Mobile Agenten als Architekturkonzept Internet-basierter Anwendungen

[Mobile Agents as an Architectural Concept for Internet-based Distributed Applications]

Friedemann Mattern

Stefan Fünfrocken

Technische Universität Darmstadt

www.informatik.tu-darmstadt.de/VS

Übersicht

- Mobile Agenten als Architekturkonzept
  - was sind mobile Agenten?
  - mobiler Code
  - Einsatzgebiete mobiler Agenten
  - Agentensysteme

MA'98

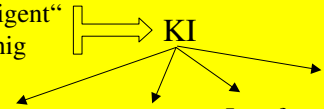
- Unser WWW-basiertes Agentensystem
  - Architektur des Systems
  - transparente Migration mit Java
  - Anwendungsbeispiele
  - Erfahrungen
- Problembereiche mobiler Agenten

Software-Agent

- Handlungsbevollmächtigter
  - es gibt *Auftraggeber* (Nutzer, Besitzer…)
  - Agent führt an ihn *delegierte* Aufgaben aus
  - *handelt* im Auftrag
- Charakterisierung?

  - autonom
  - zielgerichtet
  - sozial
  - kooperativ

  - „intelligent“
  - lernfähig

  Hewitt: Actor-Modell (1977)

  kollaboratives Problemlösen

  Interface-Agent
- Metapher verschiedener Disziplinen
  - z.B. „user agent“
  - Modewort: compression agent = Komprimierungsprogramm?
    power agent = Lichtschalter?
- Mobiler Agenten = ?

3

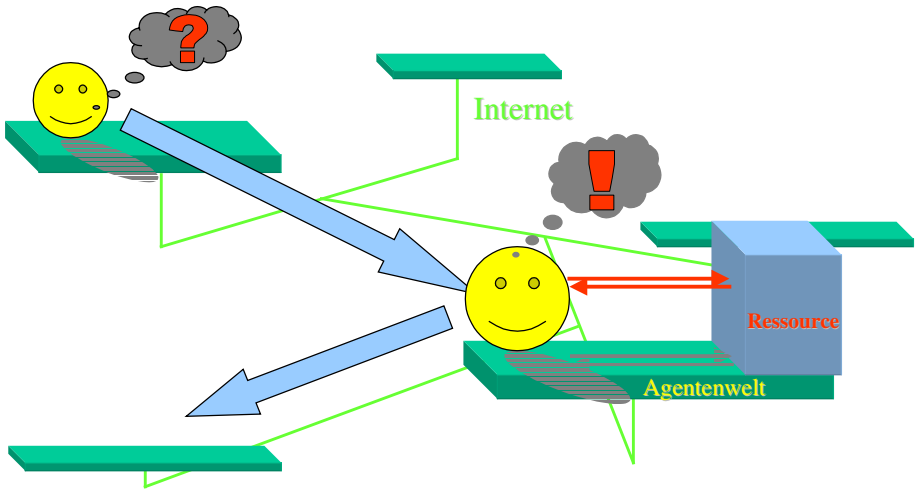
Software-Agent

- Handlungsbevollmächtigter
  - es gibt *Auftraggeber* (Nutzer, Besitzer…)
  - Agent führt an ihn *delegierte* Aufgaben aus
  - *handelt* im Auftrag
- Charakterisierung?

  - autonom
  - zielgerichtet
  - sozial
  - kooperativ
- Metapher verschiedener Disziplinen
  - z.B. „user agent“
  - Modewort: compression agent = Komprimierungsprogramm?
    power agent = Lichtschalter?
- Mobiler Agenten = ?

4

Mobile Agenten



- Umherwandern in einem Netz (Internet, Intranet)
  - dabei Verrichtung eines Dienstes (für den Auftraggeber)

5

Mobile Agenten: Migration

- Migration unter Anwendungskontrolle
  - „proaktiv“: Agent selbst entscheidet und initiiert Migration
- Schwache Migration
  - nur Programmtext
  - Agent wird am Zielort „neu“ ausgeführt
- Starke Migration
  - Agent behält Ausführungszustand (lokale Variablen, Laufzeitkeller, Aufrufkeller, “Programmzeiger“)
  - Fortführung mit der nächsten Anweisung

Unser System realisiert die starke Migration

```
x := ...;
if (x > 17) then go supercomputer;
y := f(x) + 29;
```

6

Mobiler Code: Historie

- Prozeßmigration in Betriebssystemen
  - auf lokaler Ebene (Rechner, LAN)
  - *ortstransparent*; Migration systeminitiiert
  - Zeck typischerweise: Lastausgleich
  - Probleme: Effizienz; oft diffuse Umgebungsabhängigkeiten
  - „moderner“: Objektmigration (z.B. Emerald, Cool)
- …
- Mobile Agenten

ortsbewußt und eigen-initiierte Migration

  - Begriff und Patent (1997): General Magic (Telescript)
    - „System and Method for Distributed Computation based upon the Movement, Execution, and Interaction of Processes in a Network“
  - ansatzweise: active mail, Java-Applets (schwache Migration)

7

Mobile Agenten: Voraussetzungen

- Einheitliche Sprache auch bei heterogenen Systemen
  - typw. Skriptsprachen bzw. interpretierte Sprachen
  - z.B. Tcl, Perl, Java-Bytecode
  - Sprachinterpreter bzw. VM ist einfach zu installieren
  - Konsequenz: Leistungseinbußen
- Klare Abgrenzung nach außen („Umgebung“)
  - Objekt-Paradigma (Agent = Objekt + Prozeß)
  - Schnittstellen, keine einfachen Zeiger,…
- Einheitlicher Kontext?
  - lokale Ressourcen, Bibliotheken…
- Lokale und globale Infrastruktur erforderlich

8

Gute Zeiten für mobile Agenten

- Internet wächst, verästelt sich bis in die Haushalte
  - Internet appliances, pervasive Systeme…
  - Internet wird dynamischer, mobiler
- Neue Anwendungen
  - Massenphänomen --> plug & play
  - adaptive Software
  - Fernwartung und service customization
- Kürzere Entwicklungszyklen
  - schnelle, dynamische Reaktion auf Marktentwicklung
  - entfernte Softwarekonfiguration und Installation
- Neue Marktstrukturen
  - dynamische Föderationen von Diensten und Dienstleistern
- Mehrere Projekte / Systeme in Forschung und Industrie
  - vielfach Java-basiert (Applets, RMI,…)

9

Gute Zeiten für mobile Agenten

- Internet wächst, verästelt sich bis in die Haushalte
  - Internet appliances, pervasive Systeme…
  - Internet wird dynamischer, mobiler
- Neue Anwendungen
  - Massenphänomen --> plug & play
  - adaptive Software
  - Fernwartung und service customization
- Kürzere Entwicklungszyklen
  - schnelle, dynamische Reaktion auf Marktentwicklung
  - entfernte Softwarekonfiguration und Installation
- Neue Marktstrukturen
  - dynamische Föderationen von Diensten und Dienstleistern
- Mehrere Projekte / Systeme in Forschung und Industrie
  - vielfach Java-basiert (Applets, RMI,…)

Mobile Agenten ermöglichen flexible und anpassungsfähige Strukturen; sind daher für große, offene, dynamische Systeme geeignet (*function shipping, call by visit, code on demand…*)

10

Vorteile gegenüber RPC / Client-Server-Modell

als „klassisches“ Architekturkonzept verteilter Systeme

- Agent arbeitet asynchron zum Client
- Ggf. geringerer Bandbreitenbedarf
  - große Datenmengen effizient „lokal“ durchsuchen
  - Programm selbst geht zu benötigten Betriebsmitteln
- Schnellere Reaktion auf entfernte Ereignisse
  - Handlungsbevollmächtigter ist vor Ort (z.B. Börse?)
- Potential für Fehlertoleranz
  - Agent kann z.B. autonom Backup-Server aufsuchen
  - allerdings auch mehr Fehlerarten
- Unterstützung mobiler / sporadischer Nutzer
  - Agent losschicken und dann abschalten
- Server ggf. dynamischer durch Service-Agenten

11

Anwendungsgebiete

- Electronic Commerce
  - Verkäufer, Käufer, Broker…
  - Agent geht für mich im Internet einkaufen…

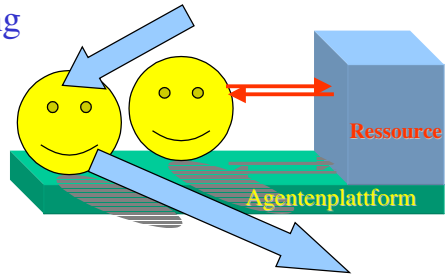
- gibt mein Geld aus?

- Informationssuche
  - Agent führt Profil mit und liefert Ergebnis zurück
- Fernwartung
  - Monitoring (z.B. Management von Netzen)
  - Installation von Softwarekomponenten
- Workflow-Management, Groupware-Anwendungen
  - Bsp: Agent „hilft“ beim Ausfüllen von (aktiven) Formularen
- Personalisierung von Diensten
- Entertainment

„Mobile agents are a solution in search of a problem“ (J. Ousterhout)

Lokale Infrastruktur für Agenten

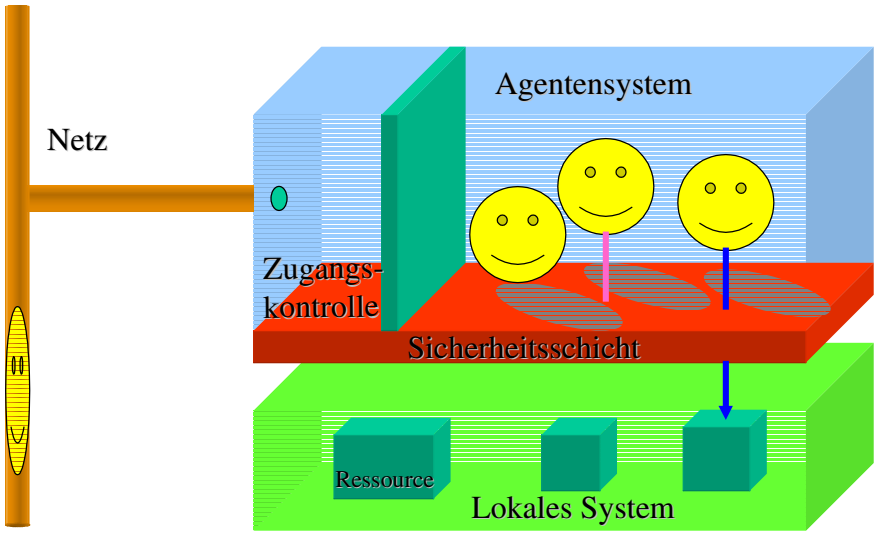
- Agentenplattform als Wirtssystem / Ablaufumgebung
- Migrationsunterstützung
- Ausnahmebehandlung
- Ereignisweiterleitung
- Schutz / Sicherheit
- Fehlertoleranz
- Informationsdienste
- Kommunikation
  - mit anderern lokal vorhanden oder entfernten Agenten
  - mit dem Auftraggeber
  - welche der diversen Kommunikationsparadigmen?
- Ressourcenverwaltung (Speicher, Rechenzeit…)
  - Zugriffsrechte
  - Schutz vor Monopolisierung



Erfordert Schnittstellenver-einbarung bzw. „-sprache“

13

Sicherheit: Zugriffsschutz



14

Globale Infrastruktur

- Finden von Agenten
  - „Namensdienst“ unter Berücksichtigung der Mobilität
  - Vermitteln von Diensten
- Nachsenden von Botschaften
- Terminieren von Agentengruppen
  - orphan detection, distributed garbage collection
- Anbindung an etablierte Systeminfrastrukturen
  - CORBA, Jini, WWW-Infrastruktur, Internet-Dienste,…
  - application frameworks
- ...

15

Das WASP-Projekt

Web Agent based Service Providing

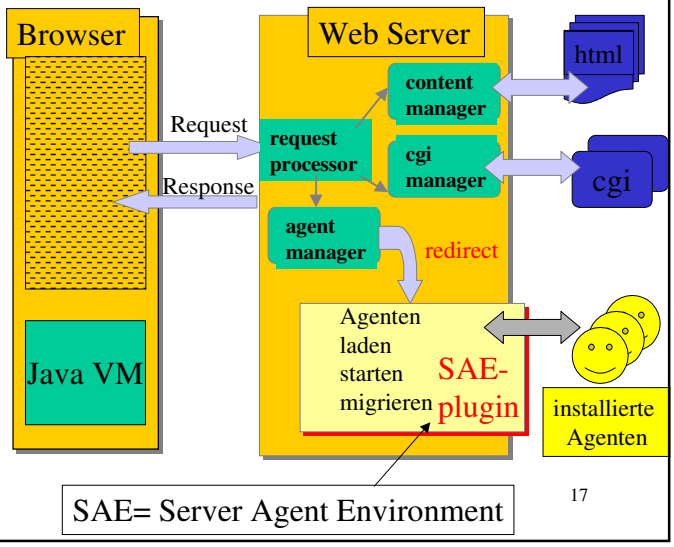
- Mobile Agenten zur Realisierung von Diensten im Internet
  - Fragstellung: Inwieweit eignet sich das Agenten-Paradima?
- Nutzung des WWW
  - WWW-*Server* als Agentenplattform
    - relevante Information ist im Internet i.a.über WWW-Server zugänglich
  - WWW-*Browser* als Benutzungsinterface
    - Starten von Agenten auf Servern; GUI für diese so gestarteten Agenten
    - Browser als Agentenplattform weniger gut geeignet (Applets werden z.B. beim Iconifizieren gestoppt)
  - Mit dem WWW läßt sich die Plattform *leicht verbreiten*
  - HTTP zum *Agententransfer* (MIME-kodiert)

16

Architektur des WASP-Servers

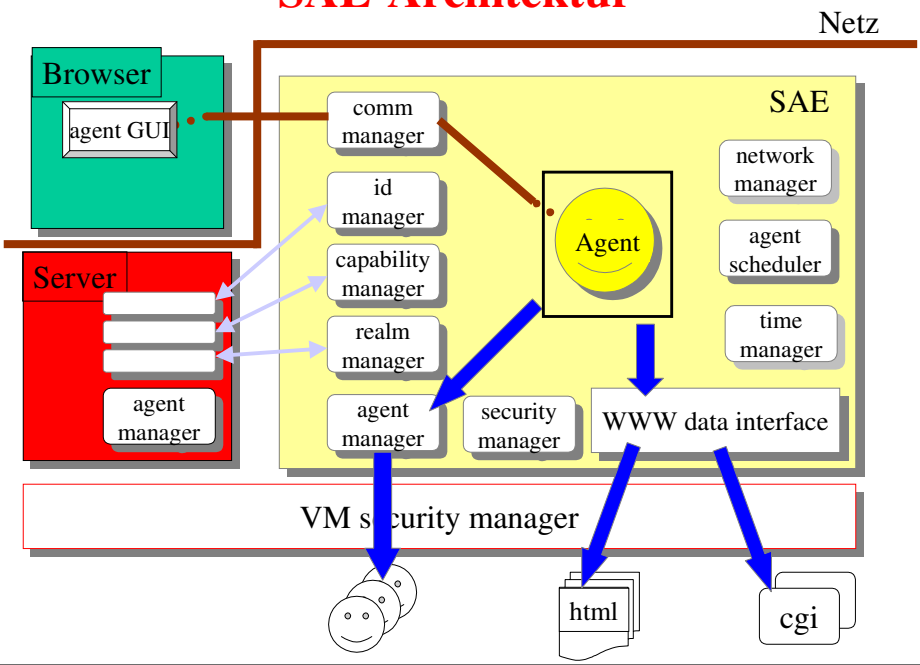
- Eigenständiger Server; SAE aber auch als Servlet für Standardserver realisiert

- Agent wird über eine URL adressiert
- Gründen von Agenten von einem beliebiegn Browser
- Browser ist auch Benutzungsinter-face von Agenten
- Modular erweiter-bare Manager
  - content manager
  - cgi manager
  - agent manager
  - script manager



17

SAE-Architektur



WASP

- Projekt seit 1996
  - TU Darmstadt, FB Informatik
- Java
  - Implementierungssprache der Plattform
  - Nutzung von Objektserialisierung, Klassenlader, RMI…
  - Programmiersprache für Agenten
    - von der Klasse ‚agent' abgeleitet
  - Applets im Browser als Agenten-GUI
- Erste (nicht ganz triviale) prototypische Anwendungen
- Weiterentwicklung

19

Funktionalität der Agentenplattform

- Migrationsunterstützung
- Zugriffsschutz
- Verwaltung von Betriebsmitteln
  - Schutz vor Monopolisierung
- Kommunikation
  - Java-Streams (automatisches Reconnect nach Migration)
  - entfernter Objekt-Zugriff (CORBA oder Java-RMI)
  - SAE-lokale Kommunikation über normale Objektreferenzen
- Sicherheit
  - z.Z. einfache Methoden und kurze Schlüssel
  - Verschlüsselung von Agenten beim Transport
  - Signieren von Agenten

20

Adressierung

- Agenten werden über URLs adressiert
  - URL der Erzeugerplattform + Zusatz
  - Proxy mit neuer Adresse, falls der Agent wegmigriert ist
  - Agent muß dort seine neue Adresse jeweils melden
- Ressourcen auf WWW-Servern (Dateien, andere Agenten etc.) können gleichermaßen von *menschlichen Nutzern* (über WWW-Browser) wie von *Agenten* angesprochen werden
  - Aufgabe an Agenten delegieren, wenn man müde ist...

21

Transparente („starke“) Migration

- Bei der Migration werden neben dem Programmcode auch alle Daten eines Agenten transportiert
  - Variablen
  - Laufzeitkeller
  - „Programmzähler“

Migration eines kleineren Agenten: ca. 20 - 50 ms

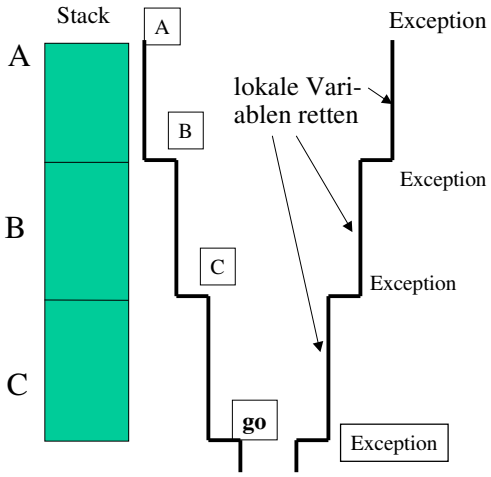
- Ohne Änderung der Virtuellen Maschine von Java
  - aber mit einem Preprocessor auf Agenten-Quellcode
- Vor der Migration (‚go‘): Zustand einsammelm
  - Save-Objekte
  - Nutzung des Exception-Mechanismus
  - Objektserialisierung von Java
- Nach der Migration: an die Stelle nach den ‚go‘ laufen durch Überspringen bereits ausgeführter Teile

22

Durchlaufen des Laufzeitkellers

- Werte des Laufzeit-kellers retten
  - nur bei tatsächlicher Migration
- 'go' erzeugt spezielle Exception:

```
try {
    go(location);
}
catch(MigrateException mig) {
<Werte lok. Variablen retten>
throw mig;
}
catch ....
```



23

Beispielanwendungen

- „Web-Zeitung“
  - Benutzer wählt über ein GUI Rubriken aus
  - Suchagenten begeben sich ins Internet zu Informationsanbietern
  - falls Information gefunden wird, wird ein Data-carrier-Agent zurückgeschickt
  - Information ist verschlüsselt; wird erst bei Eingabe eines Schlüssels freigeschaltet
- Login-Tracer
  - Rückverfolgung von remote logins
  - Agent begibt sich auf die Maschine, von der ein entferntes login kam (rekursiv --> transitive Hüller eines Clusters)
- Anwendungsmanagement
  - Konfiguration von WWW-Servern durch Agenten

24

Weitere Arbeiten

- JavaCard als „trusted computing base“
  - Ressourcen sind relativ beschränkt
- Agentenkonstruktionstool
  - Zusammenbau von Agenten aus Komponenten
- Electronic Commerce-Framework
  - z.B. Bezahlfunktionalität
- Agentenplattform als reines Server-Plug-in
- ...

25

Mobile Agenten: Problembereiche

- Verbreitung einer (standardisierten?) Agenten-infrastruktur im Internet
  - mögliche Lösung: Nutzung des WWW durch „plug ins“ für WWW-Browser (applets) und WWW-Server (servlets)
- Zugriffsrechte, Autorisierung, Authentizität
  - unabdingbar für Electronic Commerce-Anwendungen
  - Nutzung der entstehenden Internet-Infrastruktur (trust center etc.)?
- Schutz des Wirtssystems vor bösartigen Agenten
  - auch Viren sind Agenten
- Schutz von Agenten (und Agentenanwendungen)
  - getürkte Agentenplattform (Abhilfe: „trusted environment“)
  - Abfangen von Agenten
  - Ausspähen von Geheimnissen im Agentencode

26

Konzeptionelle Probleme

- Wohldefinierte Semantik von Mobilität
  - was genau macht den *Kontext einer Ausführung* aus?
  - welche Teile des Kontextes werden mitmigriert?
    - identische oder gleichartige Ressource verwenden?
  - was geschieht im *Fehlerfall*, z.B.:
    - Agent verschollen (Exactly-once-Transfer?),
    - Migrationsziel nicht erreichbar,
    - Ressource am Ziel nicht verfügbar,
    - Quelle existiert bei „copy by need“ nicht mehr?
- Kann ein Agent ein Geheimnis bewahren?
  - Strategie ausführen (lassen!), dem host aber nicht verraten?
  - elektronisches Geld, das nicht geraubt werden kann?
  - muß sich ein Agent seiner Ausführungsumgebung nicht Anweisung für Anweisung ganz offenbaren?

27

Konzeptionelle Probleme

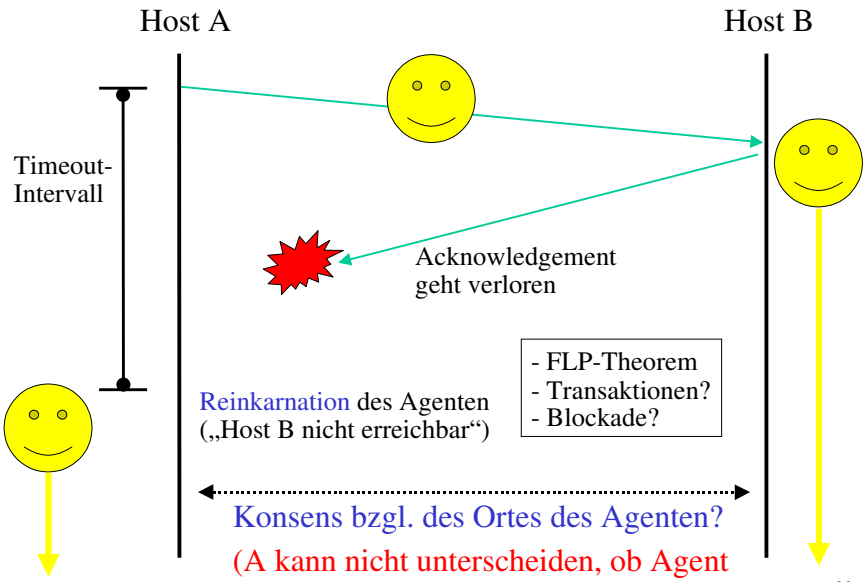
- Wohldefinierte Semantik von Mobilität
  - was genau macht den *Kontext einer Ausführung* aus?
  - welche Teile des Kontextes werden mitmigriert?
    - identische oder gleichartige Ressource verwenden?
  - was geschieht im *Fehlerfall*, z.B.:
    - Agent verschollen (Exactly-once-Transfer?),
    - Migrationsziel nicht erreichbar,
    - Ressource am Ziel nicht verfügbar,
    - Quelle existiert bei „copy by need“ nicht mehr?

Diese Probleme (sowie Sicherheit, Einbindung in existierende Umgebungen, weite Verbreitung etc.) müssen für eine praktische Verwendung (z.B. Electronic Commerce) gelöst sein!

- Kann ein Agent ein Geheimnis bewahren?
  - Strategie ausführen (lassen!), dem host aber nicht verraten?
  - elektronisches Geld, das nicht geraubt werden kann?
  - muß sich ein Agent seiner Ausführungsumgebung nicht Anweisung für Anweisung ganz offenbaren?

28

Problem: Exactly-once-Transfer?



29

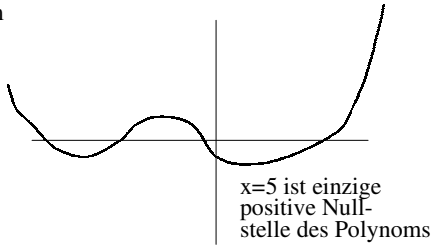
Problem: geheime Strategie?

```
x = ask_for_price(…);
if (x > 5) then go to A;
           else go to B;
```

„richtige“ Alternative; mein Preisgebot sollte 4.99 betragen

Äquivalenztransformation, um Strategie zu verbergen

```
x = ask_for_price(…);
if (x**4+6*x**3-19*x**2-144*x)
> 180 then go to A;
```



Einwegprinzip:

- äquivalenter Ausdruck ist einfach zu konstruieren
- aber „schwer“ zu lösen (und zu verstehen!)

Aber: wann nützt so etwas denn?

30

Konklusionen

- WASP: Forschungsprototyp
  - WWW-Integration durch plug-ins in Server (und Browser?)
- Agenten als ein (weiteres) Architekturkonzept für verteilte Anwendungen
  - ermöglicht flexible, dynamische Systemstrukturen
  - function shipping, call by visit, code on demand,...
  - „natürliche“ Modellierung verteilten Systemverhaltens
- Praktischer Einsatz mobiler Agenten ist aber noch durch einige ernste Probleme behindert
  - Sicherheit
  - ...
- Vermutlich Integration der Idee mobiler Agenten in zukünftige Middleware (CORBA++, Jini,…?), in Kombination mit den etablierten Prinzipien

31

ENDE

32