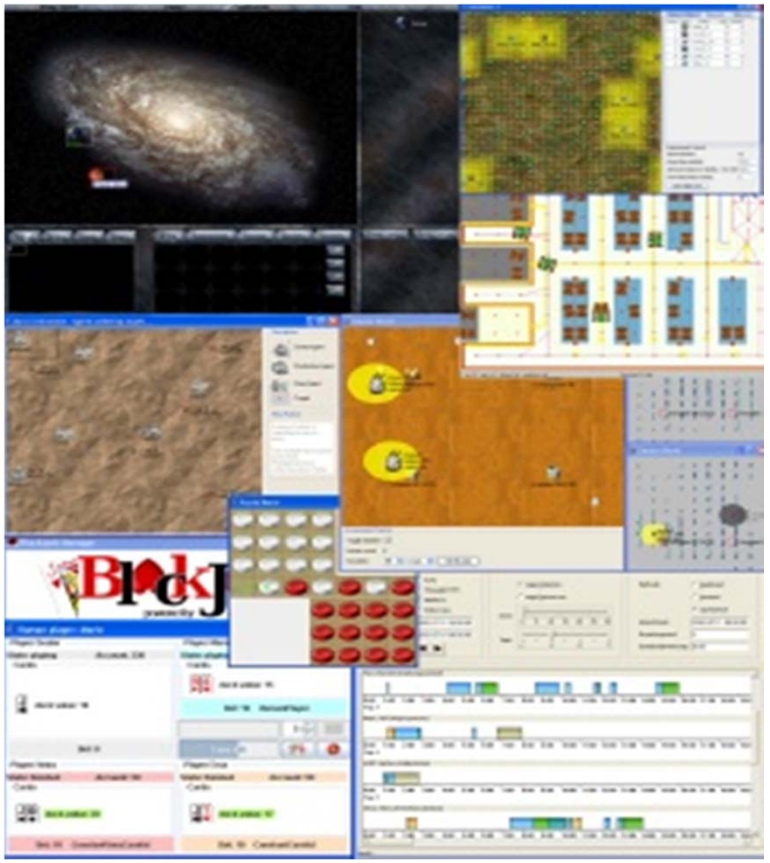
Beispiele im Web unter:

http://www.activecomponents.org/bin/view/Documentation/Examples

System frei verfügbar unter

SourceForge.net/Projects

http://sourceforge.net/projects/jadex/



Zusammenfassung

- **Diskussion der Grundbegriffe**
  - Agent
  - Umwelt
  - Multiagentensystem
- **Anwendungsgebiete von Multiagentensystemen**
  - Kategorien von Anwendungen
  - Beispielanwendungen
- **BDI-Agenten & Implementation**
  - BDI-Agentenarchitektur
  - Jadex-Agentenframework