Welcome HPCS Conference 2009

- Agent Tutorial -
Technology for the Mainstream?

Chair of
Software Engineering

Friedrich Schiller University Jena
Volkmar Schau

photo: Ernst-Abbe-Platz, Thüringen Tag 2006, Steffen Kern
Agenda

1. Agent Definition
   • Migration
   • Blueplanet cluster / simulation

2. Agent Oriented Design
   • Processes
   • Notation
   • How to write agent-oriented Software

3. Usecase Examples
   • Future of the History

4. Summary and Conclusion
Agent Definition
New Dimension

Social aspects

Web 1.0

- technology
- individual
- consumption
- function calls

Web 2.0?

- flexible
- hard
- fixed
- clustered
- html

Technical aspects

- device
- binding
- system
- media
- interaction
- usage
- creation
Next Generation of Interaction - Web 2.0

**Business Models** that proved to survive and have promise for the future.

**Approaches** such as
- services (instead of products),
- the web as application platform (instead of a data repository)

**Concepts** such as
- Folksonomies (instead of taxonomy),
- Syndication (instead of integration)
- Participation (instead of consumption)
- Reputation (instead of Anonymity)

**Technologies** such as
- AJAX, REST, Tags, Microformats, ...
Social Communities

services / cell phone / virtual space

- platform diversity
- limited social networking
- no integrated view

Human-Centered Social Mobile Community

- social community
- interrelation with real life
- integrated services
*) classification based on Franklin and Graesser
Definition (II)

Software Agent
- software entity with specific properties (autonomous, proactive, self-aware, social, personalizable)
- task by order of user
- communication by messaging
- base for agent-oriented software development

Mobile Agent
- capability for migration between nodes within a network
- migration task takes out the current state and agent code
- code-shipping versed data-shipping
Definition (III)

Agent as entity who ..

- operates independently,
  - autonomy
  - capacity to follow his own intentions
- act as representative of another entity (principal).
  - representative
  - assistant to realize pre-defined objectives

Conflict between autonomy and to be a representative (agent vs. principal)

- autonomy boundary settings by pre-defined objectives
- Limit settings for objective realizing by degree of freedom
Case study

Principal-agent challenge in microeconomics

Agency theory is concerned with resolving two problems that can occur in agency relationships.

The first is the agency problem that arises when
- the desires or goals of the principal and agent conflict and
- it is difficult or expensive for the principal to verify what the agent is actually doing. The problem here is that the principal cannot verify that the agent has behaved appropriately.

The second is the problem of risk sharing that arises when the principal and agent have different attitudes toward risk. The problem here is that the principal and the agent may prefer different actions because of the different risk preferences.

Kathleen M. Eisenhardt:
Agency Theory: An Assessment and Review,
Case study (II)

**Effect of Principal-agent challenge**

- Principal's persuasives (beliefs) and preferences (desires) should be agent's motivation
- Depending on agent's autonomy to fulfill principal's task is a measure for principal's benefit of agent results:
  
  autonomy $\rightarrow$ surprisal value $\rightarrow$ benefit of information

- Based on agent's benefit principal can learn to form stronger persuasives and preferences and can improve agent's funktionality as well

  benefit of information $\rightarrow$ learning effect $\rightarrow$ operation improvement
**Definition (IV)**  

**[Jennings & Wooldridge, Russell & Norvig]**

<table>
<thead>
<tr>
<th><strong>Weak Notion</strong> (essential properties)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Autonomy</strong></td>
<td>agents operate without direct intervention of humans, and have control over their actions and internal state</td>
</tr>
<tr>
<td><strong>Social ability</strong></td>
<td>agents interact with other agents (and possibly humans) via an agent communication language</td>
</tr>
<tr>
<td><strong>Reactivity</strong></td>
<td>agents perceive their environment and respond in a timely and rational fashion to changes that occur in it</td>
</tr>
<tr>
<td><strong>Pro-activeness</strong></td>
<td>agents do not simply act in response to their environment, they are capable of taking the initiative (generate their own goals and act to achieve them)</td>
</tr>
</tbody>
</table>

**Strong Notion** (an agent has mental properties, such as knowledge, belief, intention, obligation)

| **Mobility**                  | agents can move around from one machine to another and across different system architectures and platforms |
| **Veracity**                  | agents do not knowingly communicate false information |
| **Benevolence**               | agents always try to do what they are asked of |
| **Rationality**               | agents will try to achieve their goals and not act in such a way to prevent their goals from being achieved |

**Russell und Norvig**

**Definition** : “An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.”
Migration (I)

Read data

Communicate / Collaborate

Migration

Read data
Blueplanet Cluster / Simulation

Planetary Ecosystem
- Carbon cycle
- Day-/Nighttime and Seasons

Virtual Monad
- Metabolism
- Formation of swarms
- DNA

Genetic Code
- Mutation & Recombination
- Evolution

if (sun_intensity < 0.1) then
  if (org_mat > 250) then
    create_child
  else
    pass
  end
else
  photosyn
end
Migration (II)

**Definition**
Describe the change from agency to agency (Agency to Agency != Node to Node)

**Approach**
- client/server versus agent
- code-shipping versus data-shipping

**Goals**
- Decrease network load
- Avoiding latency and network failure
- Distributed data
- Strategy to develop distributed applications / systems

**Migration Process**
- Stop agent execution and save (mental) state
- Transfer state and code to destination node
- Restore agent execution state and (re)start agent execution
**Agent System**
- agent-based application
- system consists of single or multi agencies and executes agents

**Agency**
- comparable with Java EJB- or Servlet-container
- Agent execution environment
- principal task: scheduling und security
- allocate basic functions

**Classification**
1. *Generation*
   Static task and static route
2. *Generation*
   Static task and flexible route (compute by own)
1. *Generation*
   Abstract task description and flexible route
Related Concepts

**Internet Agents, Spider**
- robots, crawler
- remote access to network resources

**Java Applets**
- web pages with small Java programs

**Java Servlets, Java Server Pages**
- dynamic web pages with embedded Java-Code
- started by client are executed on server side

Why these concepts are no mobile agents?

„Software isn't able to proactively change the place of execution — no migration is possible.“
Operation mode

Result:
• Events
• next task

Content provider
(Library Jena)

Communication
with client application

Content provider
(Univ. Applied Science Jena)

Migration

Alert! New information.

Processing task

Configuration

Migration

Processing task

Communication
with client application
Agent Oriented Design
AOP (I) : Waterfall Approach

Analysis
Stop
The Perfect Plan

Develop
Stop
Coding Complete

Test
Stop
Golden Master

Costs / Volume

Time
Quality
AOP (II)
AOP (III): What’s the goal?

Agent based modeling

- business logic
- exchangeability by runtime
- software modeling style
- client/server versus agent
- code-shipping versus data-shipping
AOP (IV) : What do we have?

**OOAP**

Notation / Profile : UML

Model : Client/Server / ..

Process : Waterfall / Agile Development / ..

---

**AOP**

UML / AUML / AML + Profiles for UML

BDI / ..

MDA / Prometheus / ADEM

---

AML Notation

Prometheus Notation
AOP (V): What’s new?

- domain knowledge
- ontology
- goals, intentions
- agent behavior/mental modeling
- BDI (ACMAS), AGR (OCMAS), ..

- two worlds, one process
- final artifacts, intermediate design tools
- roles
- documents, model and model fragments
Approaches: AOP [Shoham]

Agent Oriented Programming is based on „social“ programming

- agent programming by mentalistical and intentional terms
- use BDI concepts

AOP system of three components by Shoham

- Logic system to define mental agent states
- Agent programming language interpreter based
- Compiler to execute agent programs in low-level systems

AOP interpreter with AGENT-0 language
Deduction / BD theory

Persuasives

World situation at that time
Deduction / BD theory

World situation at that time

1st valid possible world situation at later date

Nth valid possible world situation at later date

Persuasives

Preferences
Deduction / BD theory

World situation at that time

Persuasives

1st valid possible world situation at later date

Preferences

Nth valid possible world situation at later date

Actions
Deduction / BD theory

Persuasives

Preferences

World situation at that time

1st valid possible world situation at later date

Actions

Nth valid possible world situation at later date

Preferences
Deduction / BD theory

- Preferences
- Persuasives

World situation at that time

1st valid possible world situation at later date

Nth valid possible world situation at later date

Actions

Optimisation of subjective benefit
UTC – Unified Theories of Cognition

- Newell’s Idea: Psychology micro theories
  - UTC as took box for human cognition

  “Psychology has arrived at the possibility of unified theories of cognition – theories that gain their power by positing a single system of mechanisms that operate together to produce the full range of human cognition.” (A. Newell, 1990)

Some questions UTC should give an answer

- How flexible interact intelligent organisms with their environment?
- How can we describe goal oriented behavior`?
- How do we choose goals in rational terms?
- How do we use symbols?
- How do we create experiences?

SOAR architecture as a result of Newell's theory
Architecture: SOAR (II)

- Rules
- Semantic knowledge
  - Deklarative structures
- History knowledge
- Environment interface
- Long-Term Memories
  - Procedural
  - Semantic
  - Episodic
- Working Memory
  - Perception
  - Action
- Execute all data (environment data, hierarchy of states and dedicated operators)

Architecture: SOAR (II)
Approaches (III) : AGR [Ferber et al.]

Agent-Group-Role theory describes organization centred MAS

- Organisations consists of agents playing different roles and are organized in lap structures
- Agent role is described by behavior and function

Organisations can be considered in two different levels

- Organisation structure
  - abstract structure describing „how“
- Concrete organisation
  - Realisation of agent based organisations

Organisation by two aspects

- structure and dynamic
Theories: AGR (II)

UML Metamodell

- Agent
- Group
- Role
- RoleConstraint
- Interaction protocol
- Correspondence
- Dependency

Sequenzdiagramm (organisation dynamic)

- Group 1
  - Role 1
  - Role 2
  - Role 3

- Group 2
  - Role 4
  - Role 5

Organization structure

GS – Gruppenstruktur
R – Rolle
I – Interaktion
Platforms

- Middleware Platform
- Intelligent Platform
- Social Platform

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>URL</th>
<th>Availability</th>
<th>Tools</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>JADE</td>
<td>Middleware</td>
<td><a href="http://jade.tilab.com">http://jade.tilab.com</a></td>
<td>open source</td>
<td>A, D</td>
<td>Tutorials, Guides</td>
</tr>
<tr>
<td>Tryllian ADK</td>
<td>Middleware</td>
<td><a href="http://www.tryllian.com">http://www.tryllian.com</a></td>
<td>commercial</td>
<td>A</td>
<td>Userguide</td>
</tr>
<tr>
<td>JACK</td>
<td>Reasoning</td>
<td><a href="http://www.agent-software.com">http://www.agent-software.com</a></td>
<td>trial-version</td>
<td>I, D</td>
<td>Tutorial, Guides</td>
</tr>
<tr>
<td>Agentis</td>
<td>Reasoning</td>
<td><a href="http://www.agentissoftware.com">http://www.agentissoftware.com</a></td>
<td>commercial</td>
<td>I, A, D</td>
<td>-</td>
</tr>
<tr>
<td>SOAR</td>
<td>Reasoning</td>
<td><a href="http://sitemaker.umich.edu/soar">http://sitemaker.umich.edu/soar</a></td>
<td>open source</td>
<td>I, D</td>
<td>Tutorial, Guides</td>
</tr>
<tr>
<td></td>
<td>Reasoning</td>
<td><a href="http://www.soartech.com">http://www.soartech.com</a></td>
<td>open source</td>
<td>I, D</td>
<td>-</td>
</tr>
<tr>
<td>MadKit</td>
<td>Social</td>
<td><a href="http://www.madkit.org">http://www.madkit.org</a></td>
<td>open source</td>
<td>A, D</td>
<td>Guides</td>
</tr>
</tbody>
</table>

I: IDE, A: Administration, D: Debugging
Agents find one’s way into in god's world. Today's hip hero calls a personal assistant his own. So it's done by god Bachus (god of vine) for his abundance of tasks. He is appreciated for all help done by genie Vino, his personal assistant. So Vino manages all communication like email, sms etc. Right now god Bachus forgets about Vino’s workplace to fulfill his last task.

Now we come into play. Please help the god of vine to find his loyal assistant. The only indication for Vino's workplace is given by vitner Jean. At the moment he is also busy working in his vineyard but he sends us his personal vine assistant, agent Jean. You can call agent Jean at the remote place

ipc047.inf.uni-jena.de:31000

Please ask Jean „Dear Jean, who is the genie of the bottle?“ and follow his direction by contacting Vino's computer. In case you are able to contact Vino please be polite to him – he is the messenger of god Bachus. Send him greetings „How are you?“. As Bachus's messenger he could give an advice for you.

AOP example I

Sense of life
AOP example I: Requirements

**TracySE Agent Toolkit**
http://wiki.tracy.informatik.uni-jena.de/

**Prometheus PDT**

**SUN Java VM**
http://java.sun.com/javase/downloads/5/jdk

**Security Provider**

**Build system**
http://archive.apache.org/dist/ant/binaries/apache-ant-1.6.5-bin.<archive>
TracySE Overview
TracySE: How to get a Context?

1. `getContext(Service)` uses API

2. `getContext(Agent)` ```

3. `resultContext`

4. `resultContext`

5. `invoke`

6. `result`

5.1. `invoke`

5.2. `result`

Plugin

Agent

Context

Agency Management

MicroKernel

*1* calls

 activates

 activates

 uses API

*1* context

initializes communication

invoke

result

invoke

result

invoke

result

invoke

result
Example agent in TracySE

- Every agent owns his unique name: spoken name (nickname), UUID, home agency
- Bond.A8B3456AEC34@MainAgency.lucent.com (abbreviated)
- Voting protocol, two-phase comit protocol

```java
public class MyAgent implements Runnable {
    public void run() {
        IAgentMessageContext cxt;
        cxt=(IAgentMessageContext)Context.getContext("message");
        cxt.sendMessage( "blofeld", "message text" );
    }
}
```

• plugin specify agent's state
• plugin enforce tasks
• agency drop agent
AOP example I

Sense of life

Agents find one's way into in god's world. Today's hip hero calls a personal assistant his own. So it's done by god Bachus (god of vine) for his abundance of tasks. He is appreciated for all help done by genie Vino, his personal assistant. So Vino manages all communication like email, sms etc. Right now god Bachus forgets about Vino's workplace to fulfill his last task.

Now we come into play. Please help the god of vine to find his loyal assistant. The only indication for Vino's workplace is given by vitner Jean. At the moment he is also busy working in his vineyard but he sends us his personal vine assistant, agent Jean. You can call agent Jean at the remote place

ipc047.inf.uni-jena.de:31000

Please ask Jean „Dear Jean, who is the genie of the bottle?“ and follow his direction by contacting Vino's computer. In case you are able to contact Vino please be polite to him – he is the messenger of god Bachus. Send him greetings „How are you?“. As Bachus's messenger he could give an advice for you.
Implication: agent vs. data object

Mobile agents in BDI terms

- Persuasives = Data (content/variable) und Running state
- Preferences = Agent’s lifecycle states
- Intentionen = Code (agent program)

Mobile agent’s Proactivity

Unenforced transitions between preference states (agent lifecycle states)

Principal’s Intensions

- Principal's intensions are reflected by (system) enforced agent state changes
- Edge condition of agent's proactivity
Implication (II): Security

**Spy**
- code, data, control flow, communication

**Manipulation**
- code, data, control flow, communication

**Denial Service**
- terminate, resource blockade, spamming

**Masquerade**
- usage in combination of attacks mentioned before
AOP example II

**Online (book) proceedings store**

We would like to develop a fully online system for worldwide sale of proceeding books. The system will offer a broad range of books to customers, and a personalized, friendly user interface.

The system must facilitate fast and reliable system at all stages, from locating a desired proceeding book, to delivery of the purchase.

The store should have competitive prices.
AOP Methodology: Prometheus
Prometheus: System Spec.

- Specification of **system goals**, resulting in a list of goals and sub-goals, with associated description.
- Development of a set of **scenarios** that have adequate coverage of the goals, and which provide a process-oriented view of the system to be developed.
- Definition of a set of **functionalities** that are linked to one or more goals, and capture a limited piece of system behavior, which can be described in a few sentences.
- Description of the **interface** between the agent system and the environment in which it is situated, in terms of incoming percepts, outgoing actions and external information stores with which the system will interact.
Online (book) proceedings store

We would like to develop a fully online system for worldwide sale of proceeding books. The system will offer a broad range of books to customers, and a personalized, friendly user interface.

The system must facilitate fast and reliable system at all stages, from locating a desired proceeding book, to delivery of the purchase.

The store should have competitive prices.
Prometheus: Architectural Design

- Deciding on the agent types used in app. Group functionalities into agents considering alternatives.
- Describe the interaction between agents using interaction diagrams and interaction protocols.
- Design the overall system structure (boundaries, percepts, actions, relationships, data).
Prometheus: Detailed Design

- Further decomposition using capability
- Sub-tasks and alternative plans
- Developing details (events, messages, percepts, action and data)
Usecase Examples
Realisation by linkage Project „MobiSoft“

**Project’s objective**
- Investigate future usage scenarios of mobile agent technology
- linkage research project

**Project’s goal**
- Personalised electronic assistants for roaming mobile device users
  - smart phone
  - PDA/MDA
  - notebook
- Assistants are proxies of users which autonomously invoke
  - interaction
  - cooperation
  - negotiation
  - discovery of information
  - observation

URL http://mobisoft.informatik.uni-jena.de/
System based on Mobile-C (I)
System based on Mobile-C (II)
Knowledge Navigator
Summary
### „Agents technology ready to use“

<table>
<thead>
<tr>
<th>Efficiency savings</th>
<th>CPU consumption is limited, because a (mobile) agent execute only on one node at a time. Other nodes do not run an agent until needed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space savings</td>
<td>Resource consumption is limited, because a (mobile) agent resides only on one node at a time. In contrast, static multiple servers require duplication of functionality at every location. Mobile agents carry the functionality with them, so it does not have to be duplicated. Remote objects provide similar benefits, but the costs of the middleware might be high.</td>
</tr>
<tr>
<td>Reduction in network traffic</td>
<td>Code is very often smaller than data that it processes, so the transfer of (mobile) agents to the sources of data creates less traffic than transferring the data. Remote objects can help in some cases, but they also involve marshalling of parameters, which may be large.</td>
</tr>
<tr>
<td>Asynchronous autonomous interaction</td>
<td>Mobile agents can be delegated to perform certain tasks even if the delegating entity does not remain active.</td>
</tr>
<tr>
<td>Interaction with real-time systems</td>
<td>Installing a (mobile) agent close to a real-time system may prevent delays caused by network congestion. In Network Management systems agents usually reside close to the hardware, so this advantage might not be as clear as others.</td>
</tr>
<tr>
<td>Robustness and fault tolerance</td>
<td>If a distributed system starts to malfunction, then (mobile) agents can be used to increase availability of certain services in the concerned areas (some kind of meta-level management of agents is required to ensure that the agent based system fulfills its purpose).</td>
</tr>
<tr>
<td>Online extensibility of services</td>
<td>Mobile agents can be used to extend capabilities of applications, for example, providing services. This allows for building systems that are extremely flexible.</td>
</tr>
<tr>
<td>Convenient development paradigm</td>
<td>Creating distributed systems based on mobile agents is relatively easy. The difficult part is the mobility framework. High-level, rapid prototyping environments for agents will be needed. It is quite probable that the tools for object-oriented programming will evolve into agent-oriented development environments.</td>
</tr>
</tbody>
</table>
Future Challenges

**Quantity and Heterogeneity** - global information system
- number of available services increase every time
- different system architectures

**Quality** of service differs
- bandwidth, latenz
- availability
- Reliability

**Mobility** of user
- network connection is established in different places and times

**Performance** - nodes of distributed systems may be less powerful
- nodes: laptops, palmtops, mobile phones, e-books
- additionally refrigerators, cars and wearable computers

**Artificial life**
- artificial and social intelligence
Thank you for your time.

Questions and Answers

Dipl.-Inf. Volkmar Schau

Cell  +49-160-95684294
Mail  volkmar.schau@uni-jena.de
Web   swt.informatik.uni-jena.de