“Agent-Environment for Small Mobile Devices”

Thomas Strang and Melanie Meyer

{firstname.lastname}@dlr.de

German Aerospace Center (DLR), Site Oberpfaffenhofen

Institute of Communications and Navigation (KN-S)

Introduction

There is remarkable growth in the use of mobile devices in which the agent technology is predestined to play a significant role in the realisation of new applications, which will assist their users in a wide variety of ways.

Bringing the agent technology into the mobile device has a lot of advantages, such as providing services to the user known from the World Wide Web in face of drastically changing network conditions in a very personalized way. Typical mobile devices like PDA, cell phone or pagers are constrained through their limited resources, such as processing power and memory, userinterface capabilities and network connectivity. Thus, the agent technologies must anticipate and accommodate disconnected wireless network access and its complications. Furthermore, the agent-environment must be capable of operating on a variety of device platforms independently as well as through fixed networks.

A lot of work has been done on the area of agent systems. Examples are IBM Aglets[6], Mitsubishi Concordia[7] or the JADE/LEAP-Platform[8]. Most of them omit the implications of having resource limited mobile devices as hosting platform for agents.

Usage Scenarios

In order to demonstrate the possible fields of application one could imagine the following scenarios.

Scenario A: The system on the mobile device runs an agent which monitores the environment for Bluetooth equipped fax machines in the vicinity. When the user receives a fax to his mailbox, the agent provides the mailbox handler with the phonenumber of the fax machine nearby, to which the mailbox handler offers the user to forward the fax.

Scenario B: A user is interested in information regarding a movie in a cinema near to his current location. He starts a related agent on his mobile device and inputs the movie name and preferred starting time. Rather than having to "surf" or search for the information himself, s/he may simply start the search agent and forget it. The agent is able to autonomously acquire the information required about the user’s location and search for the nearest cinema. If that information is not available locally, the agent moves to a platform residing in the fixed network next time the device is connected using any carrier suitable (e.g. established Bluetooth or GRPS link). There the agent collects and evaluates relevant information, and moves back to the mobile device when connected again, carrying the results of its task. Furthermore, it can even suggest and acquire the tickets. Beside user initiated agents, system initiated agents are very usefull. For example, background searches for information, which may be of the user’s interest, based on his personal statistical profile, or completing application forms, thus saving the user the time and effort, are adjuvant easements when using mobile devices.

Principles in Agent Technology

There are a number of differing definitions for agents. One describes an agent as “computer program that acts autonomously in the interest of the user and helps to perform some task” [1, 2]. Yet it is the properties of the Agent that help us to classify them. The basic characteristics of an agent are reactivity, autonomy and communication. Additional properties are mobility, cooperation and, to some degree, intelligence. Any agent “lives” in an environment (often called platform), which provides a set of components to support the agent’s work. It provides services and allows communication between agent-agent, agent-platform, agent-user, platform-platform and platform-user.

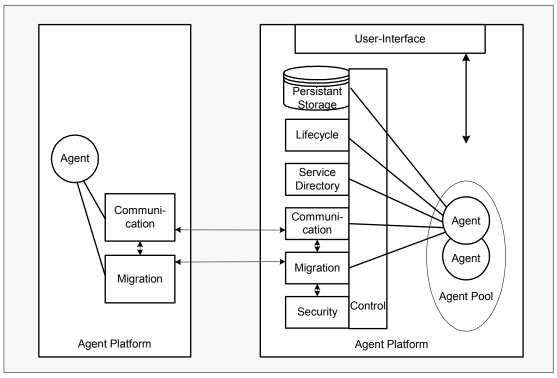

Figure 1: Overview of Components of an Agent-Environment

An overview of typical components of an agent-environment is provided in figure 1. To achieve a flexible architecture for usage on a variety of devices and to keep the restrictions by underlying hard- and middleware, these components have to be well designed with respect to the constraints given by the mobile devices. Often an adaptation from the general functionality to the abilities of the device or the operating system on the device is required.

An example for this adaptation is the persistent storage: Some mobile devices like PDAs have a (limited) file system, whereas others have a record based storage system instead. If an agent wants to store some data on the device (eg. an intermediary result), the platform has to provide an interface to the device-dependent storage facilities, even if both systems provide the same abstraction level for programming (eg. a Java Virtual Machine).

Stationary Agents on Mobile Devices

A good example for an useful stationary agent is given in scenario A – it is an agent based context sensor. The agent is sensing some environmental condition like coming into the receiption area of a bluetooth station in the vicinity. This may be done actively by periodically polling an accessible information source like the bluetooth driver of the mobile device, or passively by registering for a specific event at a responsible event generator.

How the agent aquires the state change, and how it should react to it, is part of the specialized behaviour of an agent, and thus encapsulated and hidden to any module using this information.

To explain why stationary agents are suitable especially for context sensing on mobile devices, we want to make a short excurse.

Excurse: Context-aware Services

We rely here on the term *context* as defined in [10], which is “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant [..]”. Context information is obtained by observing relevant entities with context sensors. Each context sensor is responsible for acquiring a certain type of context information while encapsulating how the information is actually sensed. The output of a context sensor (software or hardware) is a *marking* on a *symptom axis*, which may be single, range or group of values of a well-defined taxonomy or identifier system.

Each symptom axis represents the state of a specific context type against user- and system-services, making them *context aware* following a definition given in [10]. All symptom axes served by the available context sensors on a device span a *situation space*, where a specific situation is represented as a snapshot of the current markings on each axis. Services may react to changes of the current situation (e.g. onEnterSituation, onLeaveSituation,…), or pro-actively influence the environment due to some situation.

As one can see, within the above definition a context sensor is responsible for a very specific task (sensing one type of context information). It performs its action independently of any other context sensor or service by reacting to changes in its environment. The collaboration of all context sensors enables a dynamic and comprehensive mapping of the current real-world situation to the device-internal representation (as far as context sensors are available). Those parallels to the key characteristics of an agent (small specialized task, reactivity, autonomy, collaboration) clarify the motivation of using an agent system as part of a mobile service environment for sensing various types of context information.

Stationary agents on mobile devices can be seen as individual, dynamic information sources for other components of the agent platform, or other agents – stationary or mobile – as well.

Mobile Agents on Mobile Devices

Agents work with small, specialized tasks and are able to coordinate their work depending on their interpretation of the environment. Intelligent agents will enhance the functionality of applications and facilitate the users effort. In order to utilise resources more efficiently, a *mobile* agent can suspend the work and move it to a platform in a fixed network (migration), thus outsourcing resource-expensive work, and eliminating the need for a persistent connection. Network limitations and disconnections are solved by the autonomous work, the agent returns with a result when the connection is restored (see scenario B).

In that context it is interesting to have a look at the different states an agent can reach during his lifecycle (see figure 2), which is similar but not equal to the FIPA specification [9]. An agent is instantiated on its home platform having the *initiated* state after setup with some initial values (eg. input parameter, max. time before entering a checkpoint etc.) for that instance. When this agent instance is started by the platform's scheduler, the agent is as long in the *active* state as the agent (active suspend) or the platform (passive suspend) decide to enter a checkpoint, leading to the *suspended* state. If some termination criteria has been given and reached during the agent's work, the agent enters the *ready* state. Each time a *mobile* agent is in the suspended state, a decision can be made wether the agent migrates to another platform or remains on the same platform. If a migration is performed, the agent instance enters the *transit* state on the source platform and the suspended or ready state on the destination platform.

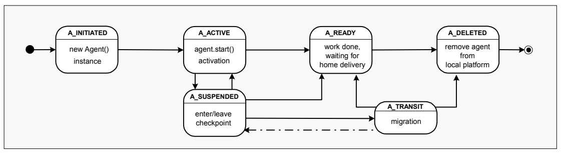

Figure 2: Agent LifeCycle

As long as the agent is in the suspended or ready state, its (intermediary) results may be accessed by the user through some platform functionality, until the agent instance is finally deleted.

Using Local Computing Power

Well-known agent systems like JADE/LEAP [8] make use of mobile devices in the sense of "thin clients". Using the local processing power offered by upcoming devices not only for displaying an interface to the user, but also performing part or all of the processing of the task itself does make sense under certain circumstances as well. This is a very interesting option if wireless access to any server infrastructure is currently not possible (e.g. no bearer available), not allowed (e.g. when using a mobile phone in an aeroplane), or to expensive for the value of received information, resulting in a (typically limited) local processing fallback feature.

Using the local computing power means in the sense of an agent system the ability to instantiate, run and control agents on the mobile device itself, making the mobile device to a platform for stationary or mobile agents equivalent to platforms of the same agent system in the fixed network. But this equivalence has its constraints in the ressource limits of the the mobile device.

The maximum amount of agent instances running on the mobile device at a time, as well as the performance of each agent itself, depends on the characteristics of the mobile device, and is usually significantly worse than if running the same agent on a platform in the fixed network.

The performance characteristics of an agent are mainly influenced by

- the internal implementation of the agent's task (algorithm)
- the external performance of the platform itself

If the platform knows about the performance characteristics of an agent, this information can be used optimize the agent's handling. As one can see in figure 3, knowing about the break even between local processing and remote processing of an agent allows for instance to estimate the best time for running into the next checkpoint before migration starts.

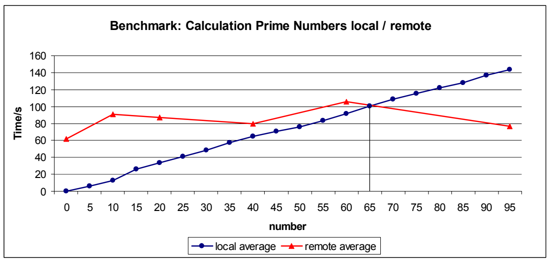

Figure 3: Local and Remote Execution

For instance it would not be senseful to run the mobile agent in this example locally with an inter-checkpoint-width of more than 66 algorithm steps, because it would be faster to migrate, compute remote and migrate back in that case.

Sample Implementation in J2ME

For testing purposes we implemented an agent platform called *Mobile Device Agent-Environment (MDA)* based on the Java2 Micro Edition (J2ME), more precisely on the very low end of Java programming, the Mobile Information Device Profile (MIDP) on top of the Connected Limited Device Configuration (CLDC). Those J2ME libraries consist of a small subset of J2SE, extended by some libraries for the specific user interface (ITU-T onehand-keyboard or touchscreen) and I/O facilities (HTTP as the one and only network protocol), targeting ressource limited devices like PDAs or mobile phones.

One of the remarkable implications of this Java version has came up in the area of agent mobility. Due to the "closed" late binding specified as part of the security concept of the CLDC, any agent implementation (classfile) intended to be used at runtime must be present at installtime of the platform itself. Thus the Migrator, responsible for serialization and deserialization of agents, as well as sending and receiving them to other platforms of the agent system, can only implement the *weak migration*, which is explained as "if the program has to prepare its migration by explicitly storing its state in some variables and is started again at the new location, and if the programmer has to provide explicit code to read and re-establish the stored state" [5].

Closed late binding is a strong restriction not only for mobile agents, but for stationary agents as well. For instance for using stationary agents as context sensors (see scenario A) it is required to package any required type of context sensor at installtime. Adding a new context sensor or replacing a context sensor by a newer version requires to re-package and re-install the whole package (containing all agents, the agent environment and any service environment using the agent environment) on the mobile device.

On the other side having small uniform exchangeable modules like agents for context sensing and processing is also an advantage. For gaining access to relevant sensor values (e.g. GSM cell id) it is often necessary to use vendor-specific functions, bypassing the Java VM. Moreover, "relevance" relys on a very personal interpretation, and thus the "relevant entities" observed by context sensors vary from user to user. Building a situation handler on top of an agent system enables much flexibility to provide context aware services on personal mobile devices.

Our implementation showed that it is possible to run multiple stationary and mobile agents on a CLDC/MIDP equipped ressource limited mobile device at the same time, even if the performance of todays devices is not very good. It showed also, that the agent technology does make sense for specific tasks only, where for instance autonomy and reactivity of a task performed in the background without user interaction is more important than performance.

Conclusion and Outlook

For specific tasks it is very usefull to have an agent platform on the mobile device itself. It has been illustrated why especially context sensing and processing can be done on the mobile device very smart using the agent technology. With the advantage of intelligent mobile software the mobile agent technology enables the development of new applications against the background of location and context awareness and will support the user with distributed information retrieval and global services.

References

[1] A. Lingnau, O. Drobnik, P. Dömel: A HTTP-based Infrastructure for Mobile Agents, 1995, http://www.w3.org/Conferences/WWW4/Papers/150/

[2] S. Franklin, A. Grasser: Is it an Agent or just a Program?: A Taxonomy for Autonomous Agents, Institute for Intelligent Systems University of Memphis, 1996, http:// www.msci.memphis.edu/~franklin/AgentProg.html

[3] Mobile Information Device Profile Specification Final, Sun Microsystems, Inc., 2000, http://java.sun.com/j2me/docs/

[4] http://kxml.enhydra.org/software

[5] S. Fünfrocken: Transparent Migration of Java-based Mobile Agents, Technische Universität Darmstadt, Fachgebiet Verteilte Systeme, 1998, http://www.informatik.tu-darmstadt.de/VS/Mitarbeiter/Fuenfrocken/

[6] Java Aglet Application Programming Interface White Paper, IBM Tokyo Research Laboratory, 1997, http://www.trl.ibm.com/aglets/JAAPI-whitepaper.html

[7] Mobile Agent Computing, A White Paper, Mitsubishi Electric ITA Horizon Systems Laboratory, 1998, http://www.concordiaagents.com/MobileAgentsWhitePaper.html

[8] http://leap.crm-paris.com/

[9] FIPA agent Management Specification, 2001: http://www.fipa.org/specs/fipa00023/

[10] A. K. Dey: Understanding and Using Context, Personal and Ubiquitous Computing, Special Issue on Situated Interaction and Ubiquitous Computing, 2001