Software Technologies

Mobile Code

Sebastian Fischmeister
fischmeister@softwareresearch.net
University of Salzburg

Mobile Agent Standardization

- Object Management Group (OMG) Agents Working Group
  - Recommends standards for agent technology
  - Mobile Agent System Interoperability Facilities (MASIF) – draft specification
  - www.omg.org
- FIPA - Foundation For Intelligent Physical Agents
  - Non-profit organization which promotes the development of specifications of generic agent technologies that maximize interoperability within and across agent-based applications
  - FIPA 98 - seven part specification
  - www.fipa.org
- New bottom up approaches

Mobile Agent Terms

From the OMG MASIF specification

- **Agent**
  - An agent is a computer program that **acts autonomously on behalf of a person** or organization. Currently, most agents are programmed in an interpreted language (for example, Tcl and Java) for portability. Each agent has its **own thread of execution** so tasks can be performed on its own initiative.
- **Stationary Agent**
  - A stationary agent executes only on the system where it begins execution. If the agent needs information that is not on that system, or needs to interact with an agent on a different system, the agent typically uses a communications transport mechanism such as Remote Procedure Calling (RPC). The communication needs of stationary agents are met by current distributed object systems such as CORBA, DCOM, and RMI.

3

Mobile Agent Terms

- **Mobile Agent**
  - A mobile agent is not bound to the system where it begins execution. It has the **unique ability to transport itself** from one system in a network to another. This submission is primarily concerned with mobile agents. The ability to travel permits a mobile agent to move to a destination agent system that contains an object with which the agent wants to interact. Moreover, the agent **may utilize the object services** of the destination agent system.
- **Agent State**
  - When an agent travels, its **state and code are transported with it.** In this context, the agent state can be either it's execution state, or the agent attribute values that determine what to do when execution is resumed at the destination agent system. The **agent attribute values include the agent system state** associated with the agent (e.g. time to live).
- **Agent Execution State**
  - An agent's execution state is its runtime state, including program counter and frame stacks.

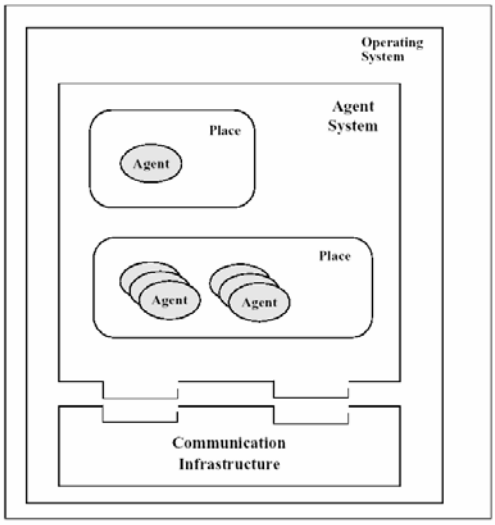
4

Mobile Agent Terms

- **Agent Location**
  - The location of an agent is the address of a place. A place resides within an agent system. Therefore, an agent location should contain the name of the agent system where the agent resides and a place name. Note that if the location does not contain a place name, the destination agent system chooses a default place.
- **Agent System**
  - An agent system is a platform that can **create, interpret, execute, transfer and terminate agents.** Like an agent, an agent system is associated with an authority that identifies the person or organization for whom the agent system acts. For example, an agent system with authority Bob implements Bob's security policies in protecting Bob's resources. An agent system is uniquely identified by its name and address. A host can contain **one or more agent systems.**

5

Mobile Agent System



6

FIPA

In parallel to the OMG specification.
More active!

- **Agents**
- **Agent Platform (AP)**
- **Directory Facilitator (DF)**
- **Agent Management System (AMS)**
- **Agent Communication Channel (ACC)**
- **Agent Communication Language (ACL)**

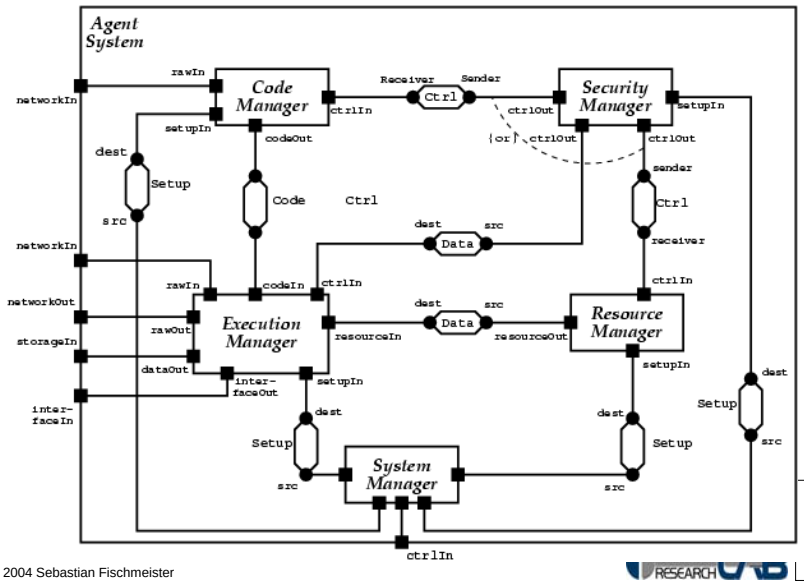
7

Mobile Agent System Building Blocks

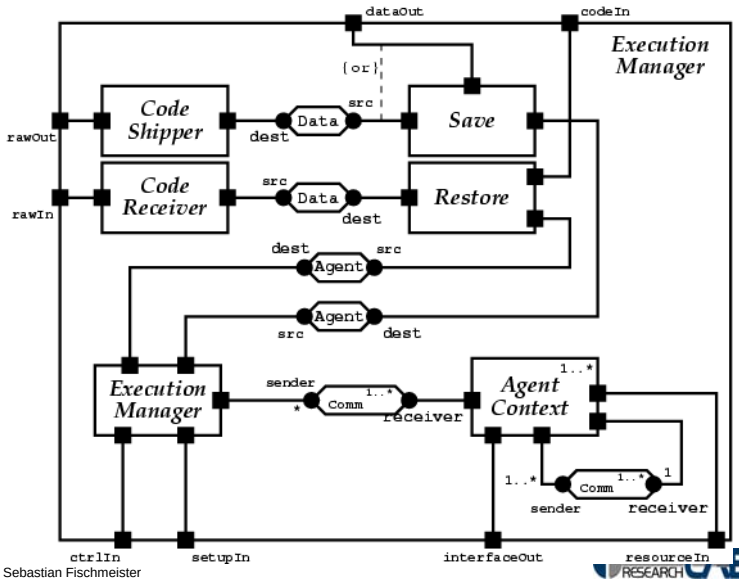
- Example architecture comprising:
  - Code Manager
  - Security Manager
  - Execution Manager
  - Resource Manager
  - System Manager
- Architecture description language
  - functionality driven
  - in/out ports
  - protocols

8

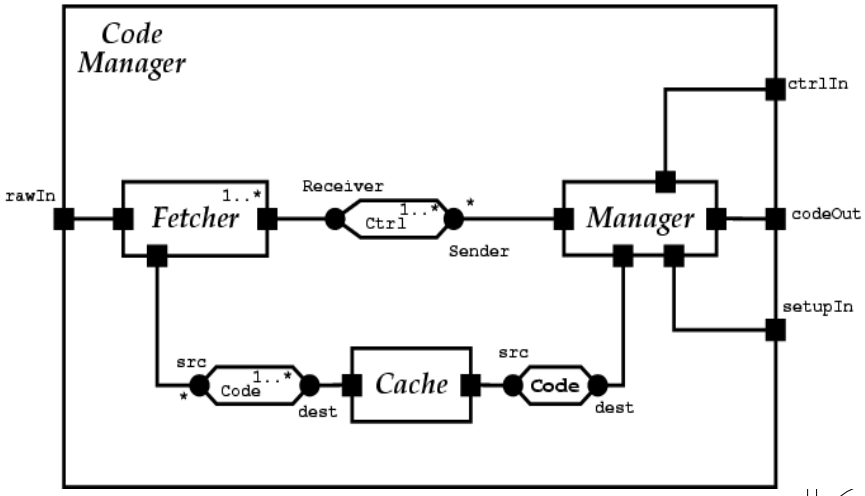
Architecture



Execution Manager

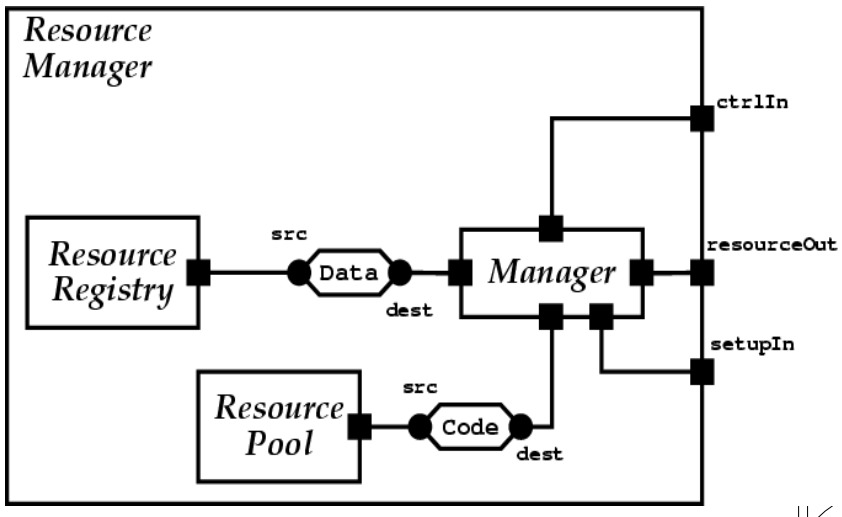


Code Manager



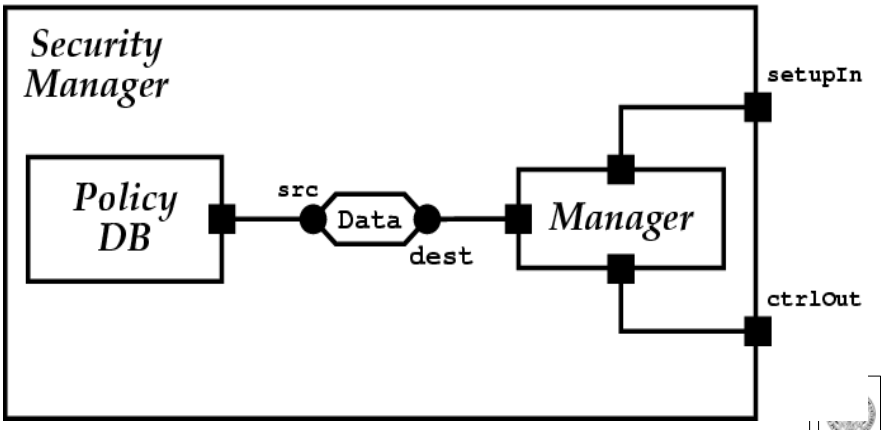
11 © 2004 Sebastian Fischmeister

Resource Manager



12 © 2004 Sebastian Fischmeister

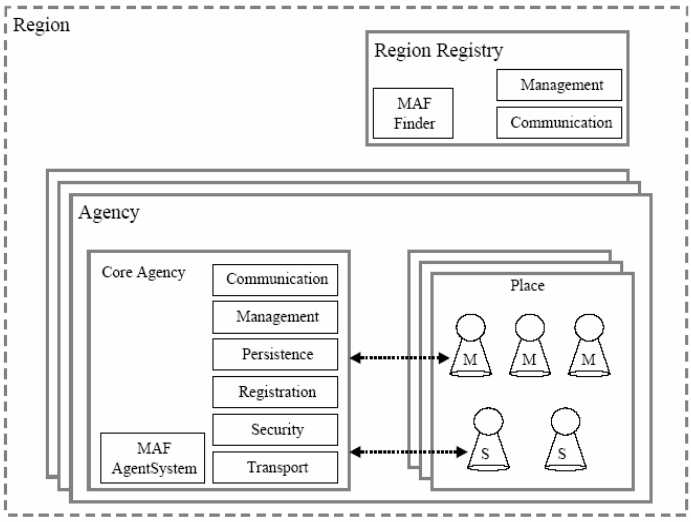
Security Manager



13

Grasshopper Distributed Agent Environment

Overview



15 © 2004 Sebastian Fischmeister

Basics

- This only covers additional information.
- Agent states
  - **active:** the agent is up and executing a task
  - **suspended:** the agent is currently not active, it stays suspended until it is resumed
  - **flushed:** the agent is stored on some sort of permanent memory; in contrast to a suspended agent, it is reactivated automatically on communication

16 © 2004 Sebastian Fischmeister

Basics

- Core Agency
  - represents the minimal functionality to execute agents
  - communication service
    - responsible for all remote interactions (e.g., proxies)
    - manages communication
  - registration service
    - manages a register of all local agents
    - required for finding other agents and communication
    - interacts with the region registry
  - management service
    - responsible for monitoring and controlling agents
    - create, remove, suspend and reactivate places
    - provide information about places (agents, agent infos, all places)
  - transport service
    - responsible for transporting agents from one place to another
    - serialize → communicate → de-serialize → continue

17

Basics

- Core agency
  - security service
    - authorization, authentication
    - external security (between agencies, external agents, region registry)
    - internal security (between local agents)
  - persistence service
    - enables storing of places and agents
    - implicit persistency (automatically, e.g., at shutdown)
    - explicit persistency
      - automatically after timeouts
      - on behalf of the agent or the user (administrator)

18

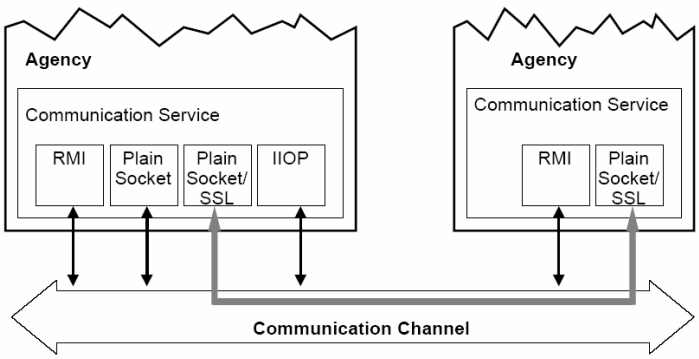
Basics

- Places
  - as described in the standard
- Regions
  - collection of mobile agent systems
  - facilitates the management
- Region registry
  - maintains information about all components associated with a specific region
    - agencies, places, agents, ...
  - useful as lookup service
- You can run Grasshopper without a region !!

19

Communication

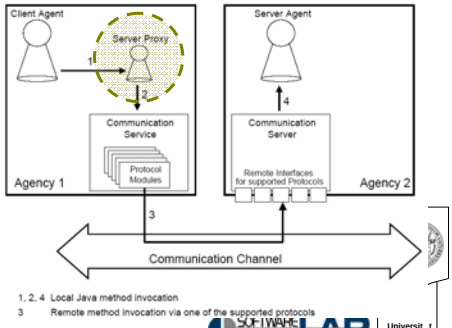
- Multi-protocol support
  - CORBA IIOP, MAF IIOP, RMI, RMI+SSL, plain sockets, plain sockets+SSL



20

Communication

- Location transparency
  - communication across places
  - requires a region registry
  - agent communicates with a proxy (stub)



Communication

- Several types of communication are available
  - synchronous
    - **blocking:** the client waits until it receives the results
  - asynchronous
    - **polling:** on a regular basis the client asks whether the answer arrived
    - **notification:** the client is notified as soon as the answer arrives
  - dynamic
    - if the communication method it not know at compile time, the agent can create a method signature at runtime
  - multicast
    - „and“ termination, „or“ terminiation, incremental termination (see concepts guide)

Grasshopper URL

- URL
  - universal resource locator
  - used to identify objects (places, agencies)

```
<protocol>://<host>:<port>/<agency>/<place>
```

- protocol: **acronym of the desired protocol**
- host: **name or IP address of the destination host**
- port: **number of the port at the destination site (optional)**
- agency: **name of the destination agency**
- place: **name of the destination place (optional)**

Mobile Agent by Example

```
public class BoomerangAgent extends
de.ikv.grasshopper.agent.MobileAgent {
    // A little data state.
    int state;

    public void live() {
        String location;

        switch(state) {

              // code the states here

        }
         log("Terminating my life.");
    }
}
```

BoomerangAgent (State 0)

```
case 0:
    log("Waiting for new location...");
    location = JOptionPane.showInputDialog(null,
                                           "Where shall I go?");
    if (location != null) {
         state = 1;
         log("Trying to move...");
         try {
             // Go away!
             move(new GrasshopperAddress(location));
         }
         catch (Exception e) {
             log("Migration failed: ", e);
         }
         // The next statement is only reached
         // if the migration failed!!!
         state = 0;
    }
    break;
```


BoomerangAgent (State 1)

```
case 1:
            log("Arrived at destination!");
            JOptionPane.showMessageDialog(null, "Let me go home!");
            state = 0;
            log("Trying to move...");
            try {
                 // Come home!
                 move(getInfo().getHome());
            }
            catch (Exception e) {
                 log("Return trip failed: ", e);
            }
            // The next statement is only reached
            //if the migration failed!!!
            break;
```

- Let‘s go to the lab !

27