Behavior Model of Mobile Agent Systems

Authors: Pr. Serge Chaumette
         Dr. Pierre Vignéras

Team: Distributed Systems and Objects
Laboratory: LaBRI – Bordeaux, France
Head: Pr. Serge Chaumette

The 2005 International Conference on Foundations of Computer Science

Monte Carlo Resort, Las Vegas, Nevada, USA
June 27-30, 2005
Outline

I. Mobile Code Paradigms
II. Mobile Agents
III. Active Containers
IV. Conclusion
I. Mobile Code Paradigms
# Mobile Code Paradigms (1/5)

[Fugetta et al. 1998]

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client/Server</strong></td>
<td>Site A</td>
<td>Site B</td>
</tr>
<tr>
<td>non mobile (HTTP, X)</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td><strong>Remote Evaluation</strong></td>
<td>code A</td>
<td>ressource B</td>
</tr>
<tr>
<td>(rsh, PostScript)</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td><strong>Code on Demand</strong></td>
<td>ressource A</td>
<td>code A</td>
</tr>
<tr>
<td>(Java Applet, Flash)</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td><strong>Mobile Agent</strong></td>
<td>code A</td>
<td>ressource A</td>
</tr>
</tbody>
</table>

A,B: execution entities (thread, process)

code: move
c: executor
## Mobile Code Paradigms (2/5)

[Fugetta et al. 1998]  

<table>
<thead>
<tr>
<th>Mobile Agent</th>
<th>Code on Demand (Java Applet, Flash)</th>
<th>Remote Evaluation (rsh, PostScript)</th>
<th>Client/Server non mobile (HTTP, X)</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A</td>
<td>Code: move resource A</td>
<td>resource B</td>
<td>A</td>
<td>Site A</td>
</tr>
<tr>
<td>Site B</td>
<td>Code: move resource B</td>
<td>resource B</td>
<td>A</td>
<td>Site B</td>
</tr>
</tbody>
</table>

I. Mobile Code Paradigms

Before

<table>
<thead>
<tr>
<th>Code on Demand (Java Applet, Flash)</th>
<th>Remote Evaluation (rsh, PostScript)</th>
<th>Client/Server non mobile (HTTP, X)</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A</td>
<td>resource A</td>
<td>resource B</td>
<td>A</td>
</tr>
<tr>
<td>Site B</td>
<td>resource B</td>
<td>resource B</td>
<td>A</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>Code on Demand (Java Applet, Flash)</th>
<th>Remote Evaluation (rsh, PostScript)</th>
<th>Client/Server non mobile (HTTP, X)</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site A</td>
<td>resource A</td>
<td>resource A</td>
<td>A</td>
</tr>
<tr>
<td>Site B</td>
<td>resource B</td>
<td>resource B</td>
<td>A</td>
</tr>
</tbody>
</table>

A, B: execution entities (thread, process)  
C: executor
# Mobile Code Paradigms (3/5)

[Fugetta et al. 1998]

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client/Server, non mobile (HTTP, X)</td>
<td>Site A</td>
<td>Site A</td>
</tr>
<tr>
<td>Remote Evaluation (rsh, PostScript)</td>
<td>code A</td>
<td>code B</td>
</tr>
<tr>
<td>Code on Demand (Java Applet, Flash)</td>
<td>ressource A</td>
<td>code B</td>
</tr>
<tr>
<td>Mobile Agent</td>
<td>code A</td>
<td>ressource A</td>
</tr>
</tbody>
</table>

**A, B:** execution entities (*thread, process*)

**code:** move

**ressource:** executor
## Mobile Code Paradigms (4/5)

[Fugetta et al. 1998]

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client/Server non mobile (HTTP, X)</td>
<td>Site A: code A, Site B: B</td>
<td>Site A: code A, Site B: B</td>
</tr>
<tr>
<td>Remote Evaluation (rsh, PostScript)</td>
<td>Site A: code A, Site B: B</td>
<td>Site A: code A, Site B: B</td>
</tr>
<tr>
<td>Code on Demand (Java Applet, Flash)</td>
<td>Site A: ressource A, Site B: code B</td>
<td>Site A: code A, Site B: B</td>
</tr>
<tr>
<td>Mobile Agent</td>
<td>Site A: code A, Site B: B</td>
<td>Site A: code A, Site B: B</td>
</tr>
</tbody>
</table>

A, B: execution entities (thread, process)

A: executor

Before

After

I. Mobile Code Paradigms
## Mobile Code Paradigms (5/5)

[Fugetta et al. 1998]

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client/Server non mobile (HTTP, X)</td>
<td>Site A</td>
<td>Site B</td>
</tr>
<tr>
<td>Remote Evaluation (rsh, PostScript)</td>
<td>Site A</td>
<td>Site B</td>
</tr>
<tr>
<td>Code on Demand (Java Applet, Flash)</td>
<td>Site A</td>
<td>Site B</td>
</tr>
<tr>
<td>Mobile Agent</td>
<td>Site A</td>
<td>Site B</td>
</tr>
</tbody>
</table>

A, B: execution entities (thread, process)

C: executor
II. Mobile Agents
Mobile Agents: Definition

• An agent is a program which is both autonomous and independant
  – autonomous: it controls its own execution
  – independant: it holds its own thread
• An agent is mobile if it can move from one execution environment (server) to another.
  – strong migration: resumes at the next instruction (requires saving the instruction pointer)
  – weak migration: resumes using an event mechanism
Mobile Agent Frameworks

- Precursor in 1996: *Telescript* -- General Magic
  - Special language
- Nexts are essentially Java-based frameworks
  - Odyssey (General Magic, successor of Telescript), Voyager (ObjectSpace now Recursion Software), Concordia (Mitsubishi), Aglets (IBM), ...
- Interoperability
  - Standard MASIF (OMG): implemented by Grasshopper
Modelisation
\((\pi\text{-calculus})\)

Mobile Agent System Model (Aglets-inspired)

Mobile Agent System Model (Aglets-inspired)

Active Container Model

Security problems underscored (system resources access by agents)

• Other calculus may be used for the proof
  ➢ Ambients, join-calculus (Chemical Abstract Machine)
• Proof tools (HOL, Pict)

Open Question!

equivalence (bisimulation)?
III. Active Containers
Active Container: a Base Mechanism for Code Migration

**Container**: put(), get(), remove()  
**Activity** in the container: call()  
asynchronous call: **parallelism**

When Client == Object, we have an agent!

These four primitives do exist in almost all systems, but they are not publicly available

→ JACOb: Java implementation

Dynamism
Use Cases of the Active Container Model
[chaumette, vignéras PARCO'2003]

● Memory Model
  put(): create a stored object (new operator)
  remove(): remove a stored object (garbage collector)
  get(): retrieve a copy of a stored object (clone())
  call(): invoke a method of a stored object (o.m())

● Application Deployment
  − Deployment of components of a parallel application
  − Software network installer

● Plugins for applications

● Asynchronous Remote Method Invocation
JACOb: a Java implementation of active containers (1/2)

```java
public interface ActiveMap extends java.util.Map {
    void call(Object key,
               java.lang.reflect.Method method,
               Object[] args,
               Future future);
}
```

- Defines the active container
- Familiar to developers
- Use of reflection (*dynamism*)
- Asynchronous Method: early reply
JACOb: a Java implementation of active containers (2/2)

- Local implementation
  - Business (container) and remote layer separation
- Remote implementation
  - Global, event-based remote exception handling
    - still allows the traditionnal (by invocation) RMI exception handling
  - Multi-protocols
    - RMI, TCP and UDP available
    - No limit (Myrinet, SOAP or JToe possible)
Conclusion
Mobile Agents: a Good Idea?

- In [Harrison and al. 1995] (IBM)
  - Cons: it is always possible to write an equivalent application without the use of the mobile agent paradigm (no "killer application"!)
  - Pros: this paradigm allows the development of many applications that would require many distinct paradigms
    - Ease of design, development and deployment
    - Used for system and network management (SNM) [Reuter and Baude 2002]
- IBM starts the design of the Aglets system
Mobile Agents: reasons of a "failure" (1/2)

- Technical reasons
  - Internal security problems
    - protection of an agent from its server
    - maximal risk in case of bugs for a server (*worms*)
      - disk, memory, CPU pollution
      - ease of infection due to the *mobile* aspect
  - Interoperability
    - migration type (strong/weak, proactive/reactive)
    - language used
    - security management, ...
- Too many choices, too many risks
Mobile Agents: reasons of a "failure" (2/2)

• Non-technical Reasons
  – Modes of Remuneration
    • Enterprises: advertisement-based *business model*
      – a mobile agent is not a user, it is not sensible to adverts!
    • Individuals: no *killer* application
  – Quality of Service
    • Service provider distinct from the agent provider
      – perception of the service by the user depends on the agent
        (common case on the web: display depends on navigators)
  – Unknown Paradigm in the business world
Facts: from Mobile Agents to Asynchronous Execution Frameworks

Mobile agent list
- Last update: 01/2004
- 71 mobile agent systems
- 54 % links are invalids today
- Most standards are no more supported (Concordia, Odyssey, AgentOS, ...)

Growing success of distributed frameworks based on communicating (possibly asynchronous) components (possibly mobile)
Facts: from Mobile Agents to Asynchronous Execution Frameworks

Mobile agent list
- Last update: 01/2004
- 71 mobile agent systems
- 54% links are invalids today
- Most standards are no more supported (Concordia, Odyssey, AgentOS, ...)

Growing success of distributed frameworks based on communicating (eventually asynchronously) components (eventually mobile)

The Mandala Project
http://mandala.sf.net/
Eases the development of concurrent and/or distributed applications