IBM Agents

Mobile Java™ Agents with Aglets™

Christian Weigel
Department of Computer Science
University of Applied Sciences Kaiserslautern
Amerikastr. 1, 66482 Zweibrücken, Germany
cweigel@gmx.net - http://www.christianweigel.com/

1 Definitions

1.1 What are Software Agents?

There are different views toward Software Agents. One is the End-User, the other one is the System Perspective.

From the End-User Perspective a Software Agent is a program that assists people and acts on their behalf. Agents function by allowing people to delegate work to them[1].

This is rather a basic kind of definition. Software Agents tend to be extremely varying in their occurrence. Software Agents are able to "live" in many different settings like computers, operating systems, databases and they're up to very different tasks.

From the System Perspective, an agent is a Software Object that is situated in an environment[1], possesses (mandatory) properties (reactiveness, autonomy, goal-driven, continuously executing) and may contain some additional other properties like:

- ability to communicate with other agents
- ability to travel from host to host
- adaption because of former experience
- believable appearance to the End-User

Combining these two points of view, we consider Software Agents to be small Software Tools that exist to help people doing a specific kind of work, to simplify given tasks and automate varying assignments. To complete the given tasks Software Agents possess many different abilities – the most important one, the ability to communicate with other Systems and Agents

1.2 What different types of Agents do exist?

Agents can be roughly subdivided into two types. Stationary and Mobile Agents. Agents mostly tend to be mobile, but do not have to.

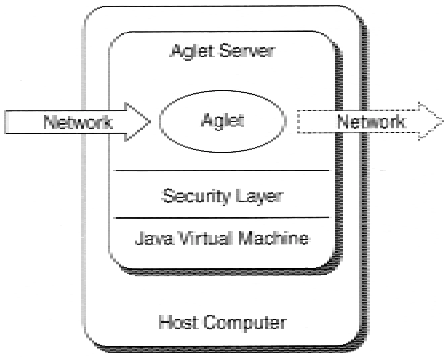
[1] Programming and Deploying Java Mobile Agents with Aglets, B. Lange, Mitsuru Oshima, Addison Wesley 1998

Many existing agents belong to the stationary category. **Stationary Agents** execute *only* in the system space they are created in. If these Agents need information that resides outside their own domain, other Agents or communication mechanisms have to be called by them.

Mobile Agents are not limited to the system space they are created in. Mobile Agents are free to travel among all possible hosts in a network. Travelling from host to host, a Mobile Agent transports its own state and code with it to resume its work in other execution environments of the network.

1.3 What are Aglets in particular?

Aglets are a special type of Agents developed by IBM Research. There is an Aglet application programming Interface (API), and an Aglet Software Development Kit (ASDK) that is freely available on the Internet[2]. Aglets are Java Objects that are able to travel from host to host (Mobile Agent). Aglets are hosted by an Aglet Server that provides the environment for them to run in. The Java Virtual Machine and a special Aglet security manager ensure safety while receiving and hosting Aglets. The word *Aglet* means "lightweight agent" it is composed of the word *agent* and the word *applet* (lightweight application).



2 Mobile Agent Systems

2.1 Properties of Mobile Agents

The Agent model consists of two fundamental parts: the **Agent** itself and its **execution environment** (place). Both will be explained in the following paragraph.

The Agent

A mobile Agent has 5 Attributes:

State	An Agent takes its execution state with it when it travels. In most cases it is important for an Agent to know where it stopped its execution, so the Agent is able to proceed its task (There may be times in which an approximation is sufficient). The State of an Agent is stored in its instance variables. After arriving at a new host, decisions are made depending on former activities.
Implementation	Executing an Agent needs code that has been formerly designed. There are two possible ways code can be executed. One way is that the Agent carries its code with it. Another way would be that the Agent travels to the destination looks for executable code there, and retrieves missing code snippets over the network (code-on-demand).
Interface	Agents have an interface that allows other Agents and Systems to interact with them.
Identifier	Every Agent has a unique identifier, using this identifier; the agent can be identified and located throughout the net.

[2] http://sourceforge.net/projects/aglets/

Principals	There are two main principals for Agents: The *Manufacturer*, e.g. the author of an agent The *Owner*, e.g. the one who has the legal & moral responsibility for the Agent (creator)

Execution Environment (place)

The four important parts of the execution environment are:

Engine	Agents are executed using an engine (Java Virtual machine & Operation System)
Resources	The engine and the place provide controlled access to local resources and services like networks, databases and so on.
Location	The location is a combination of the name of the place where the engine is running and the Agent is executing.
Principals	The Place is (like an Agent) associated with an *authority* and a *manufacturer*.

2.2 Agent behaviour: Creation, Disposal, Transfer, Communication

Creation

An Agent is created in a place. The creation can be initiated by another agent or nonagent system. The creation is done in three steps:

1. Instantiation and identifier assignment – the class definition is loaded and made executable, the agent object is instantiated. The place assigns a unique identifier to the agent.
2. Initialization – The Agent is given a chance to initialize itself. Only after successful initialization the agent is fully and correctly installed in the place.
3. Autonomous execution – After successful installation the agent starts execution, it is now able to decide about its execution process independently.

Disposal

The disposal of an Agent can be initiated by itself, by any other agent in the place, or by another agent or nonagent system outside the place the agent is executing. Other reasons why agents may be disposed are:

- the lifetime of an agent has expired
- no one refers to the agent any longer
- the agent has violated security rules
- the system an agent runs in is shutting down

Disposing of an agent is a two step process:

1. Preparing for disposal – The agent is given a chance to finalize its current task before it is disposed of
2. Suspension of execution – The place suspends the execution of the agent

Transfer

Mobile Agents have the ability to travel. This process can be initiated by the agent itself, another agent, or nonagent system. The agent is being dispatched from its current place (origin) and received by a specified place (destination). The dispatch process is managed by the origin and destination place of the agent; successful transfers and errors are negotiated between them.

Communication

Agents have the ability to communicate with other agents residing in the same place (intraplace) or with agents residing in other places (interplace and interengine). An agent can invoke a method of another agent or send it a message if it is authorized to do so. Agents support peer-to-peer and broadcast messages.

3 Mobile Agents with Java

3.1 Java Agent Characteristics

Java is excellent for designing and programming mobile agents. Java has many built in features that qualify it to be the first choice in agent programming. Cause of the byte-code concept the Java compiler offers, java applications are able to be executed anywhere, on any computer of the network, no matter of the underlying Operating System or Hardware. Java does not support dangerous pointer operations that are able to overwrite memory parts and corrupt data in such ways. The “Sandbox” principle the Java Runtime creates makes it extremely safe to execute code received from the network.

In short, the most positive aspects of using Java are:

- platform independence
- secure execution

Further aspects are:

- dynamic class loading
- multithread programming
- object serialization
- reflection

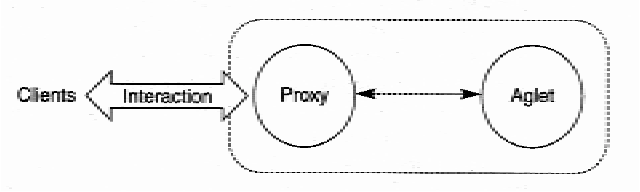
There are several drawbacks to the Java Virtual Machine Concept, these are:

- Inadequate Support for Resource Control (may result into Denial of Service Attacks)
- No Protection of References
- No Object Ownership of References
- No Support for Preservation an Resumption of the Execution State (only internal agent attributes and some external events may be used to reconstruct the execution state of a Mobile Agent)

3.2 Short Introduction to the Aglet Model

The basic elements of the Aglet Model are found in the following abstractions:

- Aglet: a mobile java object that visits aglet-enabled hosts in a computer network. It is autonomous and reactive
- Proxy: representative of an aglet that protects the aglet's public methods like a shield from direct access. A proxy is also able to hide an aglets real location (location transparency)



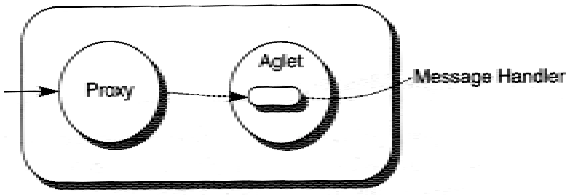
- Context: an aglets workplace. A context provides a uniform environment in a realm of heterogeneous hosts for an aglet to run in.
- Identifier: every aglet has a globally unique identifier bound to itself (unchangeable during its entire lifetime)

4 Examples

4.1 Messaging with Aglets

One of the most important properties of an aglet is, that it has the ability to communicate with other aglets. Aglets support an object based, location independent, extensible, rich and synchronous/asynchronous messaging framework. Here is a demonstration of a simple aglet messaging session.

Interaglet messaging works by implementing specific handlers for the messages the aglet is supposed to understand. Therefore an aglet's proxy serves as a message gateway to the outer world.




```
public object AgletProxy.sendMessage
(Message theMessage)
```

… this sends a message Object to the aglet and waits for the opposite aglet to reply

```
public class SimpleMessageExample extends Aglet
{
      public void run()
      {
         try
         {
           AgletProxy proxy = getAgletContext().createAglet(getCodeBase(),
           "SimpleMessagechild",null);

           try
           {
              proxy.sendMessage(newMessage("Hello"));
           }
           catch (NotHandledException e)
           {
              //Child failed to handle the Message
```

```
            }
         }
         catch (Exception e)
         {
            //Failed to create the child
         }
      }
}
```

… this creates a simple message object of the “Hello” kind and sends it to an aglet - SimpleMessageChild

```
public boolean Aglet.handleMessage(Message message)
```

… handles an incoming message object. The method has to be overridden in order to have the aglet perform a specific task

```
public class SimpleMessageChild extends Aglet
{
      public boolean handleMessage(Message msg)
      {
         if (msg.sameKind(“Hello”))
         {
            doHello();//respond to the ‘hello’ message
            return true;    //SimpleMessageChild handled this message
         }
         else
         {
            return false;   //SimpleMessageChild did not handle this message
         }
      }
}
```

… now the `SimpleMessageChild` receives the Message and if the Message received is “Hello”, the Aglet responds in a formerly defined way. A Handler is supposed to return true or false, whether the Message was understood and handled, or not understood/not handled.

If the Message received by `SimpleMessageChild` is not “Hello”, it will return a false and a `NotHandledException` will occur in the `SimpleMessageExample` aglet. Such Exceptions have to be catched!

5 Conclusion

This introduction to the field of mobile Java™ Agents should have given a small glimpse towards the habits & possibilities of mobile Java™ Agents.

Mobile Agents offer a wide range of possibilities to the user. They are versatile helpers in daily life. Mobile Agents can be customized to satisfy the user's needs regarding personal preferences and backgrounds.

Everybody should be encouraged to participate in the development of new kinds of mobile agents. The ASDK you need to design your own Aglets is freely available via Sourceforge, just follow the link at the end of this paragraph.

Resources used to create this document:

[1] Programming and Deploying Java™ Mobile Agents with Aglets, Danny B. Lange/ Mitsuru Oshima, Second Printing, Addison Wesley 1998

[2] http://sourceforge.net/projects/aglets/ (Download ASDK)

[3] http://www.trl.ibm.com/aglets/index_e.htm (IBM Aglets Homepage)

[4] http://www.www.org/ (THE WEB)