Zusammenfassung - Multi-Agenten-Systeme

Andreas Ruscheinski,

3. Februar 2016

Korrektheit und Vollständigkeit der Informationen sind nicht gewährleistet. Macht euch eigene Notizen oder ergänzt/korrigiert meine Ausführungen!

Inhaltsverzeichnis

1 Einführung **1**

2 Rolle der Logik in MAS **2**

3 Planning **4**

4 Kooperation von Agenten **7**

5 Agenten Kommunikation **9**

6 Wie sollten Agenten kommunizieren? Strategien und Protokolle **12**

7 Verhandlungen **13**

8 Trust and Reputation **15**

9 Blackboard **16**

10 Organisationskontrolle **17**

11 Agenten und Mobilität **17**

1 Einführung

1.1 Definition

- Ein Agent ist ein Computer System welches **selbstständig** Aktionen im Interesse des Benutzers ausführen kann.
- Ein Agent **befindet** sich in einer **dynamischen Umgebung**, mit welcher er interagiert.
- Ein Multi-Agenten-System besteht aus **mehreren Agenten**, welche **miteinander interagieren**.
- In einem Multi-Agenten-System ist es notwendig für **erfolgreiche Interaktion**, dass Agenten miteinander **kooperieren**, **sich abstimmen** und miteinander **verhandeln** können.

1.2 Eigenschaften

- Jeder Agent hat **keine vollständigen Informationen** über die Umgebung
- Es gibt **keine globale Kontrolle** der Agenten
- Die Daten sind **dezentralisiert**
- Die Berechnung erfolgt **asynchron**

1.3 Gründe für den Einsatz von MAS

- Problem kann nicht zentralisiert gelöst werden, da die **Ressourcen limitiert** sind
  →*Verteilte Berechnungen*
- **Reduktion der Ausfall-Wahrscheinlichkeit** gegenüber einem zentralisierten System
- **Gewährleistung der inter-konnektion und inter-operation** von verschiedenen Systemen
  →*Migration von veralteter Software*
- Lösung von Problemen, welche eine **Menge aus autonomen Komponenten behandeln**
  →*Luftfahrkontrolle, Terminkalender*

1.4 Konkrete Anwendungsgebiete

- Cloud-Management
- Ubiquitous Computing
- Grid-Software
- Spiele
- Verschiedene Gebiete der Industrie (Car-Assembly, Factory Management)
- Simulation

MAS befindet sich in der Schnittmenge aus KI, Spieltheorie, Sozialforschung und Verteilten Systemen.

2 Rolle der Logik in MAS

2.1 Gründe für Logik

- Wissensbasis + Aktionen mit Voraussetzung und Auswirkung → Plan für Lösung des Problems
- Verwaltung von Annahmen
- Logik ist ein Framework für das Verstehen von Systemen
- Verifikation, Ausführungsspezifikation, Planung
- Agenten als Theorembeweiser

2.2 Logik-basierende Architektur

- Grundidee: Beschreibung einer Regelmenge für die Beschreibung der besten Aktion für einen gegeben Zustand
- Bestandteile:
  - p: eine Theorie (eine Menge von Regeln)
  - Δ: Datenbank mit den aktuellen Zustand der Welt
  - A: eine Menge von Aktionen, welche ein Agent ausführen kann
  - $\Delta \vdash_p \phi$: d.h. ϕ kann aus der Δ unter der Verwendung von p abgeleitet werden, mit ϕ =Do(a) können wir aus den aktuellen Zustand der Welt auf die bestmögliche Aktion logisch schließen
- Algorithmus-Bestandteile (unabhängig von verwendeter Logik)
  - see(s,p), generiert Beobachtung aus der neuen Welt (schwer siehe Vorlesung Mustererkennung und Kontextanalyse)
  - next(Δ,p), Update der Datenbank mit der Beobachtung
  - action(Δ), ermittelt die auszuführende Aktion aus der Datenbank, entweder ist die Aktion direkt beschrieben oder kann aus den Regeln abgeleitet werden kann
- Algorithmus - Ermittelung einer Aktion aus einer Wissensdatenbank:
  1. Überprüfe für jede Aktion a ob $\Delta \vdash_p Do(a)$ gilt (d.h. a kann direkt abgeleitet werden)
  2. Falls ja: gebe a zurück sonst
  3. Überprüfe für jede Aktion a ob $\Delta \nvdash_p \neg Do(a)$ gilt (d.h a ist nicht explizit ausgeschlossen)
  4. Falls ja: gebe a zurück sonst gebe NULL zurück (keine Aktion gefunden)

2.3 Modale Logik

- Erlaubt Ausdrücke wie: wahrscheinlich wahr, geglaubt wahr, wahr in der Zukunft usw.
- kann genutzt werden um Informationen für die Agenten abzuleiten und über das Wissen von Agenten zu schließen
- Syntax:
  - Prädikatenlogik mit Erweiterung
  - Prop: eine Menge von atomaren Formeln
  - wenn $p, q \in Prob$ dann sind auch $\neg p, p\&q, \diamond p, \Box p$ Formeln
  - $\diamond$ p: möglicherweise p, manchmal p
  - $\Box$p: immer p, notwendigerweise p
- Semantik:
  - Kripke-Struktur: $< W, R, \mu >$
    W eine Menge von Welten
    R eine Menge von binären Relationen, beschreiben den Übergang zwischen den Welten
    μ Abbildungsfunktion welche jeder Welt Eigenschaften zuordnet ($\mu : W \to 2^{Prop}$)
  - Definition von $\diamond$ und $\Box$ Operator auf Basis von Erreichbarkeit der Welten in einer Kripke-Struktur
  - $\Box p$: p ist wahr in allen Welten, welche von der aktuellen Welt erreichbar sind
  - $< M, w > \models \Box p$ wenn für alle $w' \in W$ gilt: wenn $(w, w') \in R$ denn $< M, w' > \models p$
  - $\diamond p$: p ist wahr, wenn mindestens eine Welt erreichbar in welcher p wahr ist
  - $< M, w > \models \diamond p$ wenn es ein $w' \in W$ existiert mit: wenn $(w, w') \in R$ denn $< M, w' > \models p$
  - für R muss zusätzlich gelten:
    reflexiv für jedes $x \in W$ gilt $R(x, x)$ d.h. x ist von x aus erreichbar
    transitiv für jedes $x, y, z \in W$ gilt $R(x, y) \wedge R(y, z) \implies R(x, z)$ d.h. wenn man von x nach y und von y nach z gehen kann, kann man auch von x nach z gehen
    seriell für jedes $x \in W$ existiert ein y so dass gilt $R(x, y)$ d.h. für jede Welt ist mit einer anderen Welt in Relation
    euklidisch wenn für jedes $x, y, z \in W$ mit $R(x, y)$ und $R(x, z)$ gilt auch $R(y, z)$ d.h. wenn man von x nach y und von x nach z gehen kann dann kann man auch von y nach z gehen
  - Axiome (Beschreiben Eigenschaften für R):
    $\Box p \Rightarrow p$ Wenn immer p gilt folgt daraus das aktuell p gilt (Reflexiv)
    $\Box p \Rightarrow \diamond p$ Wenn p immer wahr ist, ist p auch in mindestens einer Welt wahr (Seriell)
    $\Box p \Rightarrow \Box\Box p$ Wenn p ist immer wahr, ist p auch immer wahr wenn wir einen Übergang machen (transitiv)
    $\diamond p \Rightarrow \Box \diamond p$ Wenn p in mindestens einer Welt wahr ist, ist p für immer Wahr wenn wir diese Welt erreicht haben (Euklidisch)
  - Anwendung der modal Logik auf Agenten durch Einführung von Indizes, welche entsprechend für Agent gelten
  - Operator - Knowledge: $K_i p$ bedeutet dass Agent i weiss p
  - der Modale Operator $\Box_i$ wird zu K_i
  - Axiome und Agenten:
    $K_i p \Rightarrow p$ Wenn Agent glaubt das p wahr ist, p ist auch in Wirklichkeit wahr
    $K_i p \Rightarrow \neg K_i \neg p$ Wenn der Agent p glaubt, glaub er nicht die Negation
    $K_i p \Rightarrow K_i K_i^p$ Wenn der Agent p glaubt, weiß er selbst dass er p glaubt
    $\neg K_i \neg p \Rightarrow K_i \neg K_i \neg p$ Wenn der Agent nicht p glaubt, weiß er dass er nicht p glaubt.
  - es gilt: $< M, w > \models K_i p$ genau dann wenn $\forall w' \in W$ gilt: wenn $(w, w') \in R_i$ denn $< M, w' > \models p$ (vgl. Definition von $\Box$)
  - Probleme:
    * Agent glaubt alles was wahr ist incl. Tautologien: wenn $\models p$ denn $K_i p$
    * Agent weiss alle Schlussfolgerungen des eigenen Wissens: $\models p \to q$ denn $\models K_i p \to K_i q$ (Kripke Axiom: $K_i(p \to q) \to (K_i p \to K_i q)$)
    * Menschliches wissen ist oft inkonsistent
    * Menschen glauben nicht alle equivalenten Aussagen

2.4 BDI-Logic

- Klassische logische Operatoren: und, oder, Negation usw...
- CTL* Path Quantifikatoren
  - $A\phi$: es gilt auf allen Pfaden ϕ
  - $E\phi$: es gilt auf einigen Pfaden ϕ
  - G: global
  - F: zukünftig
- BDI-Funktionen:
  - (Bel i ϕ): i glaubt ϕ
  - (Des i ϕ): i will ϕ
  - (Int i ϕ): i verfolgt ϕ
- Regeln:
  - (Des α) $\rightarrow$ (Bel α) (Belief goal compatibility): Wenn der Agent α erreichen will folgt daraus, dass der Agent an die Machbarkeit von α glaubt
  - (Int α) $\rightarrow$ (Des α) (Goal-intention compatibility): Wenn der Agent α verfolgt, folgt daraus, dass der Agent α erreichen will
  - (Int does(a)) $\rightarrow$ does(a) (Volitional commitment): Wenn der Agent does(a) ausführen möchte, wird er diese als nächstes ausführen
  - (Des α) $\rightarrow$ (Bel (Des α)) und (Int α) $\rightarrow$ (Bel (Int α)) (Awareness of goals & intentions): Bedingt dass neue Ziele und Intentionen als Event gepostet werden müssen in der Wissensbasis
  - done(a) $\rightarrow$ (Bel (done(a))) (No unconscious actions): Wenn ein Agent eine Aktion ausgeführt hat weiss er das er diese ausgeführt hat
  - (Int α) $\rightarrow$ AF($\neg$(Int α)) (No infinite deferral): Ein Agent willeine Intention verfolgen oder diese verwerfen
- reaktive und proaktive Agenten
  - reaktiv: Aktion wird ausgeführt durch eine Eingabe: wenn Eingabe = p denn tue a
  - proaktiv: Planung und Ausführung von Aktionen um ein Ziel zu erreichen: tue a um p zu erreichen

3 Planning

3.1 Einführung - Brooks Vision

- Ziel 1: Intelligentes Verhalten ohne explizite Repräsentation des Wissens
- Ziel 2: Intelligenten Verhalten ohne abstraktes Schließen über die Repräsentation des Wissens
- Idee 1: Echte Intelligenz gibt es nur ein einer Welt und nicht losgelöst von dieser wie, in Theorem Beweisern und Expertensystemen
- Idee 2: Intelligentes Verhalten entsteht erst als Ergebnis der Interaktion mit der Umgebung

3.2 Subsumption Architektur

- Traditionell: Die Intelligenz steht zwischen Beobachtung und Aktion d.h. aus Beobachtungen wird geschlossen welche Aktion ausgeführt wird
- Neu: Die Intelligenz entsteht beim Beobachter durch die Aktionen in der Welt d.h. ein Agent ist nicht an sich intelligent sondern wirkt intelligent für einen Beobachter
- Entscheidungsfindung durch verschiedene Aufgaben:
  - Verhalten ist eine Abbildung von Zustand auf Aktion
  - Verarbeitung der Sensorwerte mit Schluss auf den Zustand
  - Zustände und Aktionen sind direkt gekoppelt

- Mechanismus für die Auswahl der Aktionen: Prioritäten d.h Verhalten mit hoher Priorität kann Verhalten mit niedrigeren Prioritäten unterdrücken
- Formales Modell:
  - Ein Verhalten (Behavior) $b \in Beh$ ist ein Tupel (c, a) mit $c \subseteq P, a \in A$ wobei P ist eine Menge von Beobachtungen und A eine Menge von Aktionen
  - Ein Verhalten wird ausgeführt, wenn die Umgebung sich in dem Zustand $s \in S$ befindet und $see(s) \in c$
  - Subsumption Hierachie wird realisiert durch eine Hemmungs-Relation $b_1 \prec b_2$ (b_1 hemmt b_2 d.h. b_1 hat eine höhere Priorität)
- Aktionsauswahl-Algorithmus:
  1. Berechne die Menge von aktivierbaren Aktionen $FB = \{(c, a) | (c, a) \in Beh \wedge see(s) \in c\}$
  2. für jede Aktion in FB überprüfe, ob es eine Aktion in FB gibt welche eine höhere Priorität hat
  3. wenn es keine Aktion mit höherer Priorität gefunden wurde: gib a zurück sonst: NULL (Keine Aktion gefunden)
- Vorteile:
  - Einfach und hohe Ausdrucksfähigkeit
  - Die Berechnung ist einfach nachzuvollziehen
  - Robust gegen Ausfälle
  - Das gesamte Verhalten entsteht durch Interaktion mit der Umwelt
- Nachteile:
  - Verhalten wird hard codiert unter der Annahme die Umgebung genau zu kennen
  - Schwierige Entscheidung über das Standardverhalten
  - langwierige Entscheidungen schwer möglich
  - skaliert nicht in größeren Systemen

3.3 Planung

- Grundideen:
  - Beschreibung des Zieles (Intention) welches erreicht werden soll
  - Beschreibung der Aktionen, welche ausgeführt werden können
  - Beschreibung der Umgebung
  - Beschreibungen + Planner = Plan welches das Ziel erreicht
- Umsetzung durch z.B STRIPS Planner
  - Repräsentation der Umgebung durch Ontologie (Begriffe + Relationen)
  - Beschreibung der aktuellen Welt durch Verwendung der Ontologie Begriffe (closed world assumption: alles was nicht angegeben wird ist falsch)
  - Jede Aktion hat Name, Pre-Condition List (alle Bedingungen müssen wahr sein bevor Aktion ausgeführt werden kann), Delete-List (Bedingungen welche nach der Ausführung nicht mehr wahr sind), Add-List (Bedingungen welche nach der Ausführung der Aktion gelten) (können alle Variablen erhalten für allgemeine Aussagen)
- Ein Plan ist eine Liste von Aktionen, mit Variablen ersetzt durch Konstanten. Die Ausführung der Aktionen führt vom aktuellen Zustand in ein einen Zustand, welcher das Ziel erfüllt. Der Plan ist vollständig (keine weiteren Aktionen notwendig) und konsistent (alle Pre-Conditions sind erfüllt) und die Schritte können hintereinander ausgeführt werden ohne dass die Ausführung eines Schrittes beeinflusst wird.
- Formal: partially ordered Plans
  - Plan Schritt mit partieller Ordnung $\prec$: $S_i \prec S_j$ bedeutet dass S_i vor S_j ausgeführt werden muss
  - Eine Menge von variablen Zuordnungen $x = t$ mit x ist eine Variable und t ist eine Konstante
  - Eine Menge von kausalen Relationen: $S_i \rightarrow S_j$ bedeutet die Ausführung S_i macht die Vorbedingungen von S_j wahr (impliziert $S_i \prec S_j$)

- Formal: Eigenschaften Konsistenz und Vollständigkeit
  - Vollständigkeit:
    - es gilt: $\forall S_j$ mit $c \in Precond(S_j)$ und $\exists S_i$ mit $S_i \prec S_j$ und $c \in Effect(S_i)$ (die Vorbedingungen für S_j sind Teil des Effektes von S_i)
    - für eine Sequenz gilt:$\forall S_k$ mit $S_i \prec S_k \prec S_j, \neg c \notin EffectS_k$
  - Konsistenz: Wenn $S_i \prec S_j$ denn $S_j \prec S_i$ und wenn $x = A$ denn $x \neq B$ für verschiedene A und B für die Variable x.
- Vorlesungfolien für Beispiel!!!
- iterative Erstellung eines Plans durch rückwärts anwenden der Regeln d.h. in jedem Schritt wird eine offene Bedingung durch die entsprechende Aktion erfüllt
- dadurch können Konflikte entstehen
  - Ein Konflikt gdw. wenn S_3 bedroht die kausale Ordnung zwischen S_1 und S_2
  - Lösung 1: Demotion - S_3 vor S_1 und S_2 ausführen
  - Lösung 2: Promotion - S_3 nach S_1 und S_2 ausführen
  - sollte dies wieder zu Konflikte führen Backtrack und eine andere Lösung ausprobieren in dem S_3 neben S_1 zu S_2 eingeordnet wird

3.4 Planning Agents

- Erster Ansatz für den Planning Agent:
  1. beobachte die Umgebung
  2. aktualisiere das interne Modell der Umgebung
  3. ermittle welche Intention als nächstes erreicht werden soll
  4. benutze means-end Resoning für die Erstellung des Plans welche die Intention erreicht
  5. führe Plan aus
- means-end Resoning: Gib den Agent eine Repräsentation der Ziele/Intentionen welche erreicht werden sollen, Aktionen welche er ausführen kann, der Umgebung → Agent nutzt Repräsentationen um einen Plan zu generieren
- Problem: Planung und Ermittelung welches Ziel als nächstes erreicht werden soll kosten Zeit
- dadurch kann die Situation entstehen dass der Agent ein Ziel erreichen will welches nach dessen Ermittelung nicht mehr optimal ist
- Unter folgenden Annahmen ist die getroffene Entscheidung noch immer optimal: wenn Planung und Ermittelung schnell genug sind; die Welt sich nicht verändert hat; getroffene Zielentscheidung ist noch immer optimal wenn Agenten Plan gefunden hat;
- Agenten Algorithmus formal 1:
  1. B = B_0 - Ausgangs-Beliefs
  2. führe unendlich lange aus:
  3. beobachte Umwelt - p
  4. B = brf(B,p) - Aktualisierung des der eigenen Beliefs
  5. I = deliberate(B) - Ermittelung
  6. π = plan(B,I) - Plane
  7. execute(π) - führe Plan aus
- Ermittelung durch: option-generation (Erstelle mögliche Ziele) und filter (Auswahl des Ziels)
- option-generation: Nutzt aktuelle Beliefs und Intententions und ermittelt daraus eine Menge von Optionen (Desires)
- filter: Der Agent wählt zwischen verschiedenen Alternativen aus den Optionen und beginnt die gewählte Option zu verfolgen

- Agenten Algorithmus formal 2 - BDI Agent:
  1. B = B_0 - Ausgangs-Beliefs
  2. führe unendlich lange aus:
  3. beobachte Umwelt - p
  4. B = brf(B,p) - Aktualisierung des der eigenen Beliefs
  5. D = options(B,i) - Desires (Ziele)
  6. I = filter(B,D,I) - Ermittelung der Intentionen durch Beliefs(Wissen über Umwelt), Desires(Ziele) und Intentions(gewählten Ziel)
  7. π = plan(B,I) - Plane
  8. execute(π) - führe Plan aus
- Begriffserklärungen:
  - Beliefs - Weltwisssen (Alles was wir wissen d.h. was wir glauben über die Welt, unseren Fähigkeiten und Zielen)
  - Desires - Ziele (Optionen welche wir gerne erfüllt hätten)
  - Intentions - Absicht (Ziel welches ich jetzt erreichen möchte)

3.5 Commitments

- Strategien:
  - Blind Commitment - Agent will Intention erreichen bis er glaubt die Intention erreicht zu haben
  - Single-minded Commitment - Agent will Intention erreichen bis er glaubt diese erreicht zu haben oder es nicht mehr möglich ist
  - Open-minded Commitment - Agent will Intention solange er glaubt dieser erreichen zu können
- Anpassungen des Agenten Algorithmus
  1. wie bisher → Blind Commitment
  2. wie bisher - Ermittle Plan und solange dieser nicht leer ist führe diesen Schrittweise aus, nach jedem Schritt hole neuen Percept + Belief Update: wenn Ziel nicht mehr erreicht werden kann ermittle neuen Plan → Single-minded da keine neuen Intentionen erwogen werden
  3. wie bisher - Ermittle Plan und solange dieser nicht leer ist führe diesen Schrittweise aus solange Intention nicht erreicht und dieser noch nicht unmöglich ist, nach jedem Schritt hole neuen Percept + Belief Update + Desires ermitteln + Intentions ermitteln :wenn Ziel nicht mehr erreicht werden kann ermittle neuen Plan
- Problem: Agent will Intentions verfolgen selbst wenn schon klar ist das Ziel nicht mehr erreicht werden kann und Agent will konstant berücksichtigten dass evtl unnötige Zeit verschwendet wurde und deshalb nie dass Ziel erreichen
- Lösung: Meta-Level Control welches entscheidet wann der Agent seine Intention verwirft

4 Kooperation von Agenten

4.1 Einführung

- Ein Multiagenten System besteht aus mehren Agenten welche
  - Interagieren durch Kommunikation
  - Mit der Umgebung interagieren können
  - Haben unterschiedliche Einflussbereiche auf die Umgebung
  - stehen in Beziehungen zu einander (Organisation)
- Egoistische Agenten bergen Potential für Konflikte da jeder seine Ziele verfolgen möchte
- Lösung: Kooperation mittels Entscheidungstheorie

4.2 Entscheidungstheorie - Spieltheorie

- Wir haben zwei Agenten A_i und A_j
- jeder Agent hat die gleiche Menge $\Omega = \{\omega_1, \omega_2, \dots\}$ eine Menge von Ergebnissen welche für den Agenten von Bedeutung ist
- die Bedeutung für einen Agent i wird durch eine Utilty-Funktion repräsentiert $u_i : \Omega \to \mathbb{R}$ (jeder Agent hat eine Utility-Funktion)
- Utility-Funktion bildet Ordnung der Ergebnisse: $\omega \geq_i \omega'$ bedeutet also $u_i(\omega) \geq u_i(\omega')$
- Utility beschreibt den Nutzen nicht GELD!
- Modell für mehrere Agenten:
  - Agenten wählen Aktion gleichzeitig aus, als Ergebnis etwas aus Ω
  - das aktuelle Ergebnis hängt von Kombinationen von Aktionen ab
  - Jeder Agent hat nur zwei mögliche Aktionen: C und D
  - $\tau : A_c \times A_c \to \Omega$ eine Funktion welche die Aktion von den beiden Agenten auf ein Ergebnis abbildet
  - Eine Welt lässt sich mittels τ beschreiben unter Verwendung aller möglichen Kombinationen von Aktionen $\to$ hat ein Ergebnis
  - jeder Agent kann die Utility von dem Ergebnis berechnen
- Darstellung mittels Payoff Matrix (Ein Agent ist Spalte (linker Rechter), ein Agent ist Zeile (linker Eintrag)

	D	C
D	1,1	1,4
C	4,1	4,4

- Beispiel: $\tau(D, C) = \omega_1$ mit $u_i(\omega_1) = 1$ und $u_j(\omega_1) = 4$

4.3 Strategien

- Dominate-Strategie
  - Aus einer gegebenen Strategie s (Beispiel: D oder C) für Agent i ergeben sich verschiedene Ergebnisse (ω)
  - Strategie s_1 dominiert Strategie s_2 wenn jedes Ergebnis wenn s_1 gespielt wird bevorzugt wird gegenüber s_2
  - Rationale Agenten werden niemals dominierte Strategien spielen d.h. es gibt eine Strategie welche besser ist
  - Ziel: dominierte Strategie eliminieren
  - Anmerkung: es ist nicht immer möglich nur eine nicht dominierte Strategie zu finden
- Nash-Equilibrium (Nash-Gleichgewicht):
  - Strategie s_i und s_j sind ein einem Nash-Gleichgewicht gdw: Unter der Annahme dass Agent i spielt s_i und Agent j kann nicht besser sein als wenn er s_j spielt UND unter der Annahme dass Agent j spielt s_j und Agent i kann nicht besser sein als wenn er s_i spielt
  - Aber: nicht jede Interaktion hat ein Nash-Gleichgewicht und einige Interaktionen haben mehrere Nash-Gleichgewichte
- Pareto Optimum:
  - Gegeben einer initialen Zuordnung von Gütern für eine Menge von Agenten
  - Eine Änderung welche für einen Agent besser ist aber für keinen anderen Agenten schlechter nennt sich Pareto-Verbesserung
  - Eine Zuweisung ist Pareto-Optimal gdw. keine weiteren Pareto-Verbesserungen gemacht werden können
- Kompetitive- und Null-Summen-Interaktionen
  - Szenarios in denen Agenten gegenläufige Präferenzen haben sind streng kompetitiv
  - Null-Summen Spiele gdw. die Summe aller Utilitys der Agenten = 0 sind: $u_i(\omega) + u_j(\omega) = 0, \forall \omega \in \Omega$

  - Null-Summen Spiele sind streng kompetitiv, im Real-Life selten
- Beispiel: Prison Dilemma
  - zwei Männer wurden verhaftet wegen Diebstahl
  - wenn jmd gesteht und der andere nicht geht der Geständige 3 Jahre ins Gefängnis und der andere kommt frei
  - wenn beide gestehen beide gehen für 3 Jahre ins Gefängnis
  - Wenn keiner gesteht geht jeder 1 Jahr ins Gefängnis
  - Nash-Equilibirum gdw. jeder gesteht
  - Besser wäre aber wenn jeder schweigt
- Grundlegendes Problem von Multi-Agenten Interaktionen: es wird keine Kooperation entstehen wenn jeder Agent egoistisch ist
- Lösung: Annahme mein Gegenspieler ist meine Zwilling, oder Shadow of the Future (nochmaliges treffen in der Zukunft)
- Iterated Prison Dilemma
  - Lösung: Spiele das Spiel mehrmals
  - Rückwärts Indutkion: Annahme wir spielen n mal, in Runde n-1 wollen wir schweigen für höheren Payoff, dadurch wird n-2 zur letzten richtigen Runde wo wir auch schweigen würden für höheren Payoff ... Prison-Dilemma mit fixer Anzahl an Runden ist schweigen immer die beste Strategie
  - Untersuchung in Axelrods Tournament: Iterated Prison Dilemma gegen verschiedene Gegner
  - Verschiedene Stragiegien: ALLD (always defect), TIT-FOR-TAT (Kooperation in der ersten Runde, danach immer die Aktion welche der Gegner gespielt hat), TESTER, JOSS

4.4 Benevolent Agents

- Wenn wir das ganze System kontrollieren können wir Agenten bauen welche sich gegenseitig unterstützen
- Es gibt Konventionen über das Verhalten
- Unter dieser Annahme: Unsere Agenten sind mehr oder weniger gutmütig d.h. unser bestes Ziel ist deren bestes Ziel
- gutmütige Agenten vereinfachen das System-Design
- gutmütige Agenten haben eigene Interesse kollidieren aber nicht mit anderen Interessen

5 Agenten Kommunikation

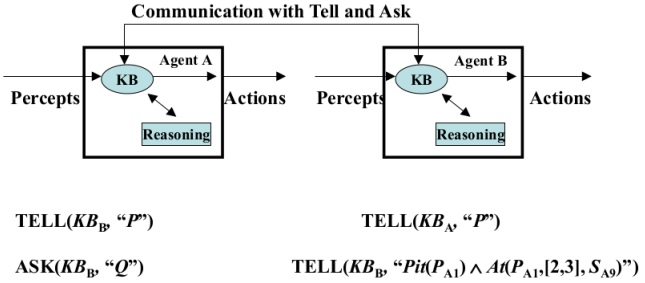
5.1 Einführung

- Kommunikation ist der Austausch zwischen Informationen
- für Kooperation ist es notwendig zu kommunizieren: Speech Acts, KQML & KIF, FIPA ACL

5.2 Speech Acts

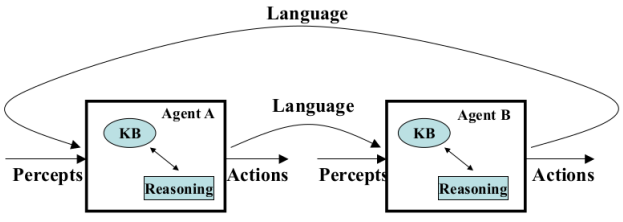
- Behandlung der Kommunikation bei multi Agenten ist häufig angelegt an die Speech-Act-Theory
- Alles was wir äußern wird geäußert um ein Ziel oder eine Intention zu erfüllen
- Speech Act Theorie: Wie Äußerungen genutzt werden um eine Ziel zu erreichen
- Verschiedene Typen von Speech Acts (Searle):
  - representives: informieren zb. Es regnet
  - directives: versucht den Höher zu etwas zu bewegen zb. Bitte mach mir Tee
  - commisives: Sprecher verspricht etwas zu tun zb. Ich verspreche ...
  - expressives: Sprecher teilt seinen mentalen Zustand mit z.b Danke
  - declarations: wie Krieg der Erklären oder Taufe

- jeder Speech-Act hat zwei Komponenten: performative Verb (anfordern, informieren, ...) + propositional content (die Tür ist verschlossen)
- Communication as Action: den Hörer zu einer Aktion zu bewegen
  - Versprechen: Sachen anbieten oder Sachen zu tun (Promise)
  - Anfrage: Anderen Agenten etwas für über die Gruppe fragen (Query)
  - Bitte: Anderen Agenten bitten etwas zu tun (Request)
- Probleme mit Kommunikation: Wann soll kommuniziert werden? Welcher Speech-Act ist der richtige für die aktuelle Situation?
- Probleme mit Verstehen: Welche Situation kann dieses Speech-Act erzeugt haben?
- Komponenten der Kommunikation:
  - am Sprecher
    1. Intention: S möchte dass H an P glaubt
    2. Generation: S wählt Worte W
    3. Synthese: S kommuniziert die Worte W
  - am Hörer
    1. Perception: H empfängt W^1 (ideal $W^1 = W$)
    2. Analysis: H schließt aus W^1 mögliche Bedeutungen $P_1, \ldots, P_n$
    3. Disambiguation: H schließt dass S P_i mitgeleit werden möchte (ideal $P_i = P$)
    4. Intercorporation: H entschließt P_i zu glauben (oder verwirft es wenn es nicht mit dem aktuellen Glauben zusammen passt)
- Modelle der Kommunikation
  - Encoded Message Model: Sprecher kodiert Nachricht in Wörter, Hörer versucht Nachricht zu dekodieren; Die Bedeutung der Sprecher Nachricht, die Übertragene Nachricht und die Interpretion sind gleich (falls keine Probleme der Kommunikation auftreten)
  - Situated Langage Model: Bedeutung der Nachricht beruht nun auf er Nachricht und der Situation
- Typen von kommunizierenden Agenten
  - Kommunikation unter Verwendung von Tell und Ask: Agenten teilen sich eine gemeinsame interne Repräsentation einer Sprache; Kommunikation ohne Verwendung einer externen Sprache



and ask.png

  - Kommunikation unter Verwendung von Formalen Sprachen: Agenten machen sich keine Annahmen über die interne Repräsentation der Sprache; Agenten teilen eine gemeinsame Sprache für die Kommunikation



language.png

- Größtes Problem: Kommunikation ist zweideutig; Lösung durch Berücksichtigung des Contexts und der vorigen Kommunikationen
- Definition der Semantik auf Basis von Planung: bestimmte Prädikate welche in der jeweiligen precondition-delete-add Listen verwaltet werden
- Beispiel: request(s,h,ϕ)
  - pre: s glaubt h kann ϕ tun; s glaubt h glaubt h kann ϕ; s glaubt s will ϕ
  - post: h glaubt s glaubt s will ϕ

5.3 KQML und KIF

- nun Betrachtung von Agent Communication Languages (ACLs) d.h. standard Formate für den Austausch von Nachrichten
- bekannteste ACL: KQML
- zwei Bestandteile: Wissens-Abfrage und -Manipulations Sprache (KQML) und Wissens-Austausch Format (KIF)
- KQML ist eine äußere Sprache welche verschiedene akzeptierbare Verben (performatives) definiert (Bsp: ask-if, perform, tell, reply)
- KIF ist eine Sprache für die Repräsentation des Nachrichteninhaltes (meistens sowas wie Common Lisp)
- für die Kommunikation müssen die Agenten das gleiche Verständnis von Ausdrücken haben (Was bedeutet Schalter1?), formale Definition in einer Ontology

5.4 FIPA

- neue Entwicklung der Agenten Standards inc. ACL
- Basisstrukltur ist ähnlich zu KQML: Permative + Housekeeping + Content
- INFORM und REQUEST sind die zwei Standard Permative in FIPA; der Rest wird auf Basis dieser definiert
- Bedeutung von INFORM und REQUEST hat zwei Parts: pre-condition (muss wahr sein damiot der Speech Act erfolgreich ist), rational effect (Ziel welches der Sender der Nachricht erreichen möchte)
- Beschreibung der pre-condtions usw als logischer Ausdruck

5.5 Ontologien

- Situation: Service Oriented Computing
- Alle Teilnehmer haben ein gemeinsames, gleiches Verständniss von Begriffen über eine Domäne (Ontologie) d.h. jeder Teilnehmer weiß genau was unter einem bestimmten Begriff zu verstehen ist
- Eine Ontologie beschreibt Begriffe und deren Relationen zu einander
- Ontologien unterstützen die Interaktion dadruch dass sie Begriffe eine Bedeutung geben
- Ontologien-Sprachen erlauben den User eine explizite und formale Konzeptionierung der Domäne
- Problem: je umfangreicher die Sprache desto schwieriger wird es daraus zu schließen
- Schließen in einer Ontology
  - Klassenzugehörigkeit: Wenn x in der Klasse C und C eine Unterklasse von D, ist x auch in der Klasse D
  - Klassenequivalenz: Wenn Klasse A ist equivalent zu B und B zu C denn ist A equivalent zu C
  - Konsistenz: Wir haben X Instanzen von A und B aber A und B sind disjunkt d.h. Fehler in der Ontologie
  - Klassifikation: gegeben Eigenschaft-Werte-Paare für Klasse A, wenn x die Kriterien erfüllt ist x in A
- Schließen ist wichtig für: Überprüfung der Ontologie und des Wissens; Überprüfung ob ungewollte Beziehungen der Klassen entstanden sind; automatische Zuordnung von Instanzen zu Klassen
- Überprüfungen sind wertvoll für: den Entwurf von großen Ontologien mit meheren Autoren; Integration und Verteilung anderer Ontologien an/von verschieden Quellen

6 Wie sollten Agenten kommunizieren? Strategien und Protokolle

6.1 Einführung

- Kommunikation ist wichtig für die Verteilung von Aufgaben(Aufteilung von Teilaufgaben) und Ergebnissen (Teilergebnisse)
- Protokolle regeln die Interaktion zwischen Agenten

6.2 Contract Net

- Realisierung von Multi-Cast Kommunikation in Agenten-basierten Systemen
- zwei Rollen: Selector und Contractor
- Contract Net: Aufgabenverteilungs-Protokol für Aufgabenzuweisung
- Schritte:
  1. Recognition: Feststellung des Problems
  2. Announcement: Mitteilung des Problems an alle Teilnehmer
  3. Bidding: Teilnehmer beantworten mit den erwarteten Kosten
  4. Awarding: Auftragsgeber belohnt den ausgewählten Teilnehmer
  5. Expediting: Ausführung
- Recognition
  - Agent nimmt Problem wahr
  - Agent hat Ziel und: merkt dass er das Ziel nicht alleine erreichen kann (Nicht genügend Kapazität); merkt dass er dieses Ziel nicht alleine erreichen möchte (Deadline, Lösungsqualität)
- Announcement
  - Agent sendet Aufgabe incl. Spezifikation der Aufgabe an alle Teilnehmer
  - Spezifikation beinhaltet: Aufgabenbeschreibung, Anforderungen (Deadlines, Qualität der Lösung), Meta-Informationen
  - Announcement ist ein Broadcast
- Bidding
  - Agenten empfangen Announcement und entscheiden sich ob sie für die Aufgabe bieten möchten
  - Faktoren: Agent muss entscheiden ob er in der Lage ist die Aufgabe zu bewältigen; Agent muss die Contraints ermitteln ggf. Preis
  - wenn sie mitbieten wollen, senden Gebot (tender) an Aufgabensteller
- Awarding & Expediting
  - Aufgabensteller muss aus den Geboten auswählen und entscheiden wer belohnt werden soll
  - Ergebniss wird den teilnehmenden Agenten mitgeteilt
  - Erfolgreicher Contractor (Teilnehmer) führt die Aufgabe aus
  - ggf. Erweiterung der Contractor Beziehung um weitere unter Beziehungen: sub-contracting
- Problem: Spezifikation von ...
  - Aufgaben: gemeinsame Sprache der Agenten
  - Lösungsqualität: Anforderungen (Ontologien), Werbungen und Reputation
  - Auswahl von konkurierenden Angeboten: Prefärenzen

7 Verhandlungen

7.1 Einführung

- Wie können Agenten eine Übereinkunft treffen wenn sie sich egoistisch verhalten?
- Worst-Case: Null-Summen-Spiele d.h. keine Übereinkunft möglich ABER meistens die Möglichkeit ein kurzeitige Übereinkunft zu treffen zum allgmeinen Intersse
- Fähigkeiten der Verhandlung notwendig
- Verhandlungen werden durch einen Mechanismus (Protokoll) gesteuert
- Mechanismus = Regeln für das Treffen von Agenten
- Mechanismus-Design: Entwicklung von Mechanismen mit bestimmten Eigenschaften
- Mechanismus-Design Eigenschaften:
  - Konvergenz und garantierter Erfolg
  - Maximierung des Gemeinwohles
  - Parto-Effizienz
  - Individuelle Rationalität
  - Stabilität
  - Einfachheit
  - Verteilung

7.2 Verhandlung - Auktionen

- Auktionen finden statt zwischen einen Auktionators und eine Menge von Bietern
- Ziel: Auktionator möchte eine Zuweisung von Gütern (Ressourcen) an einen Bieter erreichen
- Meistens möchte Auktionator den maximalen Preis erreichen, die Bieter den minimalen
- Auktions-Parameter
  - Güter haben: Privaten Wert, Allgemeinen Wert, Korrelierten Wert
  - Gewinnerermittelung durch: erstes Gebot, zweites Gebot
  - Gebote sind: für alle sichtbar (open cry); verborgen (sealed bid)
  - Gebotsabgabe: einmalig (one shot), aufsteigend, absteigend

7.2.1 Englische Auktion

- höchstes Gebot gewinnt, für alle sichtbar, aufsteigend
- Dominante Strategie: minimale Erhöhung des höchsten Gebotes bis Obergrenze erreicht, falls diese überschritten: Rückzug
- Anfällig: Fluch des Gewinners (Bezahlt meistens zuviel), Lockvögel (Agent arbeitet mit Auktionator zusammen und treibt den Preis künstlich in die Höhe)

7.2.2 Holländische Auktion

- offene Gebote, absteigend
- Auktionator startet mit hohen Startgebot
- Auktionator senkt Preis bis Agent ein Gebot zu dem Preis abgiebt
- Gewinner: Agent mit der Preisabgabe

7.2.3 First-Price Sealed-Bid Auction

- einmaliges Gebot, verborgen
- eine Runde
- Bieter sendet Gebot
- Bieter mit höchstem Gebot gewinnt
- Gewinner bezahlt Preis des höchstem Gebotes
- Beste Strategie: biete weniger als der eigentlich wert

7.2.4 Vickrey Auction

- second preis, sealed bit
- Gewinner mit dem höchstem Gebot zum Preis vom zweit höchstem Gebot
- Beste Strategie: Preisabgabe zum wirklichen Wert
- überbieten wird dominiert durch bieten des echten Wertes: Wenn Bieter höheren Wert als die anderen gewinnt dieser obwohl er überboten hat; wenn Bieter niedrigeren Wert als andere Bieter hat er verloren obwohl überboten hat oder nicht
- unterbieten wird dominiert durch bieten des echten Wertes: wenn Bieter zu gering verliert er; wenn Bieter zu hoch wird er gewinnen
- anfällig für asoziales Verhalten (???)

7.3 Probleme

- Auktionen sind anfällig für Lügen vom Auktionator und Absprache von Bietern
- alle Auktionen sind können manipuliert werden durch Absprache der Bieter
- ein böser Auktionator kann bei der Vickrey Auktion lügen beim zweit höchsten Gebot
- Shills (Lockvögel) können bei der Englischen Auktion den Preis in die Höhe treiben
- Anwendung von Auktionieren: Lastverteilung, Routing, Koordination

7.4 Heterogene und selbst-motivierende Agenten

- kein zentrales Design
- es gibt keinen globalen Nutzen
- sind Dynamisch d.h. neue Typen von Agenten können leicht hinzugefügt werden
- Agenten sind nicht wohlwollend solange sie es nicht sein wollen d.h. Agenten kooperieren erst wenn es nötig ist
- Notwendig: Entwickler einigen sich auf Standard wie Agenten in Domäne agieren
- Abstimmung von Möglichkeiten und Tradeoffs für Protokolle, Strategien und die Sozialen Regeln der Agenten
- Eigenschaften von Standards:
  - Effizienz: Pareto Optimal
  - Stabil: kein Grund vom optimal abzuweichen
  - Einfach: gerade Berechnungs- und Kommunikationskosten
  - Verteilt: kein zentraler Koordinator
  - Symmetrisch: jeder Agent spielt äquivalente Rolle
- MAS: Gruppe von Nutzen-maximierende heterogenen Agenten welche coexistieren in der gleichen Umgebung, evtl Konkurrenz
- Auktionen als Verhandlungsmechanismus für die Teilung von Ressourcen
- in Abhängigkeit von Auktion jeder Agent wählt andere Strategie
- möglich: andere Szenarien für Verhandlungen ausser die Verteilung von Ressourcen

7.5 Verschiedene Domänen

- Aufgaben orientierte Domäne: Agenten wollen Aufgaben erledigen → Aufgabenverteilung
- Zustand orientierte Domäne: Ziele sind sind finale Zustände der Welt → gemeinsamen Plan
- Wert orientierte Domäne: Zustände haben ein Wert → gemeinsamen Plan und Ziel relaxierung

7.6 Aufgaben orientierte Domäne

- Tripe: $< T, Ag, c >$
- T ist eine endliche Menge von möglichen Aufgaben
- $Ag = \{1, \ldots, n\}$ ist eine Menge von beteiligten Agenten
- $c = p(T) \rightarrow R^+$ definiert die Kosten für die Ausführung jeder Teilmenge von Tasks (p ist Potenzmenge)
- Ein Encounter ist eine Menge von Aufgaben: $< T_1, \ldots, T_n >$ mit $T_i \subseteq T$ für jedes $i \in Ag$
- Bestandteile: Domäne, Verhandlungsprotokoll, Verhandlungsstrategie
- gegeben ein Encounter $< T_1, T_2 >$ ist ein Deal für die Zuweisung $T_1 \cup T_2$ für die Agenten 1 und 2
- die Kosten für den Agenten i eines Deals: $\beta =< D_1, D_2 >$ ist $c(D_i)$ (nachfolgend als $cost_i(\beta)$)
- Utility eines Deals β des Agenten i ist mit $c(T_i)$ die ursprüngliche Zuweisung deiner Aufgabe für Agent i: $utility_i(\beta) = c(T_i) - cost_i(\beta)$
- conflict deal Ω ist ein Deal $< T_1, T_2 >$ mit den ursprünglichen Zuweisung der Aufgaben: $utility_i(\Omega) = 0, \forall i \in Ag$
- ein Deal ist individuell rational wenn dieser schwach den conflict deal dominiert
- Agenten nutzen Produkt-maximierende Verhandlungsprotokolle
- The Monotonic Concession Protocol ???
  - Verhandlung in Runden
  - Runde 1: Agenten schlagen gleichzeitig ein Deal aus dem Verhandlungsset vor
  - Übereinkunft wenn ein Agent findet ein Deal welches mindestens so gut ist wie die anderen vorgeschlagenen
  - wenn keine Übereinkunft nächste Runde
  - in Runde $u + 1$ kein Agent darf schlechteren Deal machen als der aus der vorigen Runde
  - wenn kein Übereinkunft gefunden wird nach einigen Runden: Benndet mit Conflict-Deal
- The Zeuthen Strategy ???
  - Verhandlung mit dem Ziel der Messung der Bereitschaft ein Risiko einzugehen
  - höhere Bereitschaft wenn der unterschied zwischen den aktuellen Vorschlag und Konfliktdeal ist gering
  - Erster Vorschlag: der beste Deal eines Agenten
  - Jede Runde wird ein Agent riskieren
  - Entgegenkommen des Agenten nur soviel bis Risiko erreicht

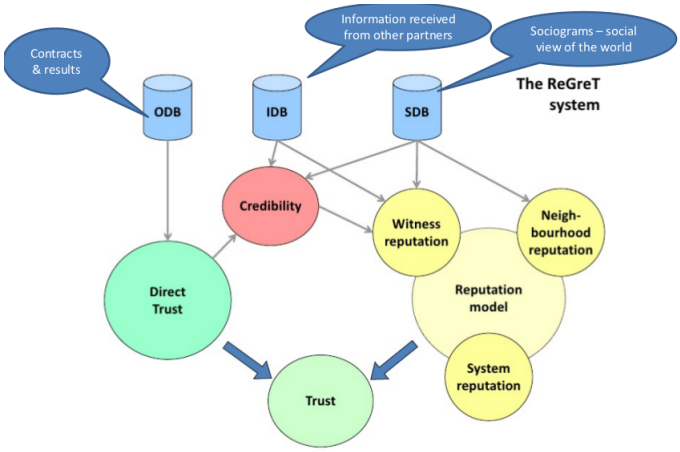
8 Trust and Reputation

8.1 Einführung

- Trust: Glaube dass ein anderer Agent eine Aktion tun wird ohne explizite Garantie um ein Ziel in einer riskanten Situation zu erreichen
- Reputation: Was die anderen Agenten sagen bezüglich seines Verhaltens
- Reputation erlaubt die Bildung von Trust; hat eine soziale Komponente welche nicht nur nützlich für ein Agenten ist sondern für alle

- jeder Agent wird durch andere Agenten beobachtet, keine zentrale Autorität
- Subjective vs. Globale Reputation
  - subjektiv: jeder Agent hat eigene Vorstellung von der Reputation anderer Agenten
  - global: Reputation als eine zentrale Resource, alle Agenten haben Zugriff auf die gleichen Reputationswerte
  - Vorteil global: Reputation ist Verfügbar auch für neue Agenten; einfachere Agenten da diese Werte nicht berechnet werden müssen
  - Nachteil global: funktioniert nur wenn Annahme das alle Agenten denken und verhalten ähnlich da interne Zustände des Agenten nicht berücksichtigt wird; nicht immer erwünscht das Agenten Information public macht oder diese an eine Zentrale Verwaltungsstelle sendet
  - Hoher Trust der Zentralen Organisation notwendig
- Vgl. Ebay: komplett zentral, Käufer hinterlassen Kommentare + Bewertung nach Käufe, jeder Teilnehmer hat einen Wert als Summe der Bewertungen
- Beispiel: Regret System

8.2 Regret System



- Modulares Trust und Reputations-System für komplexe e-commerce Umgebungen wo Soziale Relationen zwischen den Teilnehmern eine wichtige Rolle spielt
- Outcome: Initialer Vertrag + Realisierung des Partners (Attribute: Lieferungstag, Kosten, Qualität, Quanität)
- Impression: Subjekte Auswertung des Outcomes
- ??? Wie detailiert???

9 Blackboard

9.1 Allgemein

- indirekte Kommunikation d.h. keine direkte Kommunikation zwischen den Agenten sondern über ein Medium
- Kommunikation durch schreiben der Information auf Blackboard (zentraler Datenspeicher)
- Agenten können partielle Lösungen für das Problem dort veröffentlichen
- Zentrale Datenstruktur ist Flaschenhals ggf. muss irg welche Hierarchien verfügen
- subscripe-notify pattern: Agent bekündet Interesse über bestimme Ereignisse, wenn Ereignis eintritt wird Agent benachrichtigt (proaktiv)
- Annahmen: verschiedene Agenten greifen asynchron auf die Datenstruktur zu; ????
- Wissensquelle: Agent drückt ein Teil der Lösung aus; ???

- (??? IRG WAS MIT PRECONDITIONS ???)
- Verschiedene Hierarchien für Tests (??? PRECONDITIONS ???)
- Modularität: Man kann einzelne WQ des System entwickeln ohne zu wissen welche anderen WQ es gibt (also indirekte Kommunikation)
- Interessant: Wie verändert sich die Systemleistung durch Austausch einzelner WQ?
- Kommunikation:
  1. Sammlung von Daten über Events für zukünftige Anwendung
  2. Detektion von Event welche vorige Annahme widersprechen (???)
- Lokale Kontexte: Jeder hat eigene lokalen Abbild der Datenbank der von Interesse ist, Übertragung neuer Events an entsprechende Interessierte
- (??? Folie über sprungen zu Integrität ???)
- jeder Agent hat eigene Datenbank, Verteilung der Information über Blackboard; lokale Datenbanken bilden zusammen globale Datenbanken

9.2 Alternative Architekturen

- Beispiel: Speech Understanding,
- Ansatz 1: Pipe-and-Filter Architektur; Problem: keine klare Trennung, ggf Wissen aus unterschiedlichen Quellen notwendig d.h. Architektur wird unnötige komplex
- Ansatz 2: Object-Orientierte-Architektur; Problem: nicht flexibel da jede WQ wissen muss was eine andere WQ produzieren kann; Lösung: Client-Broker Ansatz aber recht komplex
- Ansatz 3: Sichten-System; Problem:
- Besser: Alle Prozesse kommunizieren über Blackboard, Ziel: Wissensverteilung der Partiellen Lösungen

10 Organisationskontrolle

10.1 Allgemein

- Organisationkontrolle (Wer kann mit den Interagieren) kann Kommunikation und Kooperation vereinfachen (kürzere Entscheidungshierachie, weniger Kommunikation)
- Strukturierung als zentrale Aufgabe reflektiert die Aufgabe und sorgt für eine effizientere Lösung der Aufgabe
- Organisationktrolle funktioniert weil Problem zerlegbar und häufig wiederholt wird
- Framework für die Skalierung der Aufgaben
- Ziel: Wissen lokal Halten damit nicht zusätzlicher Aufwand durch Verwaltung fremden Wissens entsteht
- Oragnisations Tradeoff:

11 Agenten und Mobilität

- Verteilte Orte: Wie können diese miteinander Reden?
- Verteilte Systeme: veschiedene Möglichkeiten um Methoden aufzurufen von anderen Koten (Remote Procedure Call)
- Lösungen: Stub, Skeleton. Marshalling (Serialisierung von Objekten),IDL
- Stub hat Reference zu Skeleton-Methodes
- Probleme:
  - Client kann blockieren

  - Verbindung kann nicht sicher sein (fällt ggf aus), Implementierung von Timeouts usw (unreliable links)
  - Übertragung von großer Daten kann Problem sein, besser Server sollte berechnen mit den Daten ohne zu verschicken ABER kennt Algorithmus nicht (Lösung: Mobile Agents)

- Agenten-basierter Ansatz: Alternative zu RPC ist RP
- Client macht Code für Prozeduren und Klassen bekannt
- Prozedur + Aktuellen Zustand präsentiert einen Mobiler Agent
- Weak migration: ? (standard)
- Strong migration: ?
- Remote Programming: Koventionen zwischen client und server für befehle und datentypen bilden eine Sprache