Multi Agent Systems: An Agent Platform Perspective

Frances Brazier, Martijn Warnier, Michel Oey & Kassidy Clark

Autonomic Systems group
Department of Technology, Policy and Management
Delft University of Technology
Agents

Metaphor/paradigm
• high level agent behavior spec, BDI

Agent technology
• message based interaction between autonomous processes

Simulation
• agents as individual autonomous entities

(source: Luck etal, AgentLink Roadmap)
example ... energy management

Mobach, 2006
example ... logistics
example ... traffic management in Dublin

Ivana Dusparic And Vinny Cahill, 2008
example ... distributed sources of information .... the Courts of Amsterdam

Brazier, Oskamp, Warnier, 2008
example ... supply chain management

TAC/SCM Scenario

Agent
- automated
- optimizing

Suppliers
- strict MTO
- variable supply and prices

Manufacturers
- limited capacity
- competition for uncertain supplies and orders

Customers
- different levels and variability of demand
- "hard" due dates

Offer
RFQs & orders
Production schedule
Delivery schedule

http://www.sics.se/tac

TU Delft
The Future: Multi-agent Data Centers?

- H2O Ice
- Pumps
- Chiller
- Computer Room Air Conditioners
- Central UPS
- Power Distribution Units
- DC Power
- Low Voltage 600VAC Eqpt
- Medium Voltage >600VAC Eqpt
- Parallel or Transfer Eqpt
- Generator
- CHP Fuel Cell, MicroTurbine or Turbine
- Power
- Storage
- In-row Power and Cooling
- Raised Floor
- Coordinator

Utility
- Utility, Rates, Incentives
- Substation
- Communicating Revenue Meter
- Utility
- Generator

Kephart, 2007
Data Center as a Multi-Agent System

Power Manager
- I want to increase the fan speed on that CRAC unit
- I want to turn that server off to save power

Performance Mgr
- I want to process workload on that server

Availability Manager
- I want to use that server to achieve 3-fold redundancy
- I want to leave that server in its present power state to reduce thermal stress

Reliability Manager
- I want to perform valuable computational services

Kephart, 2007
NS: Train Driver Rescheduling using Task Exchange Teams

Mobach et al, 2009
NS: Dynamic Crew Rescheduling: Task Exchange

Resolving conflicts by exchanging tasks between agents
other examples ...

• resource/load balancing

• version management
• radar coordination
• health care information management
• traffic control management

• swarm applications – simulations
Agent Applications:
autonomic systems related

- energy management
- power management
- auctions
- call center
- e-book stores
- electronic dossier
- crisis management
Agents from an AI perspective

• autonomy:
  operate independent and have some control over their actions and internal state

• social ability:
  interact via some agent communication language

• reactivity:
  perceive their environment and respond to changes

• pro-activeness:
  exhibit goal-directed behaviour by taking initiative
Agent Specification Languages – declarative

• BDI languages
  • 2APL, MetateM, dMars, AgentSpeak, JASON

• Other specification languages
  • Agent UML, Agent OWL, Desire

• Environment description languages
  • PDDL, ELM, OWL/WSDL, XML
Agents from a CS perspective

- autonomy: autonomous process, managing its own operation
- social ability: communication primitives (message passing)
- reactivity: reacts to input (from other agents, services and systems)
- pro-activeness: initiates its own actions
Entities in Multi Agent System Paradigm

- Agents
  - autonomous, pro-active, (mobile)
- Objects
  - passive, distributed
- Services
  - (re)active, distributed

Recently, web service model seems to encompass both object and service model in distributed systems
Mobile Agents

Agent (process) is composed of:
- code
- data
- execution state

Strong mobility:
- all components are migrated

Weak mobility:
- discards execution state across migration
Why Mobility of Agents?

Efficiency
- computation near data

Security
- intellectual property, containment

Reliability
- offline devices

New types of interaction
- agent cooperation and coordination
Various agent platforms exist:
- Cougaar
- Ajanta
- JADE, FIPA-OS
- SeMoA
- ... and AgentScape

Many Java-based
Agent Platforms

Agent platforms support:

- development (programming languages, agent model)
- management (life-cycle, resource access, ...)
- communication (message passing, ACL)
- mobility (weak vs. strong migration)
- security (host vs. agent, SSL, PKI)
- service access (look-up services, anonymity services,..)

and provide an execution environment
Agent Platform Standards

FIPA:
- applications
- abstract architecture
- agent communication
- agent management
- agent message transport

MASIF (OMG):
- management
  - system information
  - agent information
- mobility
- system oriented
Agent Platform Characteristics

Agent platforms can also be characterized by:

- agent model (with/without modelling tool)
- agent programming languages they support
- compliance to standards (FIPA and/or MASIF)
- mobility: strong or weak migration
- security
- scalability

and

- execution environment
Multi-Agent System Execution

Multi-agent system execution:

- **concurrent** execution of agents
  - multiple agents executing in parallel on the same host
  - synchronous communication

- **distributed** execution of agents
  - agents distributed over *multiple physical hosts* (e.g. workstations, PC, notebook, PDA, ...)
  - agent migration between hosts
  - asynchronous communication
AgentScape agent platform

AgentScape framework
- "umbrella" research project
- coordinated effort to realize scalable, secure agent infrastructure, services, and applications
- Artificial Intelligence defines application and service level requirements
- Computer Systems provide solutions for service and infrastructure development
AgentScape Infrastructure: challenges

Large scale multi-agent systems

– large number of agents
  • different groups of collaborative agents
  • different types of agents (task, language, …)

– large number of resources
  • wide-area distributed systems (interconnection bandwidth and latency)
  • heterogeneous resources (hardware architecture, operating system, …)
Design Goals & Requirements

Scalability:
- large number of agents
- large number of hosts
- wide-area distributed system

Security:
- protect agents
- protect hosts

Interoperability
- Heterogeneity
Design Goals (continued)

Efficient support for agents
- creation and deletion
- communication
- mobility
- service access
- identity management

Effective management of resources
- performance/security/fault-tolerance
- name/location/directory services
- heterogeneity (and location awareness)
AgentScape: an agent’s perspective
AgentScape: an agent’s perspective

Locations differ in the types of agents they host, the access they provide to web services and resources.
AgentScape: hides the OS
Middleware Architecture
Requirements

Minimal but sufficient agent platform
- deal with autonomy of agent
- asynchronous interactions & mobility
- security policies & mechanisms
- management of resources & services
- open & extensible architecture
AgentScape middleware architecture

Including a number of middleware services: services implement and enforce policies
AgentScape Operating System

Minimal support for agent platforms
- **agent management + mobility**
  agent container: code, data, meta-data

- **communication**
  connection-oriented communication

- **security**
  signed agent containers
  SSL communication channels (use of self-certifying identifiers)
AgentScape Middleware Services

Asynchronous interaction between services
Minimal services per location
  - agent server
  - host manager
  - location manager
  - look-up service
Additional services
  - web service gateway (SOAP/WSDL)
  - fault tolerance service: active/passive agent replication
  - anonymity service
  - configuration services
Distributed Management

Management in AgentScape is distributed

Access control and usage based on:
  - negotiation
  - agreements (WS-Agreement Specification)

Leases are issued and jointly managed by location and host managers according to pre-defined policies
AgentScape Tutorial

Practical Assignment
Objectives

- Hands-on experience with AgentScape, an Agent Middleware Platform
- An introduction to programming agents
- An example of agents in autonomic computing
AgentScape: a simple overview

Distributed platform for running agents
- Locations
- Agents
- Migration
Start AgentScape: console.jar

Type: java -jar lib/console.jar
or
Click ``Run console'' icon
Start a Location

1. Select

2. Click Start Location

3. Type name
Start an Agent

1. Select

2. Click Start Agent

3. Browse for agent

4. Select fishtank.jar
FishTank and Fish

- Start another Agent: Fish
- Start another Location and another FishTank
Assignment: context

- Multiple agents, each with multiple goals
- Specific resources are required to perform a task (DB-access, Web-access, etc)
- Resources are provided on Locations
- Agents migrate between Locations to execute their tasks
- Agents need to communicate with each other for efficiency and coping with failing/changing resource availability
**Assignment:** WorkerAgent

WorkerAgent – abstract class, extends general Agent class to hide migration/communication details

<table>
<thead>
<tr>
<th>Methods</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doTask( task )</td>
<td>Try to perform task at the current location, return boolean</td>
</tr>
<tr>
<td>hasAgentTasks()</td>
<td>Returns whether this agent still has tasks to perform</td>
</tr>
<tr>
<td>nextAgentTask()</td>
<td>Return the task this agent has to perform next</td>
</tr>
<tr>
<td>getName()</td>
<td>Returns name of this agent</td>
</tr>
<tr>
<td>getLocationTaskList()</td>
<td>Returns a list of tasks that can be performed at the agent’s current location</td>
</tr>
<tr>
<td>getLocations()</td>
<td>Returns a list of available locations</td>
</tr>
<tr>
<td>migrateTo( location )</td>
<td>Migrate to another location, returns boolean</td>
</tr>
<tr>
<td>print( msg )</td>
<td>Prints a (debug) message to the screen</td>
</tr>
<tr>
<td>doWork()</td>
<td>Abstract method, needs to be implement for assignment</td>
</tr>
</tbody>
</table>
Assignment: ExampleWorkerAgent

ExampleWorkerAgent – naive implementation
No communication, No migration => Cannot complete tasks

```java
while ( hasAgentTasks() ) {
    String task = nextAgentTask();
    if ( doTask( task ) ) {
        print( "Performed task: " + task );
    } else {
        print( "Cannot perform task: " + task );
        return;
    }
}

assignment: improve this agent
```
Assignment: Starting your agent

• Compile your agent
  • ant jar

• Start lib/tutorial.jar
  • Double-click it
  • ant tutorial
  • java -jar lib/tutorial.jar

• Or: From within Netbeans IDE
Assignment: Simulation manager

Start a copy of the agent on all locations
**Assignment:** Running simulation
Assignment: Running simulation

Displays summary of all locations and progress of the running agents
http://www.agentscape.org

Thank you for your attention!