User Interface Techniques for Mobile Agents

Matthias Grimm *Mohammad-Reza Tazari* *Matthias Finke*

Computer Graphics Center (ZGDV) e.V.
Fraunhoferstr. 5, 64283 Darmstadt, Germany
{Matthias.Grimm, Saied.Tazari, Matthias.Finke}@zgdv.de

Abstract

In mobile environments, the usage of mobile agents is very common. After being configured by the user, mobile agents can perform tasks autonomously and present their results when they have finished their task. In most cases, communication between user and agent is only necessary at the time of configuration and at the time of presentation, when the agent has accomplished its task. Our research activities deal with agent-based applications for mobile users with mobile devices of low processing power and without Java capabilities, which yields the question of how user interfaces can be implemented. In this paper, different approaches for the communication between users and agents are summarized and classified with respect to our applications. An agent platform gateway is introduced as our approach to user interface implementation and some services of the gateway are described in detail.

1 Introduction

Mobile agents are a very commonly used paradigm in mobile environments. They are software modules that are able to move through the network autonomously in order to accomplish a task they were given by their owner. In mobile environments, this ability can be used in order to reduce network traffic in online negotiations and thus reduce network costs, provided that the mobile device supports both the execution and the transport of agents by means of Java code. In this case, agents can migrate to the mobile device, be configured by the user for a specific task, and can migrate back to the network. There, they can carry out their task by negotiating with other agents on various platforms on distributed hosts, and finally, they can either migrate back to the device and present the results or can send a message to the user containing the results.
Mobile systems are characterized by manifold mobile devices with capabilities that vary over a wide range. Some typical restrictions are poor bandwidth, small displays, different operating systems and different Java capabilities. Designing and implementing software for these kinds of devices implies many considerations concerning the distribution of the software (client-side vs. server-side), programming language, and support of different platforms and capabilities.
The rest of the paper is organized as follows: In section 2, we discuss a couple of existing approaches to user-agent communication with their advantages and disadvantages. In section 3, we introduce the mobile agent framework we developed and, in section 4, our approach to HTML-based agent interfaces is described in detail.
In this paper, the concept ‘agent’ means a unit of Java software that is able to travel across the network between servers and which runs on an agent platform. By ‘mobile devices’, we mean PDAs. Laptops are mobile by means of being easily transportable, but they are not very handy to use, as they have to be booted, require considerable power and are quite heavy. We restrict our

consideration of devices to those that are only of limited Java capability, i.e. we do not have a full-blown Java VM on our devices.

2 Existing Approaches

Before discussing different approaches of user-agent communication, we want to present three different acts of communication as introduced in (Mihailescu, Gamage & Kendall, 2001).
The *initial interaction* enables the user to create an agent and set its instructions and start parameters. The *in-progress interaction* enables the agent to communicate with its creator during the processing of a given task. The agent might, for example, ask for some additional information, as the task might not be accomplished with the initial parameters. Finally, with the *completion interaction*, the agent can present the results of the completed task to the user.
We want to start our classification of approaches with respect to the location of the agent the user communicates with. Two cases are obvious: the agent might migrate to the user's device or the device is not capable of accepting and executing agent code. In the latter case, the agent can migrate across different servers in order to accomplish a given task for the user, but a different protocol has to be used for the communication between the user and the agent. We will discuss these two main cases in detail in the following subsection.

2.1 Methods of GUI Representation

2.1.1 Agent located on mobile device

When the device is capable of executing Java code, the standard methods for displaying a Java GUI – based on Swing or AWT – can be used. One problem of this approach is the code size of the agent, as it includes the code for displaying the user interface and the appropriate event handling. Since our focus targets mobile devices connected to the Internet via a radio network, code size is expensive. The other problem is that the devices and platforms we focus on are not capable of executing the agent platform and thus do not support an efficient secure migration of agents. And if the device was able to execute the Java code, the agent had to move to the device anytime communication with the user is necessary, which is again expensive due to the radio network. Therefore, this approach is not suitable for our system.
When agents migrate onto the user's device for communicating, the payload of the information the agent takes along with it must be large or the dialogue between the agent and the user must be complex. If both are not the case, the network costs are not worthwhile.

2.1.2 Agent located on server

When an agent can't migrate to the user's end-device, a different communication protocol has to be chosen for the communication between the user and the agent, and different methods are needed for the agent to display a user interface remotely on the mobile device.

Email One common approach is the presentation of task results as an email. This approach has numerous immediate benefits (La Corte, Puliafito & Tomarchio, 1999). The user can disconnect from the platform and gets a message when new information is present. Almost all devices support email and mail management software is cheap and widespread.
But, there are also disadvantages. The user is not able to interact with the agent when the agent has moved from its home-platform. One example of end-user interaction based on Active Mail (mail including executable code) is presented by (Schirmer, J. & Hayn, R., 1998).

HTML One major advantage of HTML as the user interface description and HTTP as the communication protocol is the platform independence. A web browser exists for almost every device, and as HTML is pure text, it can be compressed very efficiently. Therefore, agents can take along the compressed HTML code without growing too big in code size.
One problem is that mobile agents are difficult to address due to their mobility. When an agent carries a Java servlet with it, its URL changes with every migration. Therefore, something like a home environment is needed, which has a fixed URL. Since our agents communicate using the agent communication language FIPA-ACL, a conversion between HTTP requests, FIPA messages and HTML pages is needed. Another disadvantage is the lack of a mechanism for information *push*. When the agent has accomplished its task, it might want to push some task results to the user's mobile device. Since HTML only supports *pull* communication, an alternative solution is needed here.
Applets Very often UIs of Java mobile agents are realized as Java applets. (Da Silva, da Silva & Romao) present the combination of an agent, an applet and an HTML page as the *conventional model* of UI generation. They propose an integrated model offered by AgentSpace, where specific applets for each agent class are not needed. The idea is to use a rather generic applet that supports the management and presentation of the interfaces provided specifically by each agent. The authors differentiate between an *initialization interface* and a *management interface*, both provided as callbacks by the agent. When a user wants to interact with an agent, the generic applet is downloaded to the client's web browser. Then, it calls the management interface and the UI is displayed as a specific agent frame in the applet.

3 Our Mobile Agent Framework

Our framework is distributed across several logical units. As both users and agents can be mobile, there is the need for a personalized agent platform the users and the agents know about. We call it the user's 'homebase', and its main purpose is to facilitate homogenous access to the user's resources, quite similar to the *virtual home environment* (VHE) presented in (3GPP). The homebase consists primarily of the agent platform and some personalized services. As agent platform, we chose SeMoA (Secure Mobile Agents) developed by Fraunhofer IGD (Roth, V. & Peters, J.). SeMoA addresses several security aspects, such as protecting agents against malicious hosts and preventing agents from being changed by other users.
A context management service keeps track of the user's current location, as well as the location's characteristics – such as temperature, network access, and people present. As another part of the context, the device being used by the user and its characteristics – such as color and sound capabilities, display size, applications, as well as personal preferences of the user (static preferences, as well as conditional preferences) are provided by the context manager. Agents can use this context information in order to generate their dynamic user interfaces.
The agents running on the SeMoA platform can communicate using FIPA-ACL. As content languages, we use primarily XML and RDF. Agents have the ability to asynchronously send and receive messages or to send and receive messages synchronously, just like RPC. The agent tracking and locating service (ATLAS) of the platform facilitates the message routing between agents residing on different platforms on physically distributed systems.
As the platform is based on the Java programming language, the system on which agents should be executed has to supply support for the JDK 1.4., which is not the case for PDAs and Windows CE-based Pocket PCs. Since these devices are our target system platform, our mobile agents are not able to migrate to the end-devices in order to display their user interfaces or embed applets.

4 The Thin Client Gateway

Although the performance of mobile devices is evolving fast, there are technical restrictions, such as a missing Java platform that supports serialization, which is crucial for agent migration.
On the agent platform (homebase), there are a couple of agents that perform specific tasks. The agents communicate using FIPA-ACL. As the web-browser on the end-device does not understand

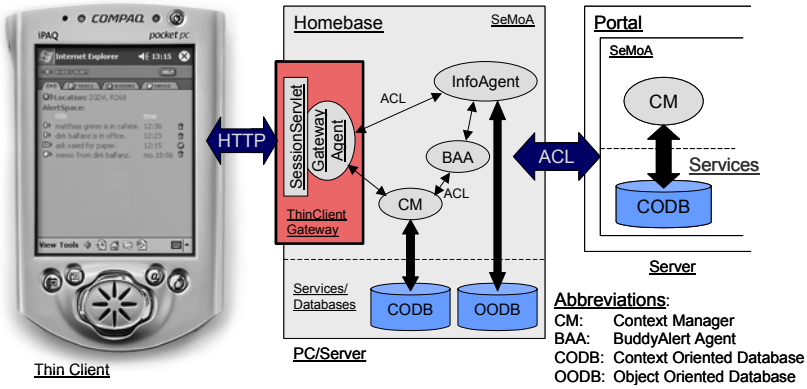

Figure 1 The homebase with the agent platform and the thin client gateway

ACL, there needs to be a translation service that translates between FIPA-ACL and HTTP. This service is the *thin client gateway*, serving as the entry point to the user's homebase (Figure 1).
The main components of the client gateway are a Java servlet (*SessionServlet*) as interface to the mobile device, and an agent (*Gateway Agent*) as interface to the agents located on the platform. Agents that provide a service for the user can register at the Gateway Agent, providing an HTTP request that invokes the service. This HTTP request contains an ACL message, which is interpreted by the gateway and sent to the specific agent when the user activates the link. As the receiver of the ACL message is a mobile agent, it might be located on a different platform than the homebase. The agent tracking service of the SeMoA platform can locate the agent and send the message to it. With this mechanism, we created a location-invariant communication protocol, which tunnels ACL messages through HTML and HTTP and vice versa. The homebase of the user does not change its location, thus, any agent can create HTML pages and send them to the user's device, enabling the user to send messages back to the agent, regardless of its location.

4.1 Services

Before an agent can use the thin client gateway for displaying the user interface, it has to register itself as a service. In order to register, an agent has to send an ACL message to the Gateway Agent where it provides a URL. The URL has to contain the web address of the SessionServlet and the ACL message that has to be sent in order to invoke the agent's service. When the user connects to the homebase, a list of all registered services is displayed, each entry as a link. Activating the link by clicking on it, the user makes the gateway send the encoded ACL message to the agent, waiting for a response. The agent can create an HTML page from a template, and return it as the content of a reply message. This page is returned to the browser by the servlet of the gateway.
The most important service for our applications is the user notification or *push* service. It enables a mobile agent to display a notification on a user's end-device. In most cases, this notification is the result of a transaction of the agent or it signals the change of the agent's state. The push service is

realized as a socket server on the end-device that listens on a specific port. When an agent needs to push a page to the end-device, the servlet sends its URL to that port, and the server causes the client's web browser to open the URL. If the device is not connected at the time the push event occurs, it is stored in the homebase and sent as soon as the device reconnects.

5 Conclusions

In this paper, we introduced some issues regarding user interface implementation for mobile agents. A couple of different approaches were discussed and evaluated for use with our system. Since our system deals with mobile devices of medium capabilities, the design and implementation of HTML interfaces turned out to be the most practicable approach.
We described our approach to an agent gateway service, which connects the agents' FIPA-ACL world with the widespread HTTP/HTML world, namely the *Thin Client Gateway*. With the gateway, the locations of the mobile agents do not change from the user's point of view, and the communication between user and agents is not restricted by the agents' capability to migrate across different platforms. The gateway could successfully be applied within the map project.
The static character of HTML is still a problem. A page cannot reload specific parts, but only a new page as a whole, which is a problem in mobile environments where communication is expensive. Furthermore, animations as result of user interaction are not possible. For these reasons, we are going to evaluate Flash as a UI language in the future within the *mummy* project.

References

3GPP Technical Specification 22.121 v4.1.0: "The Virtual Home Environment (Release 4)", March 2001.

La Corte, A., Pulifiafito, A., Tomarchio, O. (1999). An Agent-based framework for Mobile Users. 3rd European Research Seminar On Advances In Distributed Systems (ERSADS'99), Portugal, April 1999. Retrieved from http://citeseer.nj.nec.com/lacorte99agent.html

Da Silva, A.R., da Silva, M.M., Romao, A. (1999). User Interfaces with Java Mobile Agents: The AgentSpace Case Study. *First International Symposium on Agent Systems and Applications Third International Symposium on Mobile Agents*, Palm Springs, California, October 1999, retrieved January, 07, 2003, from http://berlin.inesc.pt/alb/papers/1999/asa99-asilva.pdf

FIPA (2002). FIPA ACL Message Structure Specification, Retrieved from http://www.fipa.org/specs/

Mihailescu, P., Gamage, C., Kendall, E. A. (2001). Mobile Agent to User Interaction (MAUI). *Proceedings of the 34th Hawaii International Conference On System Sciences (HICSS-34),* retrieved from http://www.pscit.monash.edu.au/~patrikm/maui.pdf

Lingnau, A., Drobnik, O. (1998). Agent-User Communications: Requests, Results, Interaction. In *Proceedings Second International Workshop, MA'98*, Stuttgart, Germany, September 1998, p.209ff

Roth, V., and Jalali, M. (2001). Concepts and architecture of a security-centric mobile agent server. In *Proc. Fifth International Symposium on Autonomous Decentralized Systems (ISADS 2001)*, Dallas, Texas, U.S.A., March 2001, 435-442.

Schirmer, J., Hayn, R. (1998): Multimedia authoring and synchronization within ActiveM3 - a system for composing active email messages as subtypes of mobile agents. *SYBEN'98*, *Interactive Multimedia Service and Equipment*, Zurich, Switzerland, 18-22 May '98, 456-465.