Mobile Agenten

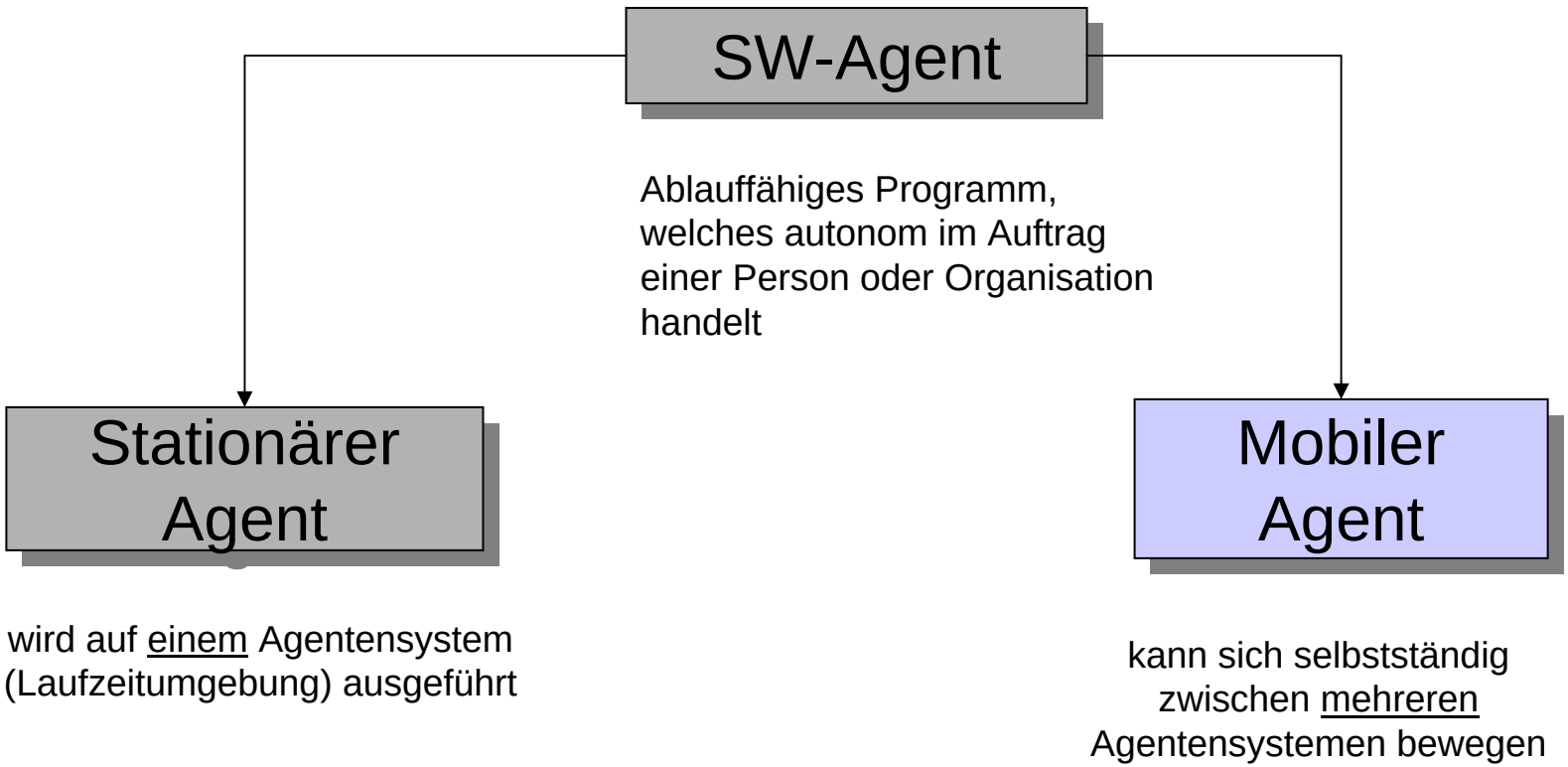
Informatik Seminar SS2004

Matthias Rohr (mi4295)

Gliederung

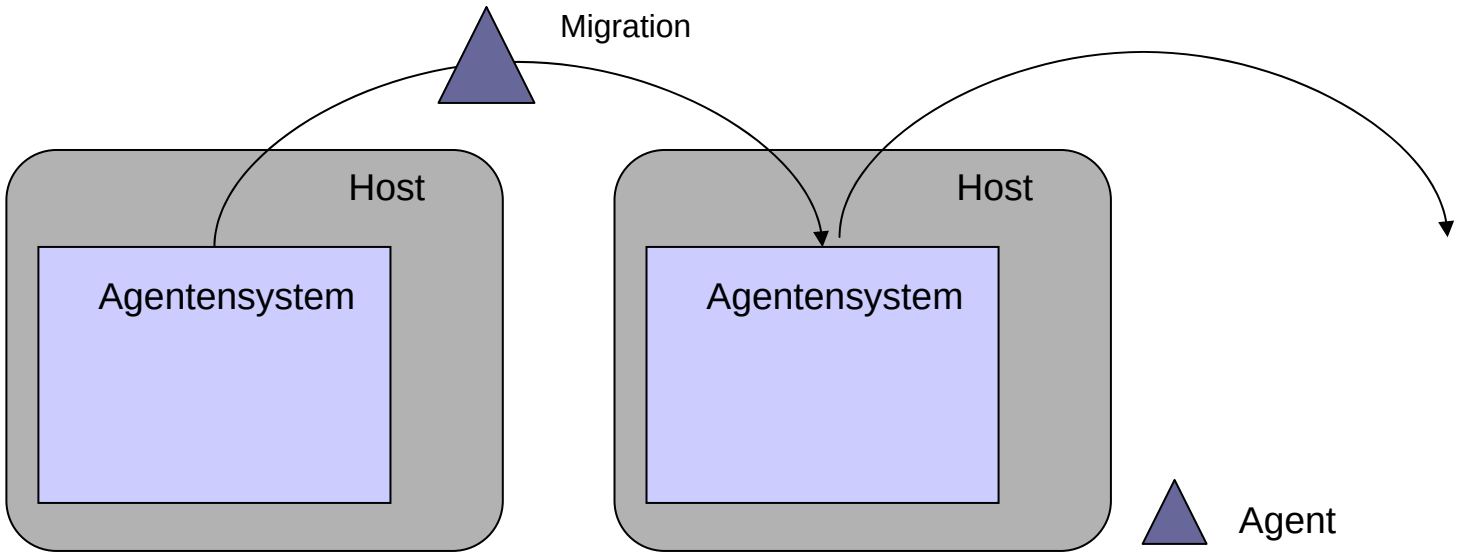
1. Was sind mobile Agenten ?
2. Aspekte eines Agentensystems
   - Migration
   - Kommunikation
   - Sicherheit
3. Standardisierung
4. Fazit

Begriffsbestimmung



Quelle: MASIF-Spezifikation

Eigenschaft Mobil

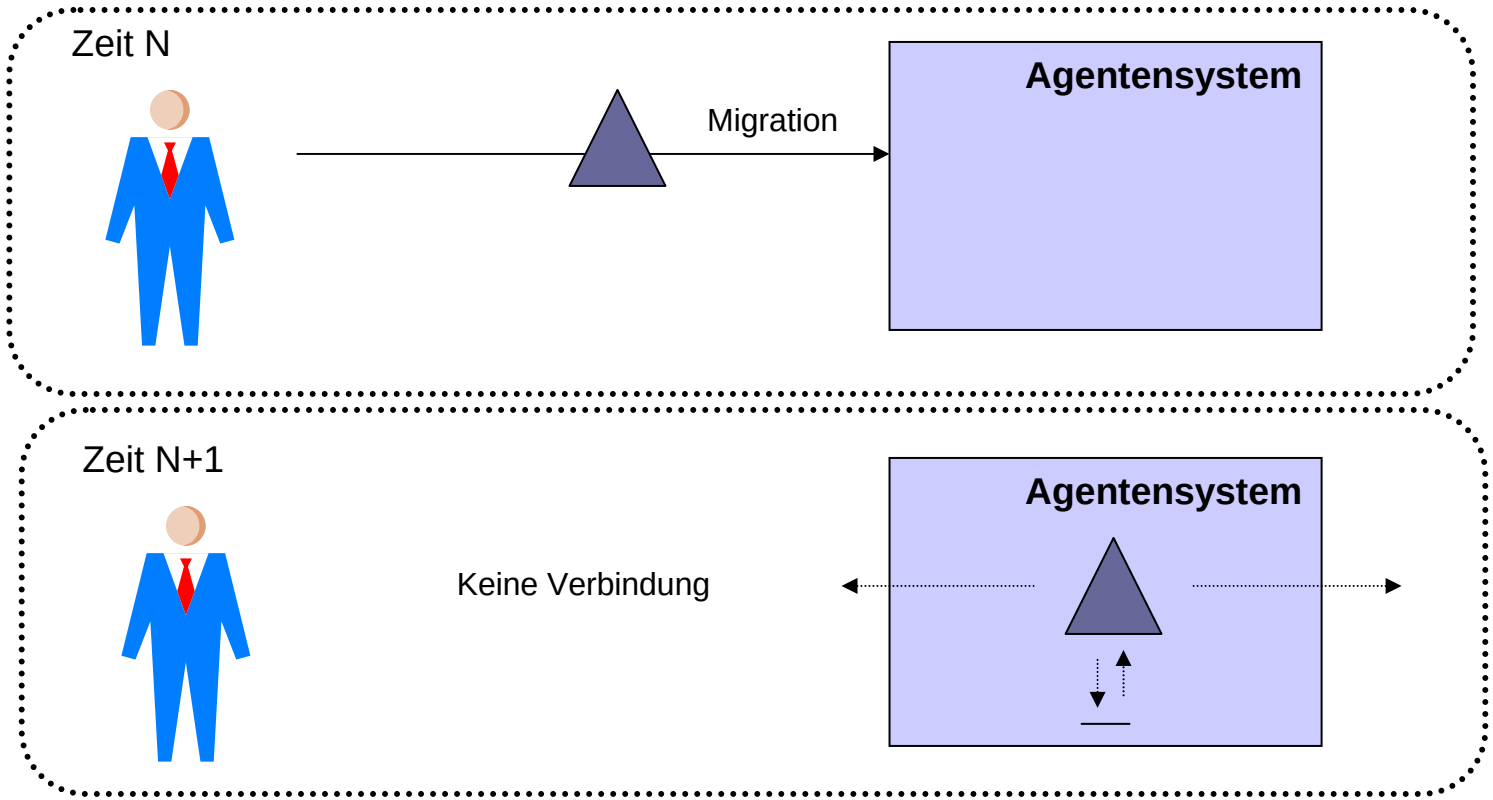


Agent kann zwischen Agentensystemen wechseln (migrieren)

Voraussetzungen:

- Agenteninstanz kann in Datenfluss umgewandelt werden (Serialisierbarkeit)
- Die Zielsystem kann den Code des Agenten empfangen und interpretieren

Eigenschaft Autom



Der Agent entscheidet selbstständig anhand bestimmter Kriterien über seine nächste Aktion (Problem: Kontrollmöglichkeiten).

Eigenschaft Soziale Kompetenz

Agent kann mit anderen

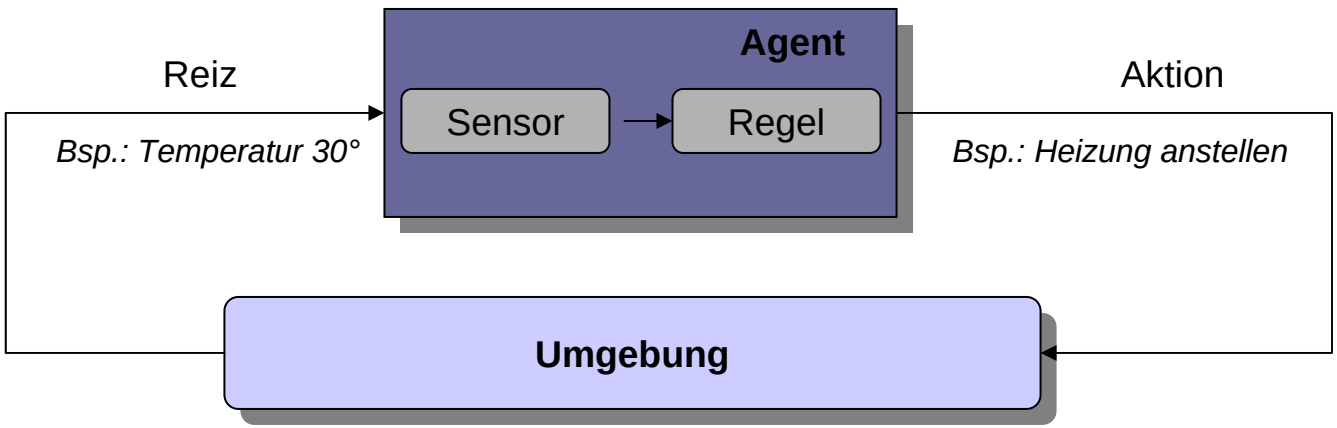
- Agenten
- (Agenten-)Systemen
- Menschen

Kommunizieren / Kooperieren

Vorraussetzungen:

- Gemeinsame Protokolle
- Gemeinsame Sprache / Schnittstellen
- Gemeinsame Ontologien

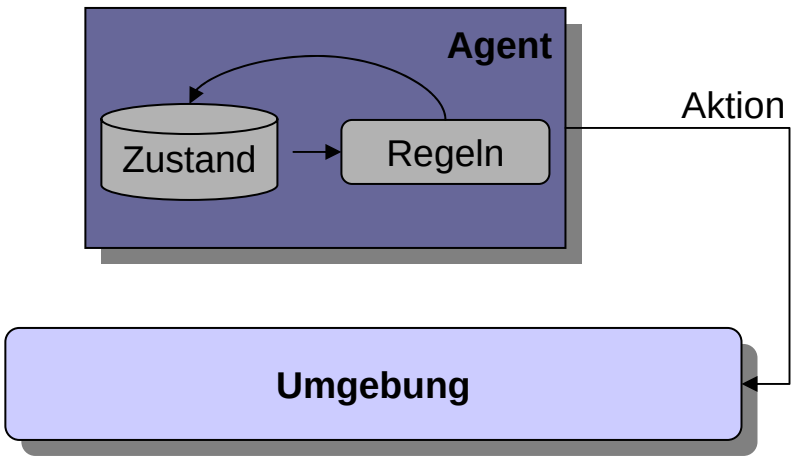
Eigenschaft Reaktiv



Der Agent reagiert auf Änderungen seiner Umgebung (engl. Environment)

Vorraussetzung: Existenz von Sensoren und Regeln

Eigenschaft Proaktiv / Zielgerichtet



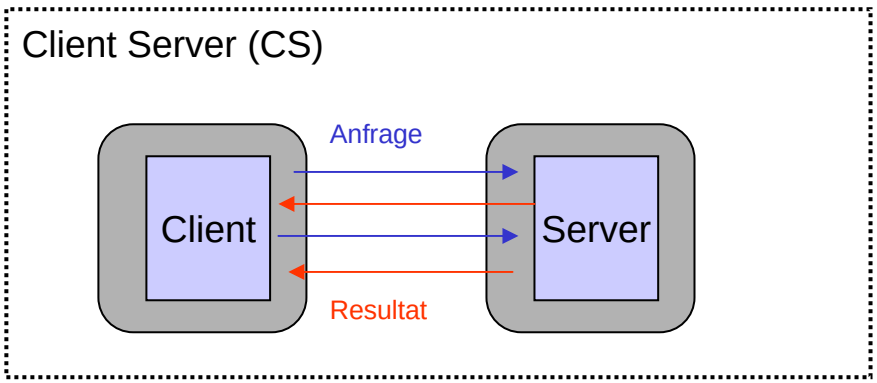
Agenten reagieren nicht nur auf Reize der Umgebung, sondern besitzen internen Zustand und sind zu zielgerichtetem Planung und Handeln fähig.

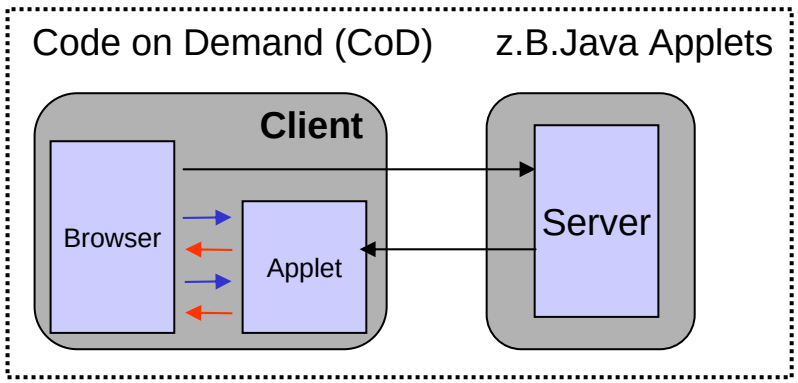
=> Sie ergreifen die **Initiative**

*„The difference between an automation and an agent is a somewhat like the difference between a dog and a butler. If you send your dog to buy a copy of the New York Times every morning, it will come back with its mouth empty if the news stand happens to have run out one day. In contrast, the butler will probably take the **initiative** to buy you a copy of the Washington Post, since he knows, that sometimes you read it instead."*

Le Du

Mobile Agenten als Softwareparadigma

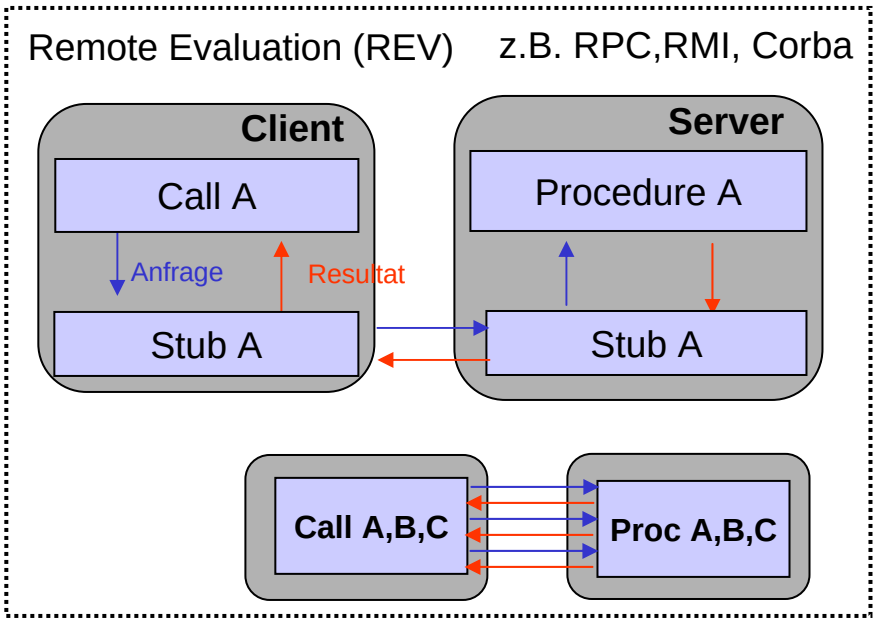


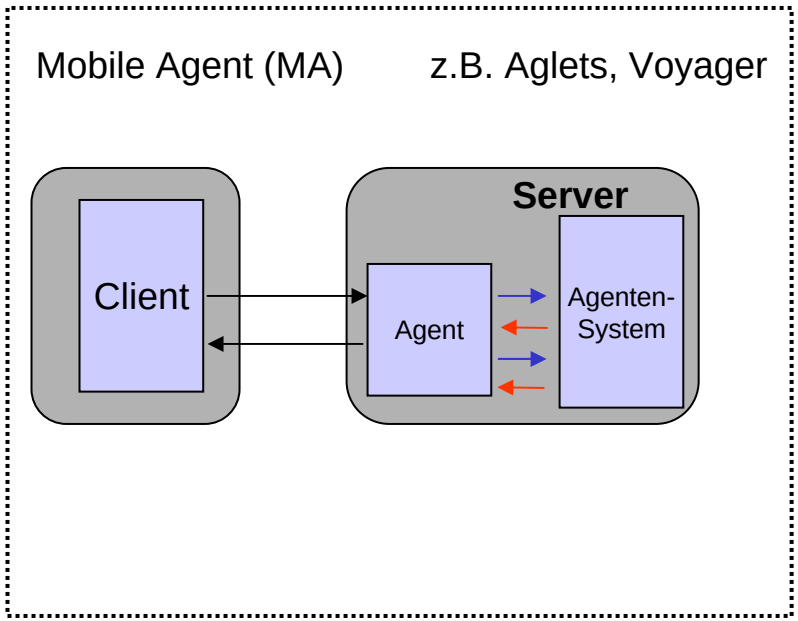


- Funktionsweise wird vom Server festgelegt
- Ressourcenverbrauch beim Server

- Funktionsweise wird vom Applet festgelegt
- Ressourcenverbrauch beim Client

Mobile Agenten als Softwareparadigma





- Funktionalität wird vom Server festgelegt
- Funktionsweise werden vom Client festgelegt
- Ressourcenverbrauch hauptsächlich Server

- Funktionalität wird vom Client festgelegt
- Ressourcenverbrauch beim Server

Mobile Agenten vs. RPCs

1. **Bandbreite**

   Relevant erst ab bestimmtem Übertragungsvolumen

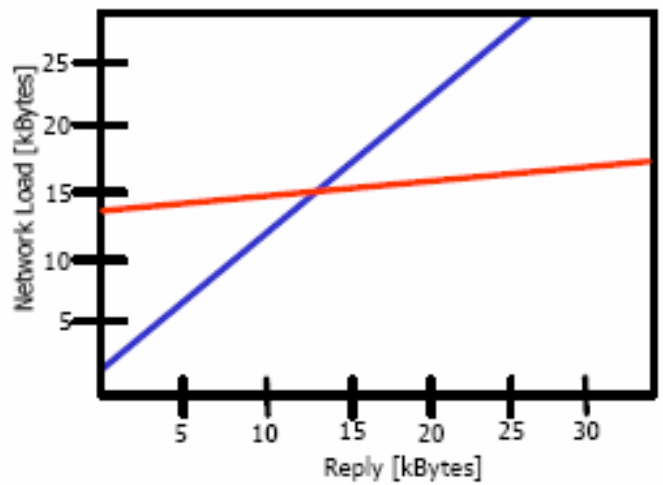
2. **Latenzzeit**

   Relevant erst ab bestimmter Entfernung der Kommunikationspartner (Bsp.: Australien -> Deutschland)

3. **Ausfallsicherheit**

Aber : Mobile Agenten mehr als RPC-Ersatz:

- Keine vordefinierte Funktionalität
- Ermöglicht Offline Betrieb
- Unterstützung Heterogener Umgebungen

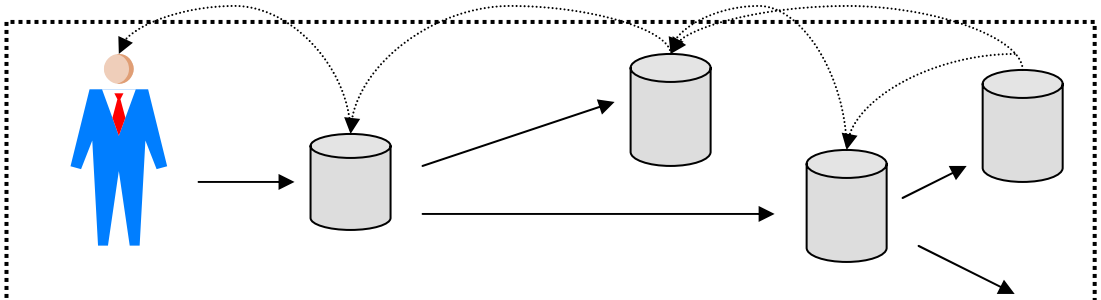


Quelle: Straßer/Schwehm: A Performance Model for Mobile Agent Systems, PDPTA‘97.

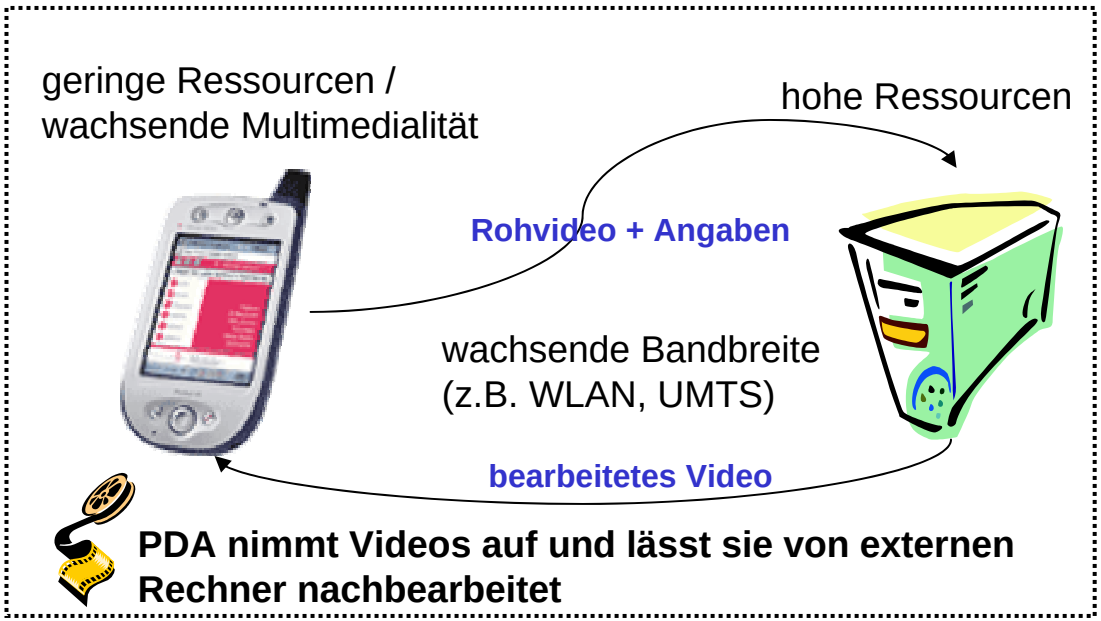
Wozu Mobile Agenten ?

- Informationssuche
- Überwachung und Steuerung
- Paralleles Rechnen
- (Transaktionsagent)
- Mobile Computing

Aber: Keine „Killerapplikation “



Intelligente Suche in verteilten Datenbanken durch Klonen von Agenten



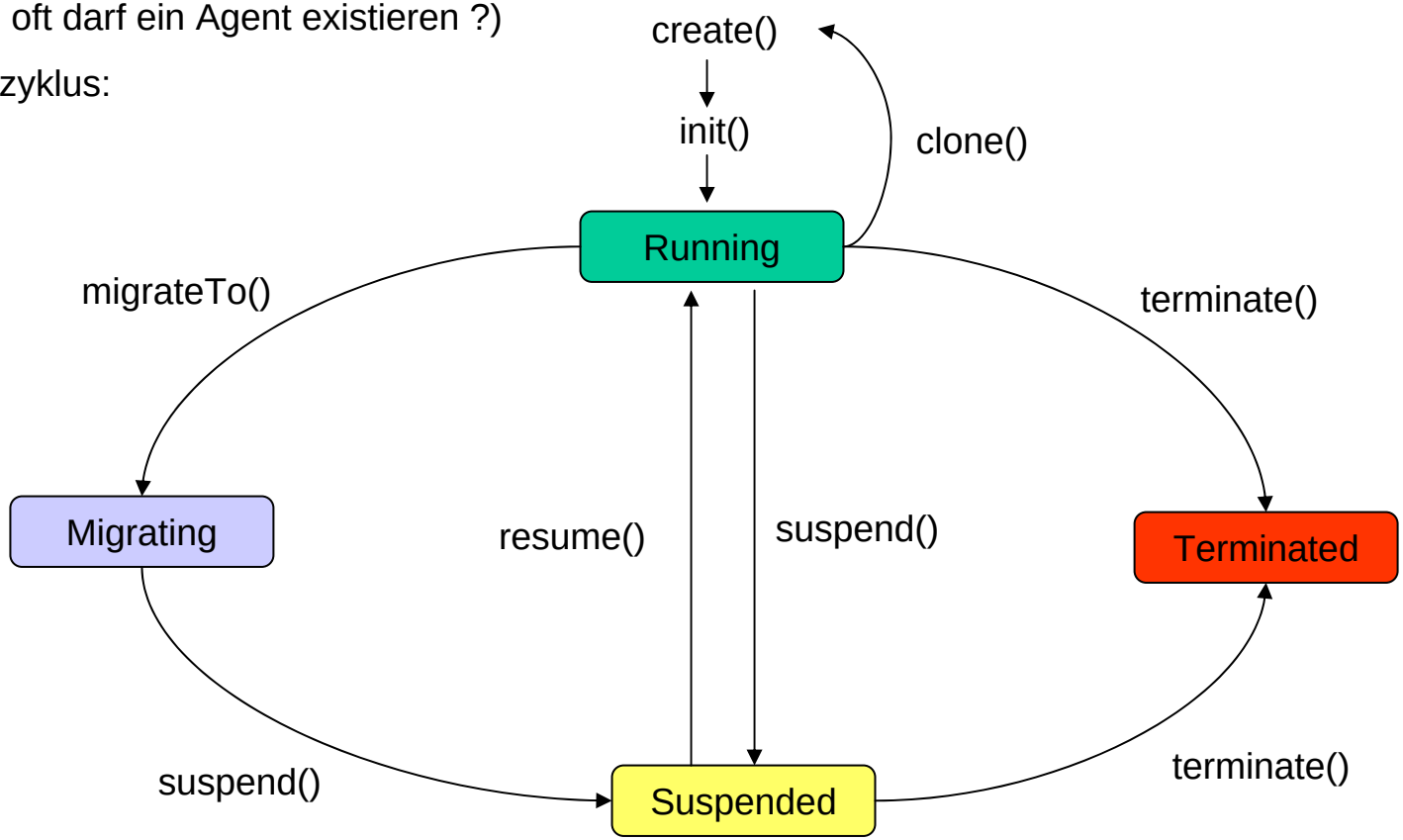
PDA nimmt Videos auf und lässt sie von externen Rechner nachbearbeitet

Gliederung

1. Was sind mobile Agenten ?
2. Aspekte eines Agentensystems
   - Migration
   - Kommunikation
   - Sicherheit
3. Standardisierung
4. Fazit

Agent aus Sicht eines Agentensystems

- Besitzt eindeutige Identifikation / Namen
- Exklusivität ? (Wie oft darf ein Agent existieren ?)
- Durchläuft Lebenszyklus:

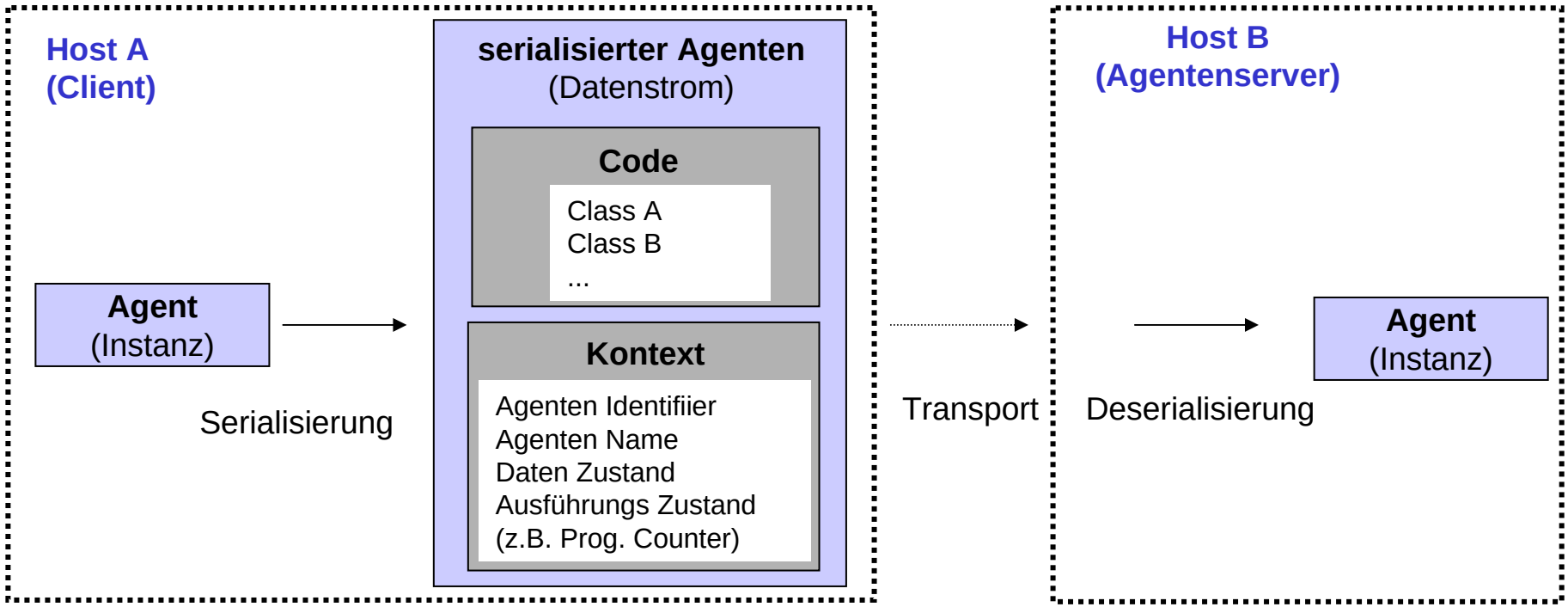


Eigenschaften eines Agentensystems

Aufgabe: Laufzeitumgebung (Wirtssystem) des Agenten und dessen Abbildung entlang eines Agentenlebenszyklussees

- **Art:** Multi Agenten System (MAS), Single Agenten System
- **Sprache**: für Agent, für AgentenServer (z.B. Java, TCL)
- **Management** u.A. Agentenlebenszyklus, Agentenadressierung
- **Migrationskonzept** Transport (z.B. TCP, HTTP, Mail), Serialisierung (z.B. Java)
- **Kommunikation:** Agent<->Agent, Agent->System, System <->System
- **Sicherheitskonzept:** für Agent, für Agentenserver
- **Interoperabilität** zu anderen Agentensystemen

Migration



Unterschiedliche Migrationsarten (was migriert ?)

- **schwache Migration:** ohne Ausführungszustand
- **starke Migration:** mit Ausführungszustand

Schwache Migration

mit festem Einstiegspunkt (z.B. Aglets)

```
public class Display
           implements java.io.Serializable
   Boolean isRemote;
   public class Example1 extends Aglet {
       public void onCreation (Object init) {
       addMobilityListener( new MobilityAdapter() {
           public void onArrival( MobilityEvent e) {
                 System.out.println(„i am remote“);
                 isRemote = true;
           }
       }
   };

public void run() {
   if (! isRemote) {
           moveTo(„Host1“);
   }
}
```

mit beliebigem Einstiegspunkt (z.B. Voyager)

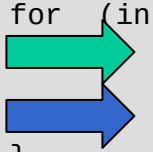
```
public class Display
           implements java.io.Serializable
{
   public void display(String message )
    {      System.out.println( message);    }
}

public class AgentClient {
    public static void main( String[] args )
   {
      Display agent1 = Proxy.of( new
                                  Display() );
   Agent.of( agent1 ).moveTo(
     "//host:8000", "display",
     new Object[]{
        „show on remote";
}
```

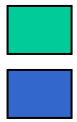
Ausstiegspunkt Host A

Einstiegspunkt Host B

Starke Migration (= Transparente Migration)

```
public class AgentClient extends StrongAgent {
    public static void main( String[] args )
   {
           String Hosts[] = { „Host1“, ... „Hostn“ };
           for  (int i=0; i < Hosts.length(); i++) {
                   //lokal
                   moveTo(Hosts[i] );
                   // remote
           }
   }
};
```





Ausstiegspunkt Host A

Einstiegspunkt Host B

- Einstiegspunkt = Ausstiegspunkt
- Möglich durch Serialisierung des Ausführungsstacks (z.B. Program Counter)
- Selten Umgesetzt da sehr schwierig zu realisieren; z.B. in D‘Agents (AgentTCL)

=> je schwächer Migrationsart, desto mehr Aufwand hat der Programmierer

=> je stärker die Migrationsart, desto komplizierter für das Agentensystem

Migration und Java

- Serialisierung erfolgt durch *Java Objekt Serialisierung*

```
class Agent implements
          java.io.Serializable
```

- Datenzustand kann separat angegeben werden

serialisiert:

```
class {
          int a;
          static String b;
```

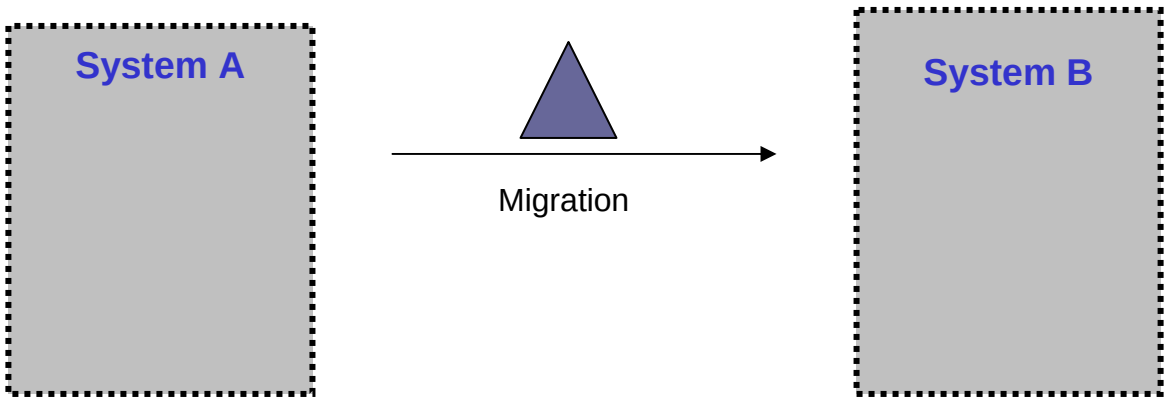
nicht serialisiert:

```
class {
           transistent int c;
```

- Starke Migration wird nicht direkt unterstützt. Möglichkeiten:

1. Durch Anpassung der JVM (z.B. CIA)
   - keine Kompatibilität zu anderen Systemen
2. Durch Anpassung des Sourcecodes
   - sehr Aufwendig zu Programmieren (siehe Fünfröcken ‘98)

Migration Ablauf



1. Ausführung des Agenten unterbrochen
2. Identifikation der zu übertragenen Teile des Agenten
3. Agent serialisieren
4. Enkodierung entsprechend dem verwendeten Transport Protokoll
5. Authentifikation bei System B
6. Agent mit Code und Zustand verschicken

1. Authentifizierung von System A
2. Decodierung
3. Deserialisierung
4. Instanzzierung
5. Agenten Zustand wiederherstellen
6. Fortsetzen der Ausführung

nach MAFO-Spezifikation

Push vs. Pull Migration (wann migriert was ?)

1. **Push Migration**
   - Migriert werden Datenzustand + Klassen des Agenten
   - Eingesetzt z.B. von Grasshopper, Aglets

2. **Pull Migration**
   - Migirert wird zunächst nur der Datenzustand
   - Benötigte Klassen werden nach Bedarf einzeln nachgeladen
   - Vorteil: u.U. Einsparung von Bandbreite
   - Nachteil: Bei kleinen Agenten zusätzliche Bandbreite (siehe RFC vs. MA)
   - Eingesetzt z.B. von Voyager

Kommunikationsbeziehungen

Agent <-> Agent	Kooperation
Agent <-> System	Aufgabenerfüllung
System <-> System	Steuerung, Agentensuche

Kommunikation zwischen Agenten (1)

1. I.S. eines mobilen Agenten

- Agent migriert auf das System des anderen Agenten (oder auf eines in dessen Nähe)
- Agent Kommuniziert lokal (z.B über RPC) mit diesem Agenten

2. i.S. eines stationären Agenten

- Möglichkeiten unterscheiden sich je nach Agentensystem
- Beispiele: Plain Sockets, RPC, Corba

Kommunikation: zwischen Agenten (2)

1. **Direkt**
   - Agent sendet Nachricht direkt an Adresse des anderen Agenten
   - Adressierung: Agent -> AgentABC@AgentHost ...

- **Indirekt**
  1. Benannt (durch Mailboxen)
     - Agent schickt Mitteilung an Mailbox des anderen Agenten
     - Adressierung: Agent -> „MailboxABC“
  2. Anonym (durch Blackboards)
     - Agent schickt Mitteilung an ein Blackboard (Schwarzes Brett) ohne direktem Agentenbezug
     - Adressierung: Agent -> „Queue123“

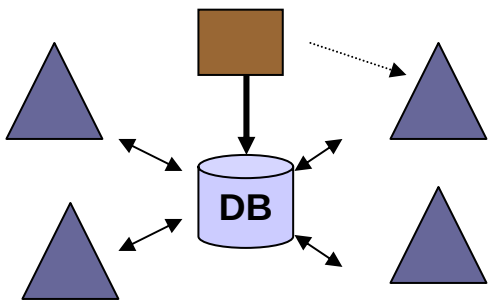
Agententracking

Motivation:

- wo befindet sich ein Agent ?
- Agent zurückholen
- Agentenstatus abfragen

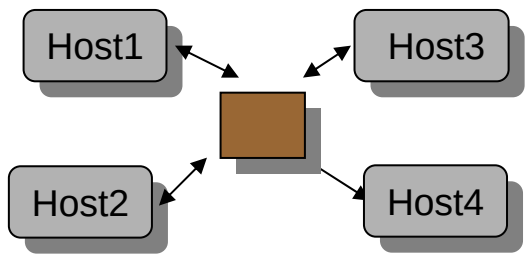
Zwei Ansätze:

1. Zentrale Registierung



=> hilft auch zentraler Überwachung von Agentenlebenszyklen (z.B. Terminierung von Waisen)

2. Broadcast



Sicherheit: für Agentensysteme

vor böswilligen Agenten

- Agenten werden von nicht-autorisierten Server geschickt
- geben falsche Identität aus
- greifen Agentensystem an (z.B. Denial-of-Service Atack)

Ansätze: Sicherheitsarchitektur des Agentensystems. U.A.:

- Signaturen
- JDK Sandbox
- Access Control Wrapper

Beispiel: Voyager Security Manager:

- Unterscheidung zwischen bekannten (Native Objects) und unbekannten Agenten (Foreign Objects)
- Zulassung nur bestimmter Operationen (z.B. Create, Listen, Migrate)

Sicherheit: für Agenten

1. **bei Transport / Migration**
   - Man-In-The-Middle
2. **vor böswilligen Agenten**
   - spähen Agenteninformationen aus
3. **vor böswilligen Agentensystemen**
   - senden Agent an anderen / falschen Ort weiter
   - spähen Agenteninformationen aus
   - verändern Code
   - manipulieren Abfragen

Lösung (Ansatz):

- Verschlüsselung + Signierung des Agenten (z.B. SSL, PGP)
- Migration nur zu vertrauenswürdigen /geschlossenen Systemen (z.B. durch Zertifizierungsstellen)
- keine sensiblen Daten im Agent ablegen (Was wird serialisiert ?!)
- Vollständige Absicherung (insbesondere für Fall 3) nur schwer möglich

Zusammenfassend

- Über 78 Agentensysteme*. Darunter:
  - AgentTCL alias D‘Agents (TCL)
  - IBM Aglets (Java)
  - Odysee (Java)
  - Voyager (Java)
  - SeMoA (Java)
  - Grasshopper (Java)
  - ...

- Teilweise sehr unterschiedliche Konzepte (Migrat. / Komm. / Sicherh.)
- Kaum Interoperlabilität / Agenten-Portabilität zwischen diesen Systemen

Quelle: The Mobile Agents List: (mole.informatik.uni-stuttgart.de/mal/mal.html)

Gliederung

1. Was sind mobile Agenten ?
2. Aspekte eines Agentensystems
   - Migration
   - Kommunikation
   - Sicherheit
3. Standardisierung
4. Fazit

Standardisierung

Motivation: Interoperlabilität zwischen unterschiedlichen Agentensystemen bisher nicht (oder kaum) möglich

Lösung: Vereinheitlichung von

- Migration
- Kommunikation
- Sicherheit

durch **Standards**

vorhandene Standards

1. FIPA (1997-1999)

- **F**oundation of **I**ntelligent **P**hisical **A**gents (www.fipa.org)
- Gebildet von 24 Unternehmen
- Schwerpunkt **Agentenkommunikation** (FIPA-ACL)

2. Masif (1998-1999)

- **M**obile **A**gent **S**ystem **I**nteroperability **F**acilities Specification
- Von der OMG - Object Management Group (www.omg.org)
- In OMG über 800 Unternehmen
- Andere Standards der OMG: UML, Corba
- Schwerpunkt: **Agentenmigration**

Andere:

- Open Agent Architecture (OAA)
- Grid Mobile Agent System (GMAS)

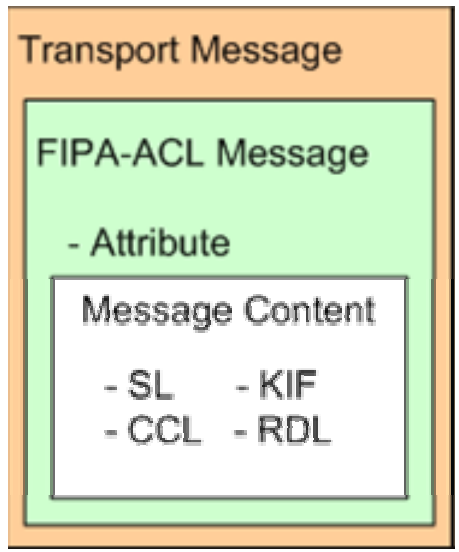
FIPA-ACL

Ansatz:

Definition einer systemtunabhängigen Agenten Kommunikations-Sprache (ACL) auf Basis von KQML* zur Kommunikation zwischen Agenten und anderen Agenten, Systemen und Menschen.

Bestandteile

- Nachrichtentransport über Message Transport Service (MTP):
  - Interne Kommunikation beliebig
  - Externe Kommmunikation, z.B. über Corba-IIOP
- Formulierung der Inhalte durch Wissensrepräsentationssprachen (z.B. KIF, FIPA-SL)
- Begriffsbestimmungen durch Ontologien
- Basiert auf Sprechakt-Theorie
- FIPA definiert 22 Kommunikations-Akte (z.B. Inform)



Aus: Steinecke 2003

*** www.kqml.org**

FIPA-ACL: Beispiel Sprachakt

Berlin

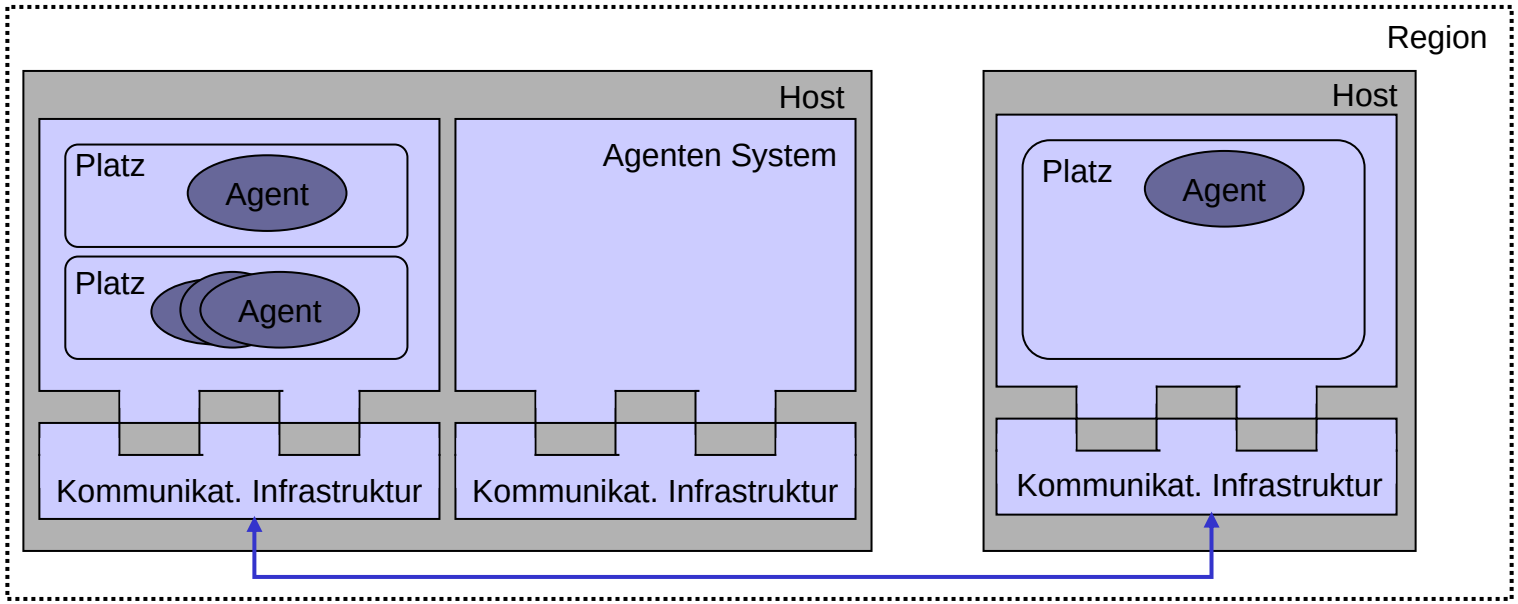
Anfrage: Wie ist das Wetter ?

Information: Es regnet

Heidelberg

```
(inform
        :sender (agent-identifier :name heidelberg)
        :receiver (set (agent-identifier :name berlin))
        :content („weather(today,raining)”)
        :language Prolog
        :ontology weather42)
```

MASIF Terminologie



Plätze (Place)

Kontext, Umgebung in dem ein Agent ausgeführt wird.

Location (= Agenten Adressierung)

Host + Agentensystem + Platz

Kommunikations Infrastruktur

Verbindet mehrerer Agentensysteme miteinander

Agent Profile (= Migrations-Kombatibilität)
Agent System Type + Sprache + Serialisierungsmetode

Autoritäten (Authorities)

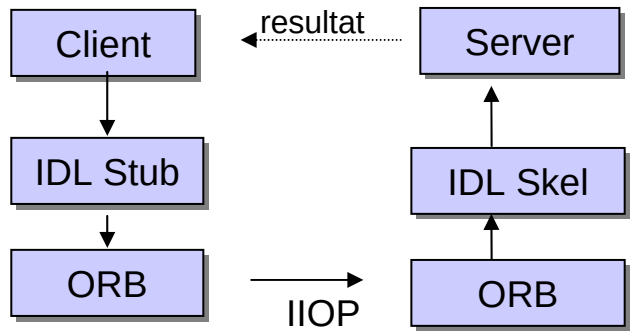
Person oder Organisation, der ein Platz oder ein Agentensystem zugewiesen sein muss

Regionen

Mehrere Agentensysteme mit gleicher Autorität

Hintergrund: Corba

- **C**ommon **O**bject **R**equest **B**roker **A**rchitecture
- Middleware Architektur für die Verteilung und Kooperation objektorientierter Softwarebausteine in heterogenen, vernetzten Systemen
- Eigenschaften: orts-, plattform- und implementations-unabhängig
- Bestandteile:

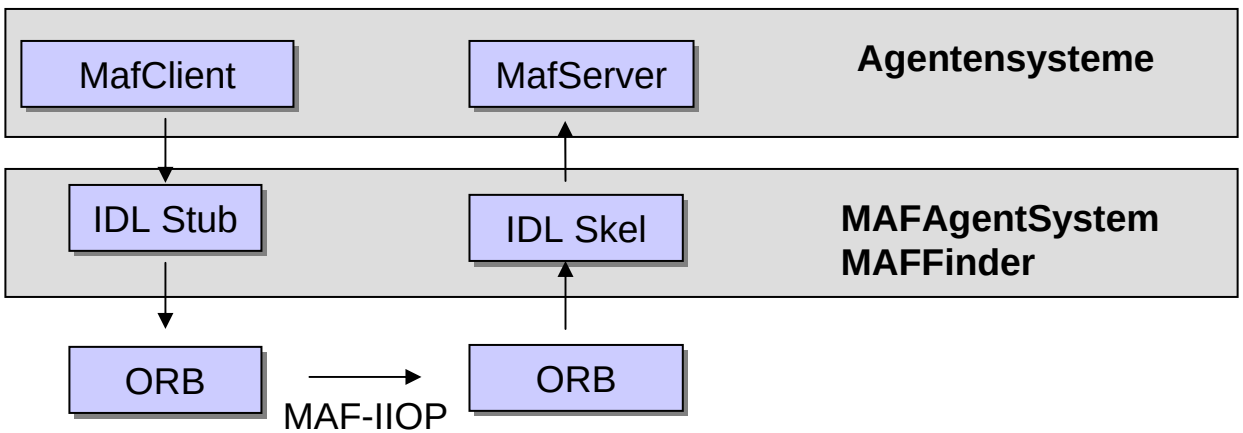


1. **ORB** (= Object Request Broker)
   - Konkrete Corba Implementierung (z.B. Avantis)
2. **IIOP**
   - Protokoll zur Kommunikation zwischen ORBs
3. **IDL** (= Interface Definition Language)
   - implementations-unabhängige Beschreibungssprache von Objekschnitstellen
4. **Services**
   - z.B. Naming-Service: hilft bei der Suche und Identifikation von Objekten

Corba und MAF

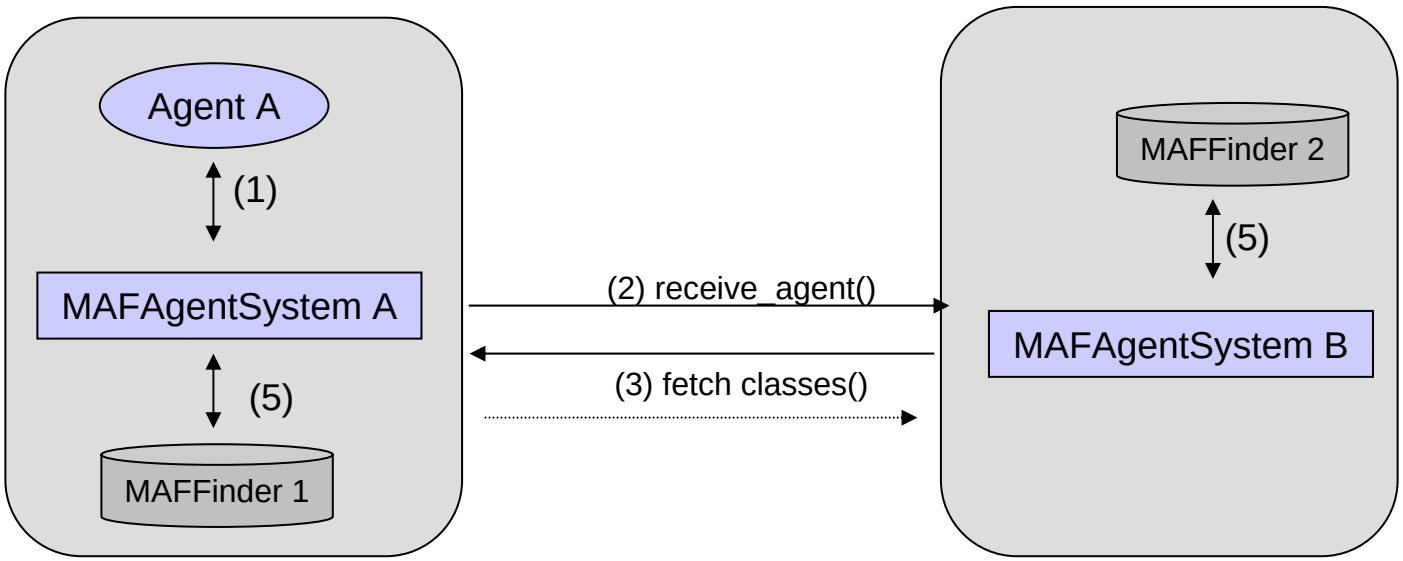
Zwei IDL-Schnittstellen vorgegeben:

1. **MAFAgentSystem:**
   - Aufgabe: Agententransport
   - Operationen: receive(), create(), suespend(), terminate()
2. **MAFFinder:**
   - Aufgabe: Finden von Agenten (Agent Tracking)
   - Operationen: register(), unregister(), locate agent(), locate agent system()



Adressierung: *mafiiop://host:port/agentensystem/place* **(= Location)**

Umsetzung der MAF

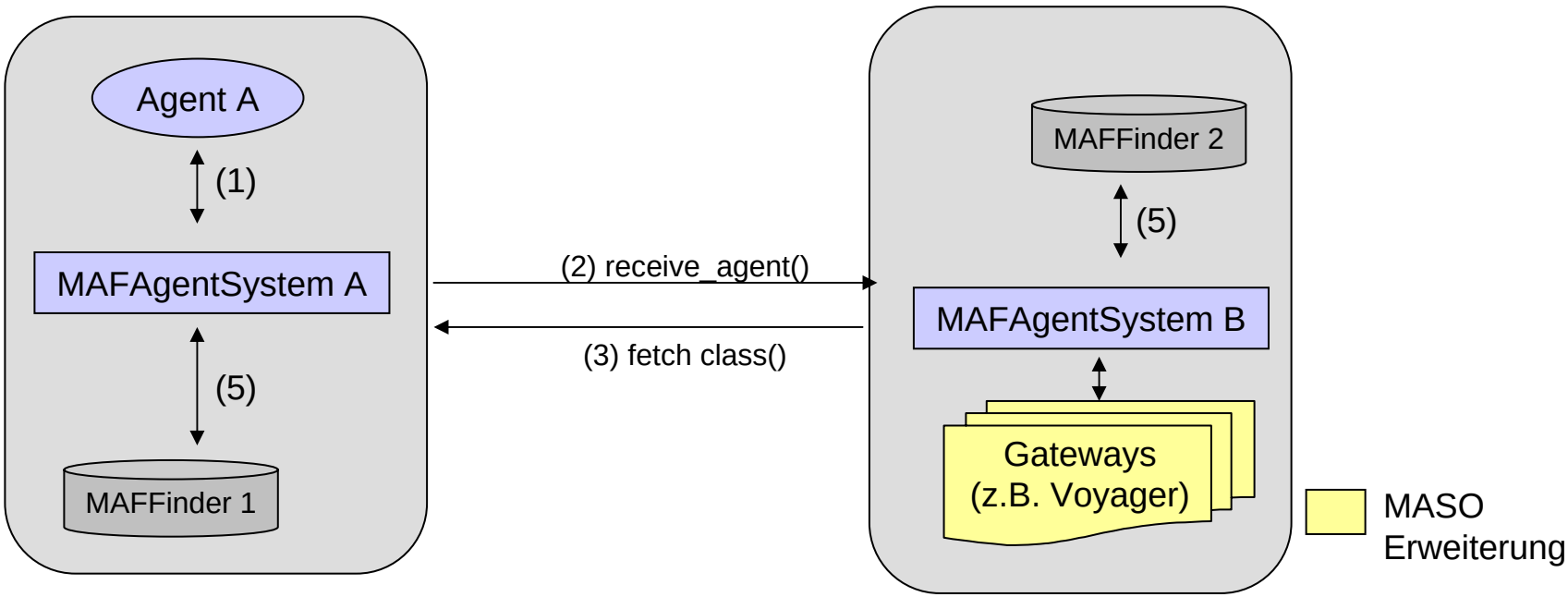


1. Agent A stellt Migrationsanfrage an sein AgentenSystem (A)
2. System A stellt Migrationsanfrage (Profil, Klassenliste)
3. Wenn System B Profil unterstützt:
   - (Stoppen von Agent A auf System A)
   - Abrufen der Agenten-Klassen über System A
4. (Starten von Agent A auf System B)
5. Umregistrierung des Agenten bei MAFFinder 1+2

Umsetzung von MASIF (Ansatz: MASO)

Problem: MASIF sagt nichts über die Gestalt von Agenten aus

=> Konkrete Implementierung von Agenten-Portabilität bleibt offen



- Ansatz MASO: Unterstützung zusätzlicher Agenten-Profile durch zusätzliches Gateways
- Hoher Aufwand
- Nie wirklich vollständig umsetzbar

To Do

Agenten Spezifikation:

- Sicherheit (Mindestanforderungen Verschl. / Authentifizierung / Sign.)
- Migration (festgelegte Migrationsarten)
- Management (Agentenlebenszyklus, Globaler Agentennamensdienst)
- Zertifizierung (MA-Zertifizierungsstelle zertifiziert Agentensysteme)
- Anwendungen (z.B. Shopping Agent, Such-Agent)

Gliederung

1. Was sind mobile Agenten ?
2. Aspekte eines Agentensystems
   - Migration
   - Kommunikation
   - Sicherheit
3. Standardisierung
4. Fazit

Fazit / Ausblick

- Viele Mögliche Anwendungsgebiete (v.a. durch steigende KI)
- Aber keine „Killerapplikation“ in Sicht
- Viele Unterschiedliche / Inkompatible Konzepte von Agentensystemen
- Standards lassen viele Punkte offen:
  - Agenten
  - Sicherheit
  - Anwendungen
- Kaum Umsetzung der vorhandenen Standards
- Konkrete Ansätze zur Portabilität in vorhandenen Systemen kaum erkennbar
- Mobile Agenten bisher noch hauptsächlich theoretisches Disziplin

Vielen Dank für Ihr Interesse!

Fragen?