Vorlesung

Multiagentensysteme

Dr. Jörg Müller

Siemens AG, CT IC 6

joerg.p.mueller@siemens.com

Struktur der Vorlesung

1. **Einführung**
   - **Motivation**
   - **Definitionen und Abgrenzung**
   - **Intelligente Agenten: Basismodell**
2. **Agentenarchitekturen, Wissensrepräsentation und Reasoning**
   - **Deduktives Reasoning**
   - **Praktisches Reasoning**
   - **Reaktive und Hybride Agenten**
3. **Heuristische Verfahren zur Entscheidungsfindung**
   - **Problemlösen als Suche**
   - **Einführung in heuristische Verfahren**
   - **Suchverfahren**

Struktur der Vorlesung (2)

4. **Interaktion in Multiagentensystemen**
   - Definition
   - Multiagentensysteme und Spieltheorie
5. **Verhandlungsmodelle**
   - Einigungsprozesse zwischen Agenten
   - Auktionen
   - Verhandlung
6. **Kommunikation, Koordination und Kooperation**
   - Kommunikationsparadigmen
   - Sprechakttheorie und planbasierte Semantik
   - MAS-Kommunikationssprachen (FIPA, KIF/KQML)
7. **Entwicklung agentenbasierter Softwaresysteme**
   - Agentenplattformen
   - Agentenorientiertes Software Engineering

S

CORPORATE TECHNOLOGY

Vorlesung Multiagentensysteme

1. EINFÜHRUNG

1.1 Motivation

- **Mega-Trends in der Informatik**
  - **Ubiquität**
  - **Konnektivität**
  - **Intelligenz**
  - **Delegation**
  - **Orientierung zum Menschen hin**
- **Weiterentwicklung der Programmierung**
  - **Sub-routining**
  - **Prozeduren und Funktionen**
  - **Abstrakte Datentypen**
  - **Objekte**
  - **Agenten und Multiagentensysteme**

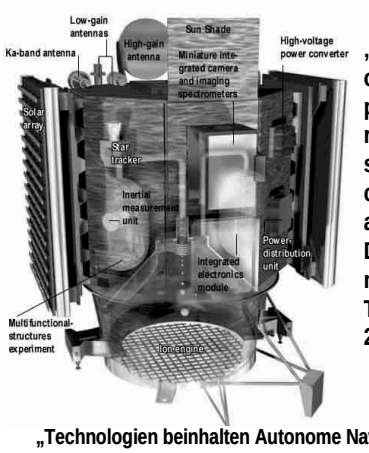
Beispiele

- **NASA Deep Space 1**
- **Autonome Serviceroboter**
- **Agenten für Entertainment**
- **Agenten in der Produktionssteuerung**
- **Intelligente Assistenz in der Westentasche**
- **Internet-Informationssysteme**

S

CORPORATE TECHNOLOGY

NASA‘s Deep Space One: Ein Autonomes Raumschiff



„Deep Space 1 launched from Cape Canaveral on October 24, 1998. During a highly successful primary mission, it tested 12 advanced, high-risk technologies in space. In an extremely successful extended mission, it encountered comet Borrelly and returned the best images and other science data ever from a comet. During its fully successful hyperextended mission, it conducted further technology tests. The spacecraft was retired on December 18, 2001“ [http://nmp.jpl.nasa.gov/ds1/]

„Technologien beinhalten Autonome Navigation und einen „Autonomen Remote Agenten“, als Intelligenter Assistent des Bodenpersonals an Bord des Raumschiffs.[http://nmp.jpl.nasa.gov/ds1/tech/autora.html]

S

CORPORATE TECHNOLOGY

Autonome Serviceroboter

- **Automatischer Postsortierer: Wahrnehmen, Entscheiden und Handeln in Alltagsumgebungen**
- **Reinigungsroboter im Supermarkt: Koexistenz mit Menschen**
- **Transportroboter in Tokioter Krankenhaus: Soziale Regeln und Normen**
- **PatientLifter: Hilfeleistung für und Kooperation mit Menschen**

Agenten für Entertainment

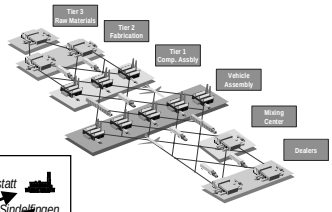
- **Robocup: Autonome Roboter spielen Fussball**
- **Physikalische und Simulationsligen**
- **Sensorik und Aktorik**
- **Dezentrale Modellierung**
- **Strategien zur Koordination und Teambildung**

- **Entertainment Software**
- **Autonome Charaktere**
- **Dezentrale Modellierung**
- **Nichtdeterministisches Gesamtverhalten**
- **Beispiele:**
  - **The Last Express (1997!)**
  - **Hunchback of Notre Dame (1996!!)**

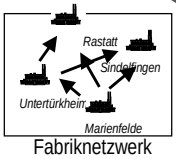
Agenten in der Produktionssteuerung (DaimlerChrysler)

Zielsetzung

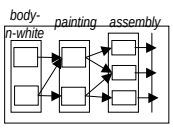
Optimierung von Produktionsprozessen



Virtuelles Unternehmen

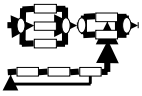


Fabriknetzwerk



Fertigungsfabrik

Produktionslinie

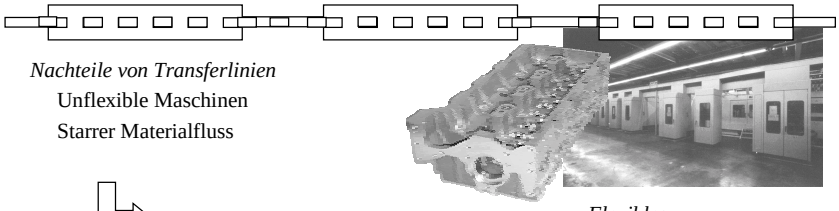


Quelle: S. Bussmann, DaimlerChrysler

S

CORPORATE TECHNOLOGY

Production 2000+: Flexibles und robustes Produktionssystem



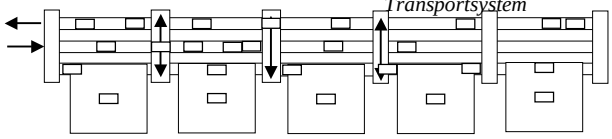
Nachteile von Transferlinien

Unflexible Maschinen

Starrer Materialfluss

Ziel: Mehr Hardwareflexibilität

Flexibles Transportsystem



Flexible Maschinen

Quelle: S. Bussmann, DaimlerChrysler

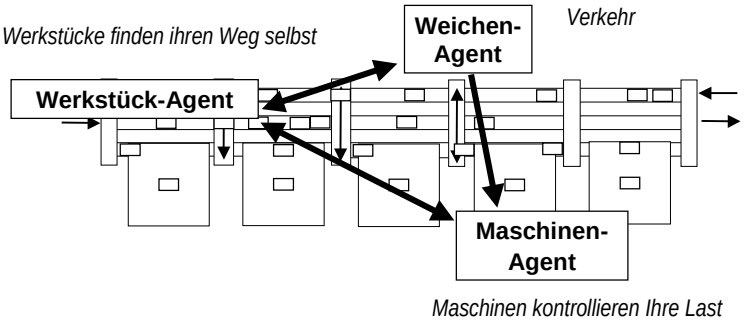
S

CORPORATE TECHNOLOGY

Production 2000+: Agentenbasierte Kontrollarchitektur

Weichen lenken den Verkehr

Werkstücke finden ihren Weg selbst



Maschinen kontrollieren Ihre Last

Kontrollmechanismen

Werkstücke wählen ihre Route

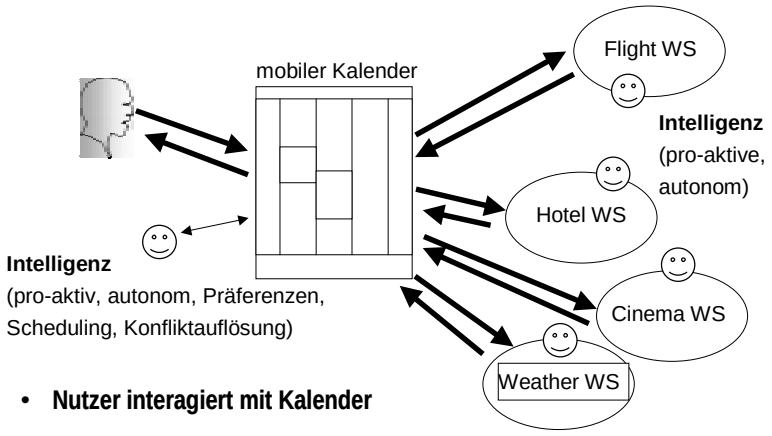
Maschinen beschränken ihre Eingangspuffer

Quelle: S. Bussmann, DaimlerChrysler

Assistenz in der Westentasche

- **Persönliche, lernfähige Assistenz auf kleinen Endgeräten**
- **Wesentlich: Ad-hoc Networking, Lokalisierungstechnologien**

Mobile Intelligenz: Agenten & Smart Web Services

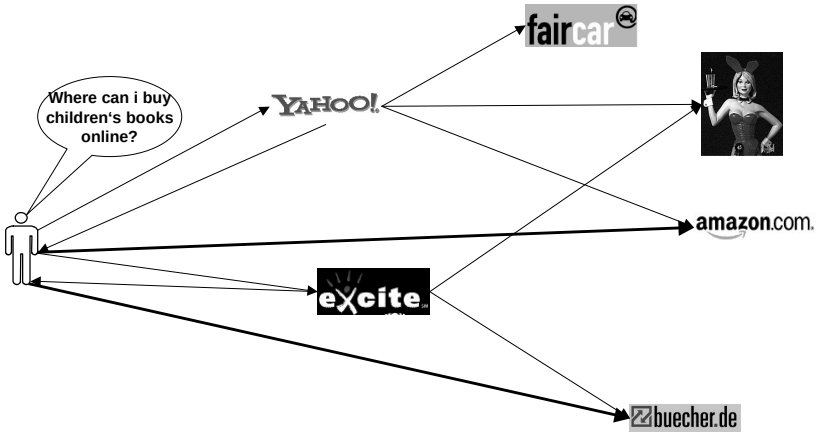


- **Nutzer interagiert mit Kalender**
- **Kalender verwendet Web-Services als Client**
- **WebServices-Agenten können Kalender benachrichtigen**
- **Intelligente Aktionen auf dem Kalender**

S

CORPORATE TECHNOLOGY

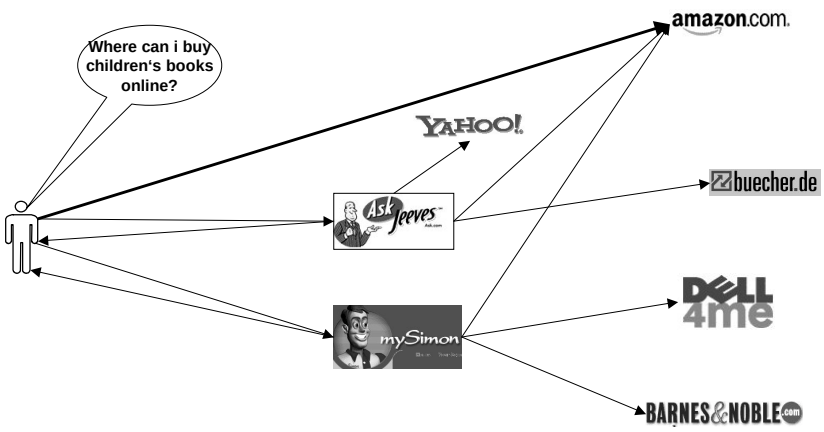
Internet Information Systems
yesterday: search engines



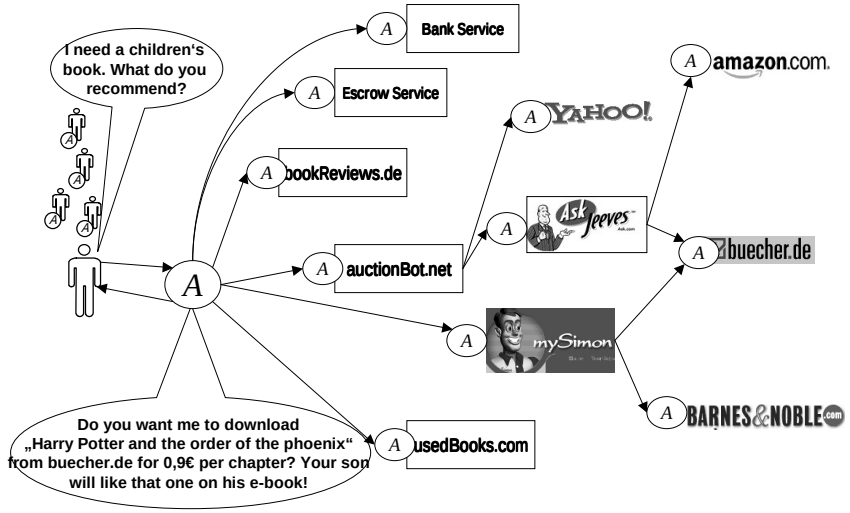
S

CORPORATE TECHNOLOGY

Internet Information Systems
today: sell-side and third-party services



Internet Information Systems tomorrow: Agents in the Semantic Web



Anforderungen an Agenten

- Autonomie (z.B. Raumschiff)
- Verrichten nützliche Aktivitäten im Auftrag von Menschen oder Organisationen (z.B. Staubsauger)
- Koexistieren und interagieren mit Menschen (Reinigungsroboter)
- Müssen sich sozialer Regeln und Normen bewusst sein (Transportroboter)
- Müssen ihre Aktivitäten koordinieren (Reinigungsroboter)
- Kooperieren miteinander oder stehen im Wettbewerb (RoboCup)
- Sollen Menschen unterhalten und belehren
- Sollen selbständig überwachen, suchen und finden, Entscheidungen treffen

S

CORPORATE TECHNOLOGY

Erster Definitionsversuch

- **Ein Agent ist ein Computersystem, das die Fähigkeit besitzt, im Auftrag seines Benutzers oder Besitzers selbständig und unabhängig Aktionen durchzuführen**
- **Ein Multiagentensystem (MAS) besteht aus mehreren Agenten, die miteinander *interagieren***
- **Erfolgreiche Interaktion erfordert die Fähigkeit zur**
  - **Kooperation**
  - **Koordination**
  - **Verhandlungsführung**

S

CORPORATE TECHNOLOGY

Kernprobleme

- **Wie konstruieren wir Agenten, die in der Lage sind, im Auftrag seines Benutzers oder Besitzers selbständig und unabhängig Aktionen durchzuführen?**
- **Wie konstruieren wir Multiagentensysteme, d.h., Agenten, die in der Lage sind, mit anderen Agenten zu interagieren (koordinieren, kooperieren, verhandeln), um die Aufgaben auszuführen, die wir ihnen delegieren, und die dabei Ziel- und Interessenskonflikte auflösen können, die sich dabei ergeben?**

Agenten und MAS: Perspektiven

- ***Agenten als Paradigma für Software Engineering***

Software-Ingenieure haben heute ein gutes Verständnis für die Charakterisierung von Komplexität in Softwaresystemen erreicht. Es ist mittlerweile anerkannt, dass Interaktion bzw. die Notwendigkeit von lose gekoppelter Interaktion wahrscheinlich die wesentlichste Charakteristik komplexer Softwaresysteme ist.

Z.B. CORBA, EJB, .NET: Entkopplung von Komponentendesign und Systemdesign. Der Entwickler einer Komponente weiß nicht, welche Clients in welchen Situationen mit seiner Komponente interagieren werden.

S

CORPORATE TECHNOLOGY

Agenten und MAS: Perspektiven

- ***Agenten als Werkzeug zum besseren Verstehen menschlicher Gesellschaften***

Multiagentensysteme bieten neue Ansätze zur Simulation menschlicher Gesellschaften. Dies kann uns helfen, neue Erkenntnisse zu unterschiedlichen sozialen Prozessen zu gewinnen.

Z.B. Warum verhalten sich Wähler in einer Demokratie so, wie sie es tun? Wie funktionieren Märkte und warum funktionieren sie manchmal nicht? Wie muß man Regeln in sozialen Systemen entwerfen, damit diese Systeme funktionieren?

Fragestellungen

- Sind Agenten/MAS nicht nur ein neuer Name für verteilte, nebenläufige Systeme?
- ... oder für dynamische Regelungssysteme?
- ... oder für Künstliche Intelligenz und Expertensysteme?
- ... oder für Wirtschafts- und Spieltheorie?
- ... oder gar für Sozialwissenschaften?

1.2 Exkurs: Agenten und Dynamische Systeme

- Dynamisches System (Regelungstheorie) besteht aus
  - System (Controller)
  - Umgebung
- Sei
  - T Menge der Zeitpunkte
  - X Menge möglicher Zustände der Umgebung
  - U Menge möglicher Eingaben an das System
  - Y Menge möglicher Aktionen des Systems
- Sei
  - x(t) Zustand der Umgebung zum Zeitpunkt $t \in T$
  - u(t) Eingabe an das System zum Zeitpunkt $t \in T$
  - y(t) Ausgabe des Systems zum Zeitpunkt $t \in T$

Beschreibung der Umgebung in dynamischen Systemen

- **Das Verhalten der Umgebung in dynamischen Systemen wird durch zwei Funktionen beschrieben**
- **Aktionen, die durch das System ausgeführt werden, verändern die Umgebung:**

$$x(t+1) = f(x(t), y(t))$$

- **Output der Umgebung (Signale) sind Input für das System und ergeben sich direkt aus dem gegenwärtigen Zustand der Umgebung x(t):**

$$u(t) = g(x(t))$$

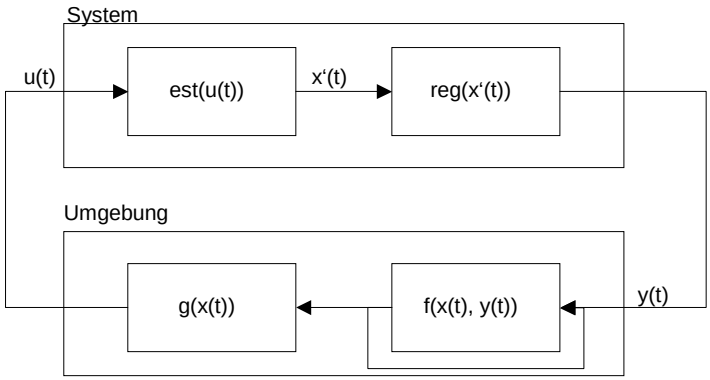
Verhalten des Systems: Das Regelungsproblem

- **Aufgabe des Systems: Lösen des Regelungsproblems**
  - **= Finden einer Folge von Aktionen, um einen Zielzustand der Umgebung zu erreichen**
- **Das Regelungsproblem wird oft in zwei Teilprobleme unterteilt:**
- **Das *Zustandserkennungsproblem* bestimmt den gegenwärtigen Zustand der Umgebung aus dem momentanen Input in das System**
- **Das *Regulierungsproblem* bestimmt eine angemessene Reaktion (Aktion) für einen erkannten Zustand der Umgebung**
- **Funktionen:**
  - **Zustandserkennung: est: U → X**
  - **Regulator: reg: X→Y**

S

CORPORATE TECHNOLOGY

Komponenten eines dynamischen Systems



S

CORPORATE TECHNOLOGY

Praktische Implikationen für Agenten

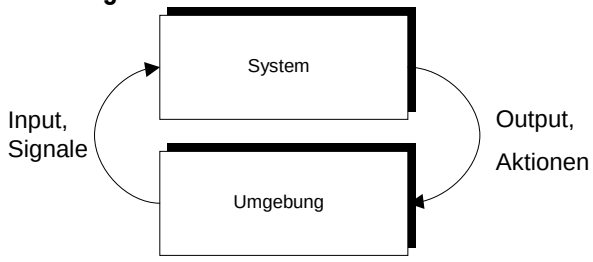
- **Agenten können als dynamische Systeme beschrieben werden, d.h., lösen Regelungsprobleme**
- **In praktischen Anwendungen sind die Funktionen est und reg oft unvollständig und inkorrekt**
- **Gründe:**
  - **Fehlerhafte, ungenaue, unvollständige Sensorik (Input in est)**
  - **Fehlerhaftes, ungenaues, unvollständiges Situationsmodell (Output von est)**
  - **D.h., das System / der Agent glaubt, die Umgebung sei in einem anderen Zustand, als sie es wirklich ist**
  - **Fehlerhaftes, ungenaues, unvollständiges Aktionsmodell (reg), d.h., Auswirkungen von Aktionen auf die Umgebung**
  - **Fehlerhafte, ungenaue, unvollständige Aktorik (Output von reg)**

Basisprobleme der Modellierung intelligenter Agenten

- **Ein wichtiger Fokus der Forschungsarbeiten im Bereich intelligenter Agenten ist die Lösung dieser Basisprobleme dynamischer Systeme**
- **Viele der Agentenarchitekturen, Wissensrepräsentations- und Reasoningmodelle sowie Entscheidungsverfahren, die wir in dieser Vorlesung kennenlernen, können in diesem Licht gesehen werden**
- **Ende Exkurs**

1.3. Was ist ein (Intelligenter) Agent?

- **Autonom:**
  - **Fähig zur eigenständigen, unabhängigen Aktion**
  - **Übt Kontrolle über den eigenen Zustand aus**
- **Agent = Computer-System, das fähig ist, in einer Umgebung autonom zu agieren**



S

CORPORATE TECHNOLOGY

Definition: Intelligente Agenten

- **Nach dieser Definition sind**
  - **ein Thermostat**
  - **ein Unix-Dämon (z.B. biff)**

  Beispiele für triviale (uninteressante) Agenten
- **Ein *Intelligenter* Agent ist ein Computer-System, das fähig ist, in einer Umgebung *flexibel* und autonom zu agieren**
- **Flexibel bedeutet:**
  - **Reaktiv**
  - **Pro-aktiv**
  - **Sozial**

S

CORPORATE TECHNOLOGY

1.3.1 Reaktivität

- **Wenn die Umgebung eines Programm statisch (bekannt) ist, wird das Programm nicht fehlschlagen**
  Beispiel: Compiler
- **In der realen Welt ändern sich Dinge, Information ist unvollständig; interessante Umgebungen sind dynamisch**
- **Die Entwicklung von Software für dynamische Umgebungen ist schwierig: Programme müssen die Möglichkeit von Fehlschlägen einkalkulieren**
- **Ein reaktives System steht in ständiger Interaktion mit seiner Umgebung und reagiert auf Änderungen rechtzeitig**

Exkurs: Exception Handling und Reaktivität

- Die Programmiersprache Java unterstützt ein Modell zur Behandlung von Fehlerbedingungen und Ausnahmesituationen (Exception Handling) im Programmablauf
- An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions.
- Java requires that a method either catch or specify all checked exceptions that can be thrown within the scope of the method.
- Typen von Exceptions
  - Runtime Exception: Exception im Laufzeitsystem (kein Catch oder throws ... notwendig)
  - Checked Exception: Wird von Compiler geprüft

Beispiel: ExceptionHandling in Java

```
public void myMethod(Vector vec) throws IOException {
...
PrintWriter out = null;
try {
      out = new PrintWriter( new FileWriter("OutFile.txt"));
       for (int i = 0; i < size; i++)     out.println("Value at:"+ I +"="+ vec.elementAt(i));
} catch (ArrayIndexOutOfBoundsException e) {
      System.err.println("CaughtArrayIndexOutOfBoundsException: " + e.getMessage()); }
catch (IOException e) {
      System.err.println("Caught IOException: " + e.getMessage());
      e.setMessage(„Foo“);
     throw e; }
finally {
      if (out != null) {  System.out.println("Closing PrintWriter"); out.close(); }
       else { System.out.println("PrintWriter not open"); }
 }  ...}
```

Exception-Handling Konstrukte

- Try-Block: Bestimmt Geltungsbereich (Scope) der Exception
- Catch-Block (für jede Exception: bestimmt im Falle des Auftretens der Exception im vorausgehenden try-Block, was getan wird.
- Finally-Block: Wird in jedem Fall ausgeführt, egal ob im vorausgegangenen try-Block eine Exception auftritt oder nicht.
- Übrigens:
  try { ...} catch (Exception e) {System.out.println(e);}

  ist sehr schlechter Programmierstil. Alles Wissen, das man zur Designzeit über das Verhalten eines Programmes hat, sollte man auch nutzen!

Java hat Exceptions. Ist Java reaktiv?

- Nein, denn Java Exceptions müssen zur Designzeit definiert werden
- Die Behandlung von Ausnahmezuständen ist in den Exception Handling-Blöcken (catch(){}) festverdrahtet und somit nicht flexibel
- Fazit: Reakitivät ist mehr als Exception Handling, aber ein solides Exception Handling Modell ist eine Grundlage, auf deren Basis Reaktivität implementiert werden kann.
- Ende Exkurs

S

CORPORATE TECHNOLOGY

1.3.2 Pro-aktivität

- **Reaktive Systeme lassen sich relativ einfach modellieren (z.B. Stimulus-Response-Regeln)**
- **Aber: Agenten sollen Dinge für uns tun, und sich dementsprechend *zielgerichtet verhalten***
- **Pro-Aktivität =**
  - **Selbständiges Generieren von Zielen**
  - **Versuch, Ziele zu erreichen**
  - **Initiative übernehmen, nicht nur ereignisgesteuertes Verhalten**

S

CORPORATE TECHNOLOGY

1.3.3 Soziale Fähigkeiten

- **Die reale Welt ist eine Multiagenten-Umgebung**
- **Bei der Erreichung von Zielen sind die Ziele anderer zu berücksichtigen**
- **Manche Ziele können nur in Kooperation mit anderen erreicht werden (z.B. mangelnde Fähigkeiten, beschränkte Ressourcen)**
- **In manchen Situationen bestehen Konflikte und Konkurrenz (z.B. Internet-Auktionen)**
- **Soziale Fähigkeit in Agenten ist die Fähigkeit, bei der Erreichung lokaler Ziele die Ziele anderer Agenten (auch Menschen) zu modellieren, und mit ihnen zu interagieren (koordinieren, kooperieren).**

1.3.4 Weitere mögliche Eigenschaften von Agenten

- **Mobilität: Fähigkeit, sich in einem Rechnernetzwerk zu bewegen**
- **Wahrhaftigkeit: ein Agen verbreitet nicht wissentlich falsche Information (d.h., Information, die er für falsch hält)**
- **Gutwilligkeit: Agenten haben keine konfligierenden Ziele, Agenten werden deshalb an sie herangetragene Aufgaben stets zu erfüllen versuchen**
- **Rationalität: Ein Agent strebt danach, seine Ziele zu erfüllen und wird nicht wissentlich in einer Art und Weise handeln, die die Erfüllung seiner Ziele beeinträchtigt**
- **Adaptivität/Lernfähigkeit: Agenten verbessern ihr Verhalten im Laufe der Zeit**

1.3.5 Agenten und Objekte

- **Objekte enkapsulieren einen Zustand**
- **Objekte kommunizieren über Nachrichtenaustausch (Message Passing)**
- **Das Verhalten (Aktionen) von Objekten ist in Methoden beschrieben, die von anderen Objekten (Clients) aufgerufen werden können**
- **In der Regel sind Methoden entweder**
  - **Öffentlich**
  - **Von Methoden des gleichen Kontexts sichtbar**
  - **Nur für die Klasse und Ihre Unterklassen sichtbar**
  - **Nur für die Klasse sichtbar (privat)**

S

CORPORATE TECHNOLOGY

Agenten und Objekte: Unterschiede

- **Agenten sind *autonom***

  Agenten realisieren eine stärkere Form von Autonomie als Objekte; sie entscheiden selbst, ob sie eine Aktion ausführen oder eine Anfrage seitens eines anderen Agenten bearbeiten

- **Agenten besitzen *Intelligenz***

  Agenten sind fähig zu flexiblem Verhalten; dies liegt jenseits dem, was Standard-Objektmodelle beschreiben können

- **Agenten sind *aktiv***

  MAS sind inhärent nebenläufig, da jeder Agent einen aktiven Kontrollthread besitzt.

S

CORPORATE TECHNOLOGY

„Objects do it for free,

Agents do it for money!“

(or: because they want to)

1.3.6 Agenten und Expertensysteme /KI

- **Expertensysteme beschreiben typischerweise „körperlose“ Expertise auf einem Gebiet**
- **Beispiel: MYCIN besitzt Wissen über Bluterkrankungen des Menschen, gespeichert in Form von Regeln.**
  Ein Mediziner kann von MYCIN Ratschläge über Bluterkrankungen erhalten, indem er MYCIN Information (Fakten) über den Zustand eines Patienten gibt und Fragen stellt sowie beantwortet

Agenten und Expertensysteme: Hauptunterschiede

- **Agenten sind in eine Umgebung eingebettet („situiert“)**

  MYCIN hat keine Information über /Rückkopplung zur realen Welt, außer der Möglichkeit, dem Benutzer (Mediziner) Fragen zu stellen
- **Agenten können agieren (Aktionen ausführen)**

  MYCIN operiert nicht, es gibt nur Empfehlungen
- **Einschränkung: Es gibt Echtzeit-Expertensysteme (typischerweise in der Prozess-Steuerung), die man als Agenten bezeichnen kann.**

1.3.7 Agenten und Künstliche Intelligenz (KI)

- **Ziel der KI ist es, Systeme zu bauen, die (langfristig) menschliche Sprache verstehen, Szenen erkennen und verstehen, gesunden Menschenverstand besitzen und kreativ sind.**
- **Brauchen wir nicht alles dieses, um Agenten zu bauen?**

Agenten und KI: Unterschiede

- **Anforderung an Agenten: Fähigkeit, die richtige Aktion auszuwählen in einer *eingeschränkten Umgebung***
- **Wir müssen nicht alle Probleme der KI lösen, um einen *nützlichen Agenten* zu bauen (z.B. Email-Filter)**
- ***„We made our agents dumber and dumber and dumber ... Until finally they made money“* (Oren Etzioni, über die kommerziellen Erfahrungen bei NETBOT Inc.)**
- **Aber: Beim Design von Agenten und MAS werden häufig Methoden der KI verwendet**

1.4 Agenten als intentionale Systeme

- Zum Erklären menschlichen Verhaltens sind oft Beschreibungen nützlich wie:
  - Konrad nahm seinen Regenschirm mit, weil er *glaubte*, es würde bald anfangen zu regnen
  - Jörg arbeitete hart, weil er Professor werden *wollte*.
- Menschliches Verhalten wird oft intentionale Begriffe (Modalitäten) wie glauben, wollen, hoffen, befürchten etc beschrieben
- Daniel Dennett: Intentionale Systeme = Systeme, deren Verhalten vorausgesagt werden kann, in dem man ihnen intentionale Haltungen sowie rationales Entscheidungs-verhalten attributiert
- Ist es erlaubt / sinnvoll, Computer-Systeme als intentionale Systeme zu beschreiben?

Intentionale Systeme: McCarthy‘s Argument

‘To ascribe beliefs, free will, intentions, consciousness, abilities, or wants to a machine is *legitimate* when such an ascription expresses the same information about the machine that it expresses about a person. It is *useful* when the ascription helps us understand the structure of the machine, its past or future behaviour, or how to repair or improve it. It is perhaps never logically required even for humans, but expressing reasonably briefly what is actually known about the state of the machine in a particular situation may require mental qualities or qualities isomorphic to them. Theories of belief, knowledge and wanting can be constructed for machines in a simpler setting than for humans, and later applied to humans. Ascription of mental qualities is *most straightforward* for machines of known structure such as thermostats and computer operating systems, but is *most useful* when applied to entities whose structure is incompletely known’.

S

CORPORATE TECHNOLOGY

Was kann / sollte man als intentionales System beschreiben?

- **Man kann eigentlich alles ....**
- **'It is perfectly coherent to treat a light switch as a (very cooperative) agent with the capability of transmittingcurrent at will, who invariably transmits current when it believes that we want it transmitted and not otherwise; flicking the switch is simply our way of communicating our desires'. (Yoav Shoham)**
- **Trotzdem empfinden wir eine solche Beschreibung als absurd. Warum?**

S

CORPORATE TECHNOLOGY

Entscheidend ist der Nutzen

- **... Weil sie uns nichts bringt, weil wir den Mechanismus genügend verstehen, um eine einfachere, mechanistische Erklärung seines Verhaltens zu besitzen**
- **Je mehr wie über ein System wissen, desto weniger brauchen wir intentionale Konzepte zu seiner Beschreibung**
- **Je komplexer Computersysteme werden, desto mehr sind wir auf Abstraktionen und Metaphern angewiesen, um ihr Verhalten zu erklären.**
- **Die Betrachtung als intentionales System ist solch eine Abstraktion, somit ein Werkzeug zum Erklären, Verstehen und (letzthin) Programmieren von Computersystemen**

1.5 Abstrakte Agentenarchitekturen

- **Endliche Menge diskreter Zustände der Umgebung:**

  $E = \{e, e', ...\}$

- **Agenten verfügen über Aktionen, die den Zustand der Umgebung verändern können**

  $Ac = \{a, a', ...\}$

- **Ein Durchlauf r eines Agenten in einer Umgebung ist eine Folge:**

  $r: e_0 \xrightarrow{a_0} e_1 \xrightarrow{a_1} e_2 \xrightarrow{a_2} e_3 \xrightarrow{a_3} \ldots \xrightarrow{a_{u-1}} e_u$

- **Sei weiter:**
- **R die Menge aller möglichen endlichen Sequenzen r (über E und Ac).**
- **R^{Ac} die Teilmenge von R, die mit einer Aktion endet.**
- **R^{E} die Teilmenge von R, die mit einem Zustand endet.**

Zustandstransformationen

- Eine Zustandstransformation repräsentiert das Verhalten der Umgebung:

$$\tau: R^{Ac} \rightarrow \wp(E)$$

- Wenn $\tau(r) = \varnothing$, gibt es keine möglichen Nachfolgezustände für r. Wir sagen: der Durchlauf des Systems ist beendet.
- Formale Definition der Umgebung:

  Eine Umgebung Env ist ein Tripel $Env = (E, e_0, \tau)$, wobei E eine Menge von Zuständen der Umgebung ist, $e_0 \in E$ der Initialzustand, und τ eine Zustandstransformation

Agenten

- Ein Agent ist eine Funktion, die Durchläufe auf Aktionen abbildet:

$$Ag : R^{E} \rightarrow Ac$$

- Ein Agent trifft also eine Entscheidung darüber, welche Aktion er ausführt, basierend auf der Historie des Systems, die ihm zum Zeitpunkt der Entscheidung vorliegt.
- Sei Ag die Menge aller Agenten

Systeme

- **Ein System ist ein Paar bestehend aus einem Agent und einer Umgebung**
- **Jedem System ist eine Menge möglicher Durchläufe zugeordnet. Wir bezeichnen die Menge der Durchläufe von Agent *Ag* in der Umgebung *Env* mit *R(Ag, Env)***
- ***Eine Folge (e_0, α_0,e_1, α_1, e_2,...) ist ein Durchlauf eines Agenten in der Umgebung Env = (E, e_0, τ) wenn:***
  - e_0 **der initiale Zustand von *Env* ist**
  - $\alpha_0 = Ag(e_0)$**, und**
  - **Für *u* > *0*:**

$$e_u \in \tau(e_0, \alpha_0, \ldots, \alpha_{u-1}), \text{ wobei}$$
$$\alpha_u = Ag(e_0, \alpha_0, \ldots, e_u)$$

Rein reaktive Agenten

- **Agenten, die ihre nächste Aktion ohne Bezug auf ihre Historie entscheiden, nennen wir rein reaktive Agenten**
- **Für rein reaktive Agenten gilt:**

$$action: E \rightarrow Ac$$

- **Ein Thermostat ist ein rein reaktiver Agent**

$$action(e) = \begin{cases} aus, \text{ wenn } e = \text{„Temperatur ok“}; \\ an, \text{ sonst} \end{cases}$$

Perzeption

- **Als nächstes erweitern wir unser formales Modell um Perzeption, d.h., die Fähigkeit eines Agenten, die Umgebung wahrzunehmen**
- **Die Funktion *see* bildet Zustände der Umgebung auf Wahrnehmungen ab:**

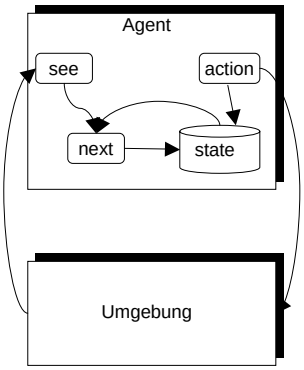
$$see: E \rightarrow W$$

- **Wir definieren Aktion nun als Funktion**

$$action: W^* \rightarrow A,$$

die Folgen von Wahrnehmungen in Aktionen abbildet. Wir nennen diese Funktion *Aktionsauswahlfunktion.*

Zustandsbasierte Agenten



- **Annahme: Agenten verfügen über interne Datenstruktur, in der Information über den Zu-stand der Umgebung und die Historie des Agenten gespei-chert wird**
- **Sei I die Menge aller internen Zustände des Agenten**

Perzeptions- und Aktionsmodell für zustandsbasierte Agenten

- **Die Perzeptionsfunktion see für zustandsbasierte Agenten ist unverändert:**

  $see: E \rightarrow W$

- **Die Aktionsauswahlfunktion action ist nun definiert als Abbildung:**

  $action: I \rightarrow Ac$

  von internen Zuständen nach Aktionen

- **Wir führen eine weitere Funktion next ein, die einen internen Zustand und eine Wahrnehmung in einen neuen Zustand abbildet:**

  $next: I \times W \rightarrow I$

Kontrollmechanismus eines Agenten

- **Jetzt haben wir das nötige Handwerkszeug, um das Verhalten eines Agenten in Form einer Kontrollschleife zu beschreiben:**
  1. **Der Agent startet in einem initialen Zustand i_0**
  2. **Er beobachtet den Zustand der Umgebung e und erzeugt eine Wahrnehmung *see(e)***
  3. **Sein interner Zustand wird über die next Funktion aktualisiert als $next(i_0, see(e))$**
  4. **Die Aktion, die der Agent auswählt, ist nun gegeben durch $action(ext(i_0, see(e))$.**
  5. **Die Aktion wird ausgeführt**
  6. **Gehe zu Schritt 2**

2.5 Agenten und Aufgaben

- **Aufgabe von Agenten ist es, Aufgaben für uns durchzuführen**
- **Diese Aufgaben müssen spezifiziert werden**
- **Fragestellung: Wie können wir Agenten sagen, *was* zu tun ist, ohne ihnen zu sagen, *wie* sie es tun sollen**
- **Diese Fragestellung führt uns zu dem Begriff der Präferenz oder des Nutzens**

Nutzenfunktionen über Zustände

- **Idee: Ein Zustand der Umgebung kann im Rahmen einer AUfgabenspezifikation mit einem Nutzenwert assoziiert werden.**
- **Die Aufgabe des Agenten ist es, Zustände zu erreichen, die diesen Nutzenwert maximieren**
- **Eine Aufgabenspezifikation ist eine Funktion:**

$$u: E \rightarrow \Re$$

die mit jedem Umgebungszustand eine reelle Zahl assoziiert

- **Fragestellung: Wie definiert man den Nutzen eines Durchlaufs eines Agenten**
  - **der maximale/minimale Nutzen eines Zustands in dem Durchlauf?**
  - **die Summe der Nutzenwerte über die Zustände des Durchlaufs?**
  - **ihren Durchschnitt?**
- **Problem: Die Zuordnung von Nutzenwerten zu Zuständen macht es schwierig, eine längerfristige Perspektive bezüglich des Nutzens zu spezifizieren**

  (Möglichkeit: Die Nutzenwerte für Zustände, die in der Zukunft liegen, wird mit einem Discount versehen)

Nutzenfunktionen über Durchläufe

- **Idee: Nutzenwerte werden nicht Zuständen, sondern Durchläufen zugewiesen**

$$u: R \rightarrow \Re$$

  Oft ist es sinnvoll, den Nutzen als Differenz von zwei Funktionen zu definieren:
  - **einer Wertfunktion** w
  - **einer Kostenfunktion** c

$$u(r) = w(r) - c(r)$$

- **Generelle Probleme mit nutzenbasierten Verfahren:**
  - **Woher kommen die Zahlenwerte?**
  - **Menschen denken nicht in Nutzenkategorien, das Formulieren von Aufgaben damit ist schwierig**

Beispiel: Nutzenfunktion in der „Steinwelt“

- **Steinwelt:**
  - **zweidimensionale Gitterumgebung mit Agenten, Steinen, Hindernissen und Löchern**
  - **Die Steinwelt verändert sich, indem Löcher dynamisch auftauchen oder verschwinden**
- **Aufgabe des Agenten: möglichst viele Steine in Löcher schieben**
  - **kann sich in vier Richtungen bewegen (links, rechts, hoch, runter);**
  - **wenn er auf dem Nachbarfeld zu einem Stein steht, kann er ihn verschieben, indem er auf das Feld des Steins zieht**
- **Nutzenfunktion eines Durchlaufs *r*: Quotient aus der Anzahl der in *r* gefüllten Löcher und der Gesamtzahl der Löcher in *r***

Erwarteter Nutzen und Optimalität

- **Wir bezeichnen mit P(r|Ag, Env) die Wahrscheinlichkeit für das Auftreten von Durchlauf r, wenn Agent Ag in der Umgebung Env gestartet wird**
- **Es gilt:**

$$\sum_{r \in R(Ag, Env)} P(r \mid Ag, Env) = 1$$

- **Ein Agent Ag* ist optimal in einer Umgebung Env ist, wenn er den erwarteten Nutzen maximiert:**

$$Ag^* = \arg\max_{Ag \in AG} \sum_{r \in R(Ag, Env)} u(r) P(r \mid Ag, Env)$$

Beschränkt optimale Agenten

- Manche Typen von Agenten können auf manchen Computern nicht implementiert werden (z.B. kann eine Funktion $Ag : R^E \rightarrow Ac$ mehr Speicher brauchen, als auf dem Computer verfügbar ist)
- Sei Ag_m die Menge der Agenten, die auf Computer m implementiert werden können. Dann ist ein beschränkt optimaler Agent Ag^* definiert als:

$$Ag^* = \arg \max_{Ag \in AG_m} \sum_{r \in R(Ag, Env)} u(r) P(r \mid Ag, Env)$$

Aufgabenumgebungen

- Spezialfall für die Zuweisung von Nutzenwerten zu Durchläufen ist der Boole'sche Fall:
  - u(r) = 1: der Durchlauf ist erfolgreich
  - u(r)=0: der Durchlauf schlägt fehl
- Aufgabenspezifikationen mit Boole'schen Nutzenwerten heißen prädikative Aufgabenspezifikationen Ψ mit $\Psi: R \rightarrow \{0,1\}$
- Eine Aufgabenumgebung ist ein Paar (Env, Ψ), wobei Env eine Umgebung und $\Psi: R \rightarrow \{0,1\}$ ein Prädikat über Durchläufen ist
- Sei TE die Menge der Aufgabenumgebungen
- Eine Aufgabenumgebung definiert die Eigenschaften eines Systems, in dem ein Agent agiert, sowie die Kriterien, nach denen Erfolg und Fehlschlag des Agenten bewertet wird.

- Sei $R_\Psi(Ag, Env)$ die Menge von Durchläufen eines Agenten Ag in der Umgebung Env, die Ψ erfüllen:

$$R_\Psi(Ag, Env) = \{r \mid r \in R(Ag, Env) \text{ mit } \Psi(r) = 1$$

- Wir definieren: Ein Agent ist erfolgreich in einer Aufgabenumgebung (Env, Ψ), wenn:

$$R_\Psi(Ag, Env) = R(Ag, Env)$$

D.h., Ag ist erfolgreich, wenn jeder Durchlauf von Ag in Env die Spezifikation Ψ erfüllt (pessimistische Definition)

Erfolgswahrscheinlichkeit

- Erweiterung des Modells: Zustandstransformationsfunktion τ wird erweitert um eine Wahrscheinlichkeitsverteilung über mögliche Zustände
- Sei $P(r \mid Ag, Env)$ die Wahrscheinlichkeit für das Auftreten von Durchlauf r für Agent Ag in Umgebung Env.
- Dann definieren wir die Erfolgswahrscheinlichkeit
  $P(\Psi \mid Ag, Env)$
  für Env in Ag bezüglich der prädikativen Aufgabenspezifikation Ψ durch die Wahrscheinlichkeit, dass Ψ durch den Agenten erfüllt wird:

$$P(\Psi \mid Ag, Env) = \sum_{r \in R_\Psi(Ag, Env)} P(r \mid Ag, Env)$$

Aufgabentypen

Die beiden gebräuchlichsten Typen von Aufgaben sind:

- **Aufgaben der Form: „Erreiche Zustand Φ“ (engl. *achievement tasks*)**
- **Aufgaben der Form: „Bewahre Zustand Φ“ (engl. *maintenance tasks*)**

- **Eine *achievement task* ist definiert durch eine Menge G angestrebter Zustände (Zielzustände): $G \subseteq E$**
- **Ein Agent ist erfolgreich in einer Umgebung, wenn er (garantiert) zumindest einen dieser Zustände erreicht (hierbei wird nicht zwischen unterschiedlichen Zielzuständen unterschieden)**
- **Eine *maintenance task* ist definiert durch eine Menge F von Fehlerzuständen: $F \subseteq E$**
- **Ein Agent ist erfolgreich in einer Umgebung, wenn er (garantiert) alle Fehlerzustände vermeidet, d.h.: er führt nie eine Aktion aus, die ihn in einen Zustand $f \in F$ bringt.**

Lerninhalte dieses Kapitels

- **Motivation: Warum beschäftigen wir uns mit Agententechnologie?**
- **Definition: Was sind die Haupteigenschaften von Agenten?**
- **Was sind die Bestandteile einer einfachen formalen Agentenarchitektur?**