Context-Aware Services based on Secure Mobile Agents *

Ulrich Pinsdorf*, Jan Peters*, Mario Hoffmann† and Piklu Gupta‡

* Fraunhofer Institute for Computer Graphics
Fraunhoferstraße 5, D-64283 Darmstadt, Germany
{ulrich.pinsdorf|jpeters}@igd.fraunhofer.de

† Fraunhofer Institute for Secure Telecooperation
Rheinstraße 75, D-64295 Darmstadt, Germany
mario.hoffmann@sit.fraunhofer.de

‡ Fraunhofer Institute for Integrated Publication and Information Systems
Dolivostraße 15, D-64293 Darmstadt, Germany
gupta@ipsi.fraunhofer.de

Abstract: In this paper we introduce the concept of *context-aware services* as an extended form of location-based services. We describe a new architecture for realizing context-aware services which has a number of key benefits. This architecture relays on mobile software agents as security-centric middleware. Our approach allows to combine disjunct databases by establishing application-dependant links between remote data sets with a proprietary access. We describe all necessary roles and places to establish a multilateral security concept for all participants.

Keywords: Mobile software agents, context-aware services, location-based services, databases, profiles, security.

1 Motivation

Presently there is a wide range of information available on the Internet stored in disjunct databases which to date can only be queried individually. Additionally such publicly accessible databases are also typically designed for proprietary access which may also only be available on a limited number of computing platforms. The challenge therefore remains of being able to optimize the querying of these discrete data sources so that information can be *combined* in order to achieve the maximum benefit for the user. To this end we propose the creation of so-called *context-aware services*.

We define this term as tailor-made services delivered to the user on the basis of a number of factors, which are as follows: user's current location, user preferences based on a personal profile reflecting his/her usual habits, time of query, type of device to which service is being delivered. Thus context-aware services can be seen as the next logical development of currently well established location-based services. Commonly, they integrate multiple disjunct data sources in order to fulfill their task.

Availability of context-aware services can be beneficial in a number of scenarios both in the leisure and business domain; for instance a business traveller may whish to spent a fixed amount of time for sightseeing trip. This trip shall be arranged by means of a service stored on his PDA. Furthermore the service may not only provide a route map, but also deliver pictures and additional information on the different point-of-interests. This requires a whole lot of database accesses and most of them are context-dependand. For instance they depend on the user's average speed, age, interests, actual time, departure time, current location and departure location.

Personal preferences can determine the relevance of information delivered to a user; vegetarian tourists for instance will not have steakhouses recommended to

*The architecture is being deployed in the MOBILE project, which is funded by the German Federal Ministry of Education and Research. Please refer to http://www.mobile-projekt.de for more information.

them. The time of a query is a factor in determining the relevance of a query answer; shop opening times, which differ from country to country can therefore be considered in relation to the time of a query. Last of all, services which are aware of what kind of device they are being delivered to can obviate problems of inappropriate presentation formats; for instance large graphic files will therefore not be sent to a mobile device using a standard GSM connection for data transfer.

2 Approach

Since it is no longer a problem to store huge amounts of data, the problem has shifted towards the question of how to *find* specific information. Moreover, enabling the querying of disjunct databases with totally incompatible table structures or even different storage concepts is a major problem, which has still not been solved satifactorily.

In our opinion the sole way to integrate incompatible data sources is by means of proactive components. Proactive components, such as software agents, are able to establish links between data according to the application.

We suggest *mobile* software agents as proactive components realizing interconnections between data sources. Mobile agents offer the optimum solution due to their benefits with respect to aspects of security, flexibility and data compression. For each type of query in an application we suggest a specific agent. If a query has to be performed, the according agent will be instantiated, migrate to a number of data sources and return with a report which aggregates the results of its queries. Since it is sufficient for each agent to know the structure of the small number of specific databases it is working on to perform the query task, the development becomes very effective, scalable and therefore manageable on a large scale.

The interconnection between two data sources becomes even more valuable when context specific queries become involved as defined in section 1. Besides information about current time and the user's location usually used to provide location-based services (LBS) [1], information about the user's current situation, his habits and needs enables us to focus the query on a small subset of a huge amount of available data and thereby enables effective context aware data mining [6] as an extension to "common" LBS.

In other words: depending on the current user's context a specific agent creates an application-specific link between two or more databases. This link exists just as long as the agent has to be alive to provide the information. Of course, an agent could communicate its "experience" to other agents before it dies, but since the user's context [4] is highly dynamic as well, one would usally want to renew these database links for each query.

3 Architecture

We define different roles (see Figure 1) appearing in our approach and the dependencies between them.

So-called *service providers* with detailed knowledge about the interfaces to data sources of certain *information providers* develop specific services as mobile software agents. Those services will be collected, reviewed and published by *service brokers* as pieces of executable code. Besides structuring and filtering the variety of provided services quality inspection is a major issue. Through a cryptographic signature on the agent's code the service broker can be identified unequivocally as instance of trust, wherever the agent is to be executed. This is one requirement to set up fine-granular security policies on those agent servers on which an agent is executed to fulfill its tasks. Thus it is the job of the service broker to ensure the harmlessness of agents by means of the often addressed malicious agent problem [3], as far as possible.

An *user* who wants to access a service downloads this service as a mobile agent from the service broker and installs it on his mobile device - subsequently he is able to benefit from its features. Two kinds of services are conceivable here: those which are instructed by the user in direct interaction before they investigate the validity of the information and those which are activated once and then run as autonomous background processes providing the user with information and assisting him in case of special events.

To optimize the instruction process of services the *personal profile* keeps track of the user's preferences. These include the user's generic favours, program settings, and productive data such as datebook, phonebook, or diary. Agents will query the profile database

if they require certain information about the user. An user will therefore not be inconvenienced by information requests until a requested piece of information cannot be found in the profile. In that case it is the task of the personal profile as active component to request the required information in communication with the user. Of course the user has the possibilty to alter automatic generated settings, add new settings, or delete particular entries from the profile.

The underlying infrastructure implicitly defines several logic places separated by means of different security domains and is described in the following.

Mobile device The mobile device acts as the standard unit of execution for the user's needs. Usually it has small computing power, but a high connectivity. It stores just the information which is actually needed and those which enable the user to continue work even if a network connection is not available. The mobile device could be equipped with additional (generally plugable) extensions such as GPS receivers for the user's current location.

User's homebase The homebase keeps the user's personal profile. This data should be (authorized) accessible via the Internet as well. The user is always reachable by means of his homebase since the home base remembers the mobile device that was recently used by the user, e.g. by the device's IP address. A messenger agent sent by Alice with the order to meet Bob would migrate first to Bob's home base. A stationary personal agent at Bob's homebase would check the identity and intention of Alics agents, and occasionally allow him to migrate to Bob's current location in order to hand over the message.

Service portal The service portal provides services for download. These services are manifested as a group of agents[1] which interact with the user, migrate back and forth between mobile device, homebase and data sources, and perform the task accoring to the users preferences. The service provider usually integrates database access, which are coordinated with the information providers. Hence he can rely on different but stable APIs for each data source. Query agents could access them by means of standard protocols (e.g. SQL or LDAP) and even migrate to the sources and access the data locally. Hence, the agent metapher allows to combine disjunct data sources which only could be managed difficultly otherwise.

Execution server The execution server provides both the runtime environment for mobile agents and access to data sources. Hence, one could interpret it as a mobile agent server providing an interface which enables agents to access local data. There are three types of execution servers. First, execution servers run by information providers. These kind states the majority of the exectuion servers. Second, the homebases which have additional functionality such as running the personal agent, hosting profile data, etc. And third, the mobile devices which act as execution servers with the ability of direct agent-user interaction. We refer to the concrete architecture of the agent servers as well as to security aspects in section 4.

Basic services The basic services provide access to server functionality for the agent. These services are typically for data access, special computing algorithms, encrypting[2] and decrypting, or user interaction. In conclusion basic services allow an information provider to make the own data accessible for mobile agents and hence combinable with other data sources he has no knowledge about.

Each of these places belong to the security domain of the operation authority. The sum of places provide multilateral security for all participants. The mobile device and the homebase are typically controlled by the user, whereas the execution servers and the basic services are under supervision of the resceptive provider. This has two main advantages. First, an agent is able to perform secure computations in the user's security domain which lessens the malicious host problem. Second, an information provider keeps full control over the provided data. He could implement any data access policy he desires. This should be kept in mind when developing secure services, since frequent migations between these security domains can be less expensive than most algorithms against malicious hosts.

[1] These may be both mobile and stationary agents. Typically a service requires both types.

[2] Encryption and signing operations should always performed by a service at the user's homebase, since it is no good idea to equip a mobile agent with a secure communication key.

4 Secure mobile agent platform

As middleware for data access we need a mobile agent platform which provides the runtime environment for the agents. It is quite clear that this platform should follow a secure aware design concept.

For our architecture we used the SeMoA (Secure Mobile Agent) platform [5], which fits our needs best. The SeMoA project developes an open server for mobile agents with a special focus on all aspects of mobile agent security, including protection of mobile agents against malicious hosts. SeMoA builds on JDK 1.3 and is a "best effort" to provide adequate security for mobile agent systems, servers as well as agents [2].

The security architecture of the SeMoA server compares to an onion: agents have to pass all of several layers of protection before they are admitted to the runtime system (see Figure 2) and the first class of an agent is loaded into the server's JVM.

The first (outer) security layer is a transport layer security protocol such as TLS or SSL. This layer provides mutual authentication of agent servers, transparent encryption and integrity protection. Connection requests of authenticated peers can be accepted or rejected as specified in a configurable policy. The second layer consists of a pipeline of security filters. Separate pipelines for incoming agents and outgoing agents are supported. Each filter inspects and processes incoming/outgoing agents, and either accepts or rejects them. Subsequent to passing all security filters, SeMoA sets up a sandbox for the accepted agent (which can be regarded as layer three). Each agent gets a separate thread group and class loader. This class loader supports loading classes that came bundled with the agent, as well as loading classes from remoted code sources specified in the agent. Agents cannot share classes so one agent cannot not load a Trojan Horse class into the name space of any other agent. Agents are separated from all other agents in the system; no references to agent instances are published by default. The only means to share instances between agents is to publish them in a global *environment*. Each agent gets its own view on this global environment, which tracks the instances registered by that agent. All published objects are wrapped into proxys which are created dynamically.

SeMoA supports a lot of (security) standards, such as JAR archive format, X.509, PKCS, and ASN.

5 Conclusions

The advantages of our approach fall into three broad categories: flexibility, feasibility and security.

New query agents can be easily integrated in an existing query infrastructure which is a major building block for the high flexibility of the presented approach. Since each agent establishes just application-dependend links between disjunct data sources, the major problem of integration databases is broken in smaller parts, which guarantees its feasability. The security of the undelying mobile agent system allow fine grained access control and secure transport of valuable query results.

References

[1] Ajay Magnon and Reena Shukla. LBS, the ingredients and the alternatives. In *The Asian GPS Conference 2001 Proceedings*, pages 113–117. GISdevelopment, 2001.

[2] Volker Roth and Mehrdad Jalali. 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. IEEE Computer Society Press.

[3] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*, volume 1419 of *Lecture Notes in Computer Science*, pages 44–60, Berlin Heidelberg, 1998. Springer Verlag.

[4] Jürgen Schirmer and Holger Bach. Context-management within an agent-based approach for service assistance in the domain of consumer electronics. In *Proceedings of Intelligent Interactive Assistance & Mobile Multimedia Computing (IMC2000)*, Rostock, Germany, November 2000.

[5] SeMoA Project. Website. Available at URL `http://www.semoa.org/`.

[6] Christopher Westphal and Teresa Blaxton. *Data Mining Solutions - Methods and Tools for Solving Real-World Problems*. Wiley Computer Publishing, 1998. ISBN: 0-471-25384-7.

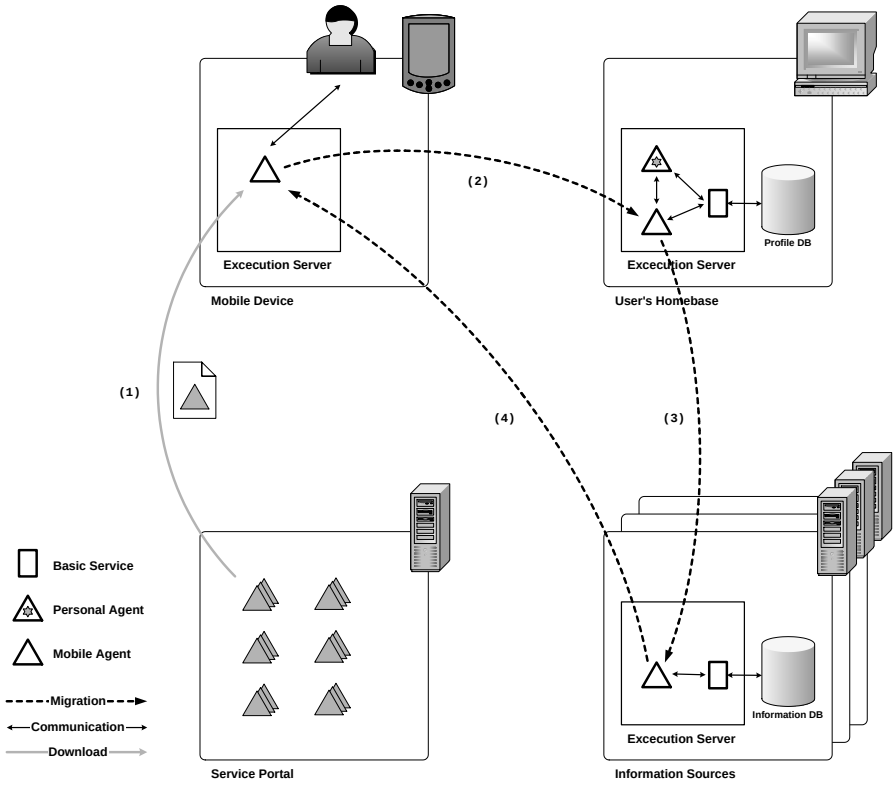

Figure 1: General architeture with roles and places.

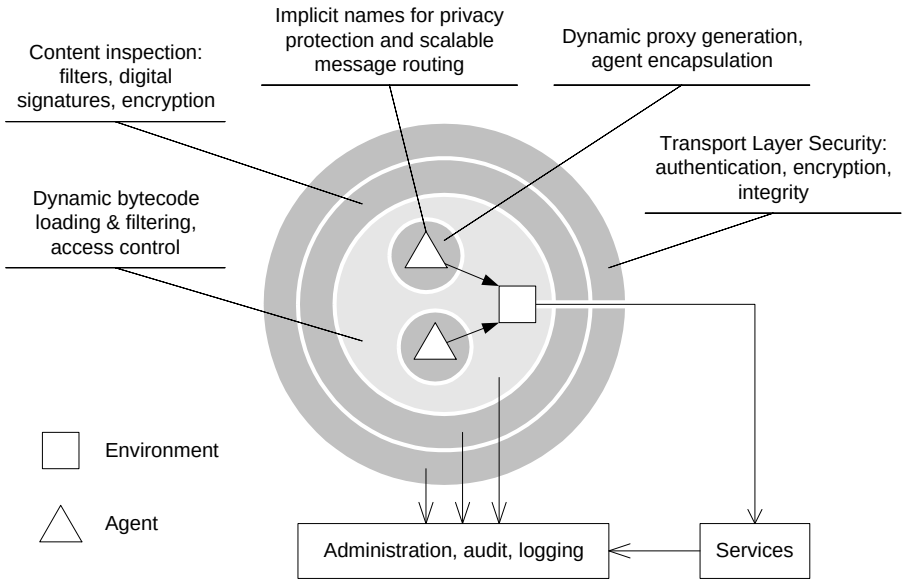

Figure 2: The SeMoA security architecture.

Citation:

Pinsdorf, Ulrich; Peters, Jan; Hoffmann, Mario; Gupta, Piklu:
Context-Aware Service based on Secure Mobile Agents.
In: Rozic, Nicola (Ed.) u.a.; IEEE Communications Society u.a.: 10th International Conference on Software, Telecommunications & Computer Networks. SoftCOM Proceedings 2002.
Split - Croatia : University of Split, 2002, pp. 366-370