Flexible Middleware Support for Future Mobile Services and Their Context-aware Adaptation

Marcin Solarski[1], Linda Strick[1]
Kiminori Motonaga[2]
Chie Noda[3] and Wolfgang Kellerer[3]

[1] Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31a
10589 Berlin, Germany
{solarski, strick}@fokus.fraunhofer.de

[2] NTT DATA, Kayabacho Tower Bldg. 21-1 Shinkawa 1-chome Chuo-ku Tokyo
motonagak@nttdata.co.jp

[3] DoCoMo Communications Laboratories Europe GmbH, Landsberger Strasse 312,
80687 Munich, Germany
{noda, kellerer}@docomolab-euro.com

Abstract. This paper presents a flexible peer-to-peer-based middleware for future user-centric mobile telecommunication services, which supports key functionalities needed to address personalization, adaptation and coordination of services running on top of it. The underlying communication pattern is based on dynamic negotiation that enables interworking of autonomous decentralized entities in a rapidly changing and open environment. This paper focuses on the middleware's support for context-aware adaptation of a multimedia service for mobile users. Service adaptation takes into account both user preferences and contextual changes to modify the service behavior and contents. The middleware implementation is based on JXTA extended by a mobile agent platform and is deployable on a range of mobile devices including mobile phones and PDAs.

1 Introduction

With the recent developments in 4G mobile environments (heterogeneous access networks, 4G radio access, ad-hoc sensor networks) and the large availability of any kind of mobile computing devices such as Personal Digital Assistants, mobile smart phones, we are experiencing the availability of an increasingly powerful mobile computing environment, which is exposed to high dynamicity.

In a future mobile environment, the emergence of ubiquitous computing is envisioned [1]. Heterogeneous devices, ranging from low-end devices (e.g., tiny sensor devices) to high-end devices (e.g., a 3D video streaming terminal), are surrounding humans. Heterogeneity of radio access networks (e.g., 3G/4G radio access and ad-hoc

sensor network) is another attribute of future mobile services [2]. Thus particular users' environments, including devices, networks and, services have to be adapted and configured according to user's changing environment, context, status, and preferences, i.e. *context awareness*. To enable access to services from heterogeneous devices through heterogeneous access networks, *context-aware adaptation* is one of the fundamental requirements of a middleware for future mobile services.

Furthermore, there is no centralized gateway to that all available services are registered or that acts as an intermediary component to support their adaptation to the environmental contexts such as network bandwidth and terminal capabilities. These requirements lead us to design a novel middleware platform, called Mercury which is an acronym for ***M****iddl****E****ware a****RC****hitecture for* ***U****ser cent****R****ic s****Y****stem*, which supports context-aware adaptation of mobile services and is based on a peer-to-peer and asynchronous interaction paradigm.

The structure of this paper is as follows; Section 1 gives a short introduction to future mobile services and the Mercury architecture. In Section 2, the overview of the Mercury middleware is given and the communication pattern called Dynamic Service Delivery Pattern in presented briefly. Related work and the advantages of our approach are described in Section 3. In Section 4, we discuss our design of context-aware adaptation and related subsystems realized in the Mercury middleware. Section 5 discusses an application scenario, along a video-on demand service, to show how context-aware adaptation works by utilizing the middleware subsystems. Section 6 summarizes the approach towards a flexible middleware support for context-aware service adaptation.

2 Mercury Overview

Mercury is a service middleware for user-centric service provisioning in mobile environments driven by the demands and the behavior of users. There are two fundamental requirements for a middleware platform for mobile services.

One is to support asynchronous communication, since wireless access can be disconnected when mobile devices are out of the coverage area, or due to data rate variation, since the downlink data transmission rate is usually much faster than uplink. Another requirement is the distribution of middleware components to resource-constrained devices, in terms of low performance, limited memory, and low power consumption. These requirements heavily influence the decision on the selection of enabling technologies to build our middleware platform.

A dynamic service delivery pattern, which enables to publish/subscribe advertisement of services, to negotiate interactively between peers, and aggregate for service execution, is introduced in Section 2.2.

2.1 Design Overview

In the Mercury middleware, the complex functionality needed to match the above requirements on future mobile services is divided into a number of subsystems, which

are then grouped into three layers for the sake of modularity and reusability[7,8]. In this way we introduce an additional user support layer on top of a service support and traditional network layer. The user support layer includes subsystems for *Adaptation*, *Community*, and *Coordination*. The introduction of this layer reduces unnecessary user interactions with the system and enables the provision of user-centric services. Furthermore, the user support layer allows the establishment and maintenance of virtual ad-hoc communities, and their coordination toward a common guidance for service usage.

The service support layer contains most functionality of traditional middleware, such as *Discovery & Advertisement* for getting establishing entity communication, *Authentication & Authorization* for secure communication and *Contract Notary* used for managing service agreements. These subsystems are used by the *Dynamic Service Delivery Pattern*, which is the basic interaction paradigm between the Mercury entities (see Section 2.2). This paper focuses on the other subsystems in this layer, namely *Transformation*, *Context Management* and *Profiling*, used for context-aware adaptation (i.e. used by *Adaptation* subsystem); these are presented in more detail in Section 4. Finally, the network support layer provides connectivity in IP-based networks, including *Mobility management*, *Call & Session management*, and *QoS management* subsystems.

The above-mentioned requirements of asynchronous communications and resource-constrained devices support lead us to build our middleware on a mobile agent platform, enago Mobile [9] in our current prototype. Mobile agents support asynchronous communications and can autonomously work temporarily disconnected from other facilities. Connectivity is only required for the period for migration, but not for service execution. They also solve the problem of resource-constrained devices by allowing on-demand and flexible deploying code to both mobile devices and the infrastructure.

2.2 Dynamic Service Delivery – underlying interaction pattern

The Dynamic Service Delivery Pattern describes the process of (1) agreeing on entity interaction behavior, called negotiation scheme, leading to a service agreement, (2) establishing the concrete service agreement itself, which determines the conditions of the service delivery and, finally, (3) managing the service delivery phase using the notion of a service session. A Mercury entity is any piece of software which can play one or more roles, including the predefined ones: Participant, Coordinator (negotiation-related roles), User and Service Provider (service delivery-related roles). Optionally authentication and authorization mechanisms can be used, which are defined in separate middleware subsystems. The dynamic service delivery pattern is used for every communication between any Mercury entity, including the middleware subsystems.

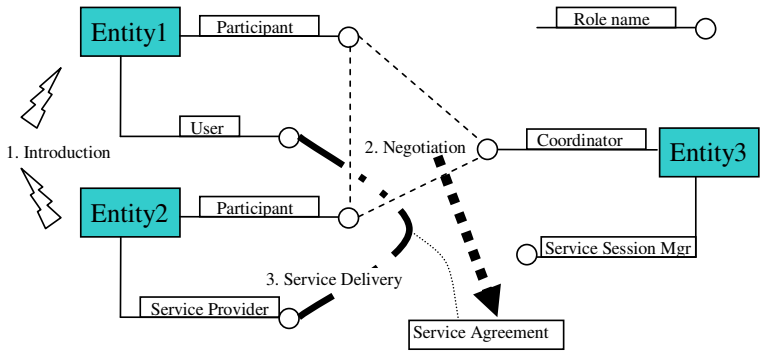

Fig.1. Example entities participating in the Dynamic Service Delivery Pattern

The dynamic service delivery pattern contains three subsequent phases:

1. *Introduction Phase* – entities are introduced to each other by using the publish/subscribe interface for service discovery and advertisement.
2. *Negotiation Phase* – entities interact in a negotiation dialog, playing the role of *Participant* or *Coordinator*. The negotiation handles negotiation schemes, credentials, proposals and agreements in order to reach a service agreement.
3. *Service Delivery Phase* - entities, in the role of *Service Provider* and *User*, interact to fulfill the terms of the negotiated contract. The service delivery phase is organized as service session, which is managed by an entity playing role *Service Session Manager*.

Figure 1 depicts example entities participating in the negotiation phase of the Dynamic Service Delivery Pattern. It is assumed that `Entity1` and `Entity2` have been introduced to each other using the *Discovery and Advertisement* system and that there is an entity which they both trust so that it can coordinate the negotiation process. The negotiation is then started by agreeing on the negotiation scheme, followed by the actual negotiation interaction, which is coordinated by `Entity3`, playing the role of the coordinator. As a result of the negotiation process, a service agreement is established which is then distributed to all the participants before the service delivery phase starts, i.e. the service session is started by the Service Session Manager (in the example `Entity3`). A more detailed description of the Dynamic Service Delivery Pattern can be found in [7].

3 Related Works

There have been quite a lot of research efforts for adaptation of services and context-aware adaptation. [3] proposes a framework for multimedia applications adaptation, by utilizing differentiation within IP session(s) based on proprietary of packets indicated by QoS-aware applications. [4], a middleware platform for context aware services is built on active networks and mobile agent technologies, where active networks

is used as communication channels for mobile agents, i.e. mobile agents migrate to other nodes are realized by an active packet containing data and program in active networks. In [5], the tasks for adaptation of multimedia applications are divided into two: selection of the most applicable adapted output format in accordance with the information form the Composite Capability/Preference Profiles (CC/PP) [6], and adaptation execution by using different adaptation agents. Context-aware adaptation of web-line service content has been a hot topic in the literature [10,11]. In [10], the author present a proxy-based (centralized) content adaptation architecture and an adaptation mechanism based on content semantics analysis and a device-independent model for content description. In their implementation, they apply CC/PP-based Universal Profiling Schema to define both different type of user context, including device profiles and XQuery for extracting contextual information relevant to the content adaptation process. The approach presented in [11] presents an adaptation engine which automates selection of content transcoding mechanisms considering user preferences, device capabilities, networking parameters as well as content metadata so that the content presentation quality is optimal. The selection is determined user-given weighted quantification of content quality along various quality axes. Previous work [12], focused on application-aware adaptation, which proposes building underlying systems to discover the context and notify the application to take the adaptive actions. Such applications become complex as they have to handle the adaptation by themselves. Mercury follows systems which advise hiding the adaptation complexity into the underlying system layers.

In our approach, we design a flexible middleware for context-aware adaptation by introducing a dynamic service delivery pattern as a generic negotiation framework between peers, which can be applications, devices, or even subsystems in the middleware. The negotiation scheme defines the negotiation protocols [13], which are used to automatically exchange and align contracts. Compared to other approaches to context aware service adaptation, Mercury has the following advantages:

a) *Adaptation as an aggregated service.* The adaptation subsystem is designed so that it combines a number of adaptation strategies specialized for different services to adapt. A most suitable strategy is automatically chosen at the moment of negotiation the usage of the adaptation; the selection is done by evaluating the meta information of the service to adapt as well as the auxiliary resources and services that may be used.

b) *Adaptation dynamics.* On one hand, adaptation offers adapted version of some existing services, on the other, it uses other services, including context providers, profile managers and transformers when needed. Like other Mercury entities, the adaptation subsystem follows the Dynamic Service Delivery Pattern both to provide its functionality and to use other services and resources it needs. Thus, the actual adaptation functionality is created dynamically and may include interactions to spontaneously discovered services needed to achieve the adaptation objective.

c) F*unctionality decoupling and modularization.* The Mercury middleware defines two key subsystems needed for service adaptation: adaptation and transformation subsystems. Whereas the first defines so called adaptation strategies, which realize the autonomous functionality and logic determining the conditions when some and what adaptive actions are to be taken, the transformation subsystem maintains concrete adaptation mechanisms, called transformers, which are used by the adaptation strate-

gies. Additionally, the context-related and profile-related functionality is defined in separate middleware subsystems.

4 Service adaptation support in the Mercury middleware

The complex tasks needed to perform user-centric service adaptation is divided into four subsystems in the Mercury middleware, as mentioned in section 2. These are the two key subsystems, adaptation and transformation, and two supplementary subsystems, context management and profiling. The Adaptation subsystem implements the adaptation logic specifying how to change the target service and the related resources under certain conditions on the service context, the Context Management and Profiling subsystems are used by the adaptation to (1) acquire needed information on the service context and relevant preferences, and the Transformation subsystem to (2) carry out the service adaptation and other resource usage adjustment.

The interrelations of the subsystems are shown in a UML class diagram in Figure 2.

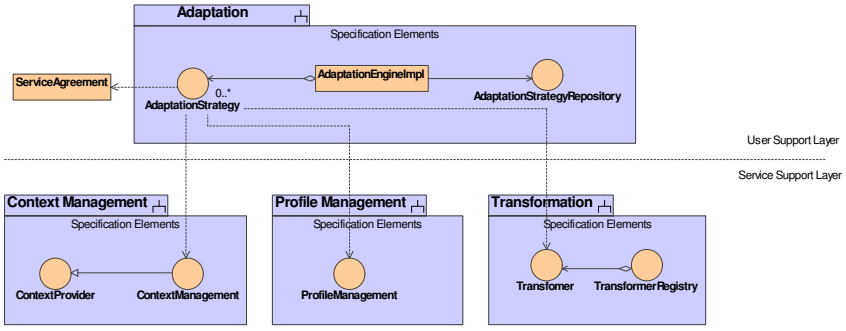

Fig. 2. An design overview of the interrelationships between Adaptation subsystem and other Mercury middleware subsystems involved in service adaptation

The application of each subsystem is illustrated with an example used in our application scenario described in Section 5.

4.1 Adaptation

The Adaptation subsystem is designed to support any type of service adaptation by (1) personalization the service to user profiles (e.g., preferences) and (2) modifying the service considering the user contexts (e.g., user environmental capabilities and available resources). The adaptation subsystem is implemented as a Mercury service. Its adaptation strategies handle adaptation specialized with regard to the service type and available resources are maintained in a repository. In this sense, the adaptation subsystem can be seen as an aggregation of specialized adaptation services. Whenever service adaptation is requested, the requestor has to provide some information on the service itself and the resources, including other services, which have to be used. The

adaptation service checks the availability of adaptation strategies suitable to handle the given service during the negotiation process. If a matching strategy is registered within the subsystem, the service adaptation is possible and a service agreement may be established only if the negotiating parties agree on the non-technical factors as well. This behavior is facilitated by implementing a specialized negotiation template which allows for checking the demands on the registered adaptation strategies.

The framework for defining concrete adaptation strategies allows for flexible adaptation to specific services. Strategies compliant to that framework may be inserted into the system on demand and are provided by the adaptation service. An adaptation strategy in the model represents a process of service adaptation which is expressed by transforming the service to the required quality, as specified in the service agreement. The service agreement includes information on the constraints on the quality of service as well as on available resources and auxiliary services to use for adaptation of the given service. Each concrete adaptation strategy defines the relevant context information needed to react to and retrieves it in a preferred way by using one or many context providers, or context managers, from the management system.

An example adaptation strategy has been implemented for the use in the Video Service scenario described in Section 5.

4.2 Context Management

Context-aware services are those that make use of the context information for their purposes. In particular, adaptation strategies in the adaptation subsystem belong to this class of services.

Context is a set of environmental states that either determines an application's behaviour or in which an application event occurs. Contexts can be classified in the following four categories: *user context* (e.g., the user's location, and the current social situation), *time context*, *physical context* (e.g., lighting, traffic condition, temperature), and *computing context* (e.g. communication bandwidth, available memory and processing power, and battery status). The Context Management subsystem offers a means to provide context information in either a push or pull mode. Moreover, the context information can be pushed either if the required context matched the given criteria or at regular time intervals. The subsystem has also a mechanism to manage access to the so called *Context Providers*, which are the entities that actually sense and retrieve some specific context, e.g., via sensors. Figure 3 presents the relation between context provider, context management subsystem and context-aware services. Even though *ContextProvider*s may also be directly used by context-aware services, The Context management system provides additional functionality to aggregate and filter context of several context providers.

4.3 Profiling

Profiles are collections of data that may be used to adapt services to a user's specific environment and preferences. Profiles can be classified to the following categories:

user profile, *service profile*, *terminal profile,* and *network profile*. This subsystem supports management of profiles, by offering a means to select, retrieve, store and update profiles to users. A user may have more than one profile and select the most appropriate one, or it may be chosen automatically upon other criteria. Additionally, each profile include access right information: two access rights are distinguished, profile read and write, which can be granted to different entities or entity groups. The profile manager is the component responsible for giving access to the managed profiles according to the access rights. As profiles may include references to other profiles, possibly managed by other profile managers, the profile manager may optionally access other profile managers.

```
[CCPP:component]
  ┌ RDF:ResourceDescription ->
  │    mercury:ReadACL -> ["All"]
  │    mercury:WriteACL -> ["ResourceOwnerID"]
  │    mercury:ProviderUuid -> "uuid1"
  │    mercury:ResourceUuid -> "uuid2"
  │    mercury:ResourceType -> "VideoDisplay"
  │
  └ RDF:TerminalPlatform ->
       mercury:ReadACL -> ["All"]
       mercury:WriteACL -> ["ResourceOwnerID"]
       [CCPP:HardwarePlatform] - CCPP:BitsPerPixel- > 16
       [CCPP:HardwarePlatform] - CCPP:ScreenResolutionX ->240
       [CCPP:HardwarePlatform] - CCPP:ScreenResolutionY ->320
       [CCPP:HardwarePlatform] - CCPP:NetworkAdapter ->802.1b
```

Fig. 4. An example of a CC/PP –based device profile in the graphical notation

In the Mercury prototype, the profiles are represented as XML documents compliant to the W3C CC/PP standard [6]. Each profile includes a part on the resource description and additional resource-specific parts called CC/PP components which correspond to different aspects of the resource the profile describes. An example profile, presented in Figure 4 in a semi-graphical notation proposed in the W3C documents on CC/PP, of a graphical display device contains an additional component describing the hardware-specific parameters of the terminal (`TerminalPlatform` in the example), which includes another component `HardwarePlatform` with its `ScreenResolutionX` and `ScreenResolutionY`. The example also contains some statements determining access rights to the associated components: the part on resource description can be accessed by all, whereas it can be modified only by the resource owner having the `ResourceOwnerID` identifier.

4.4 Transformation

The transformation system defines a framework for handling software components, called transformers, processing some service data, typically stream-oriented contents, for the purpose of service adaptation. Transformers are defined as components implementing a process of changing the input data to some other data produced at the transformer output. It is assumed that the data to transform come from a number of

sources that the *Transformer* may retrieve by itself or it is notified of the data availability. The transformed data at the transformer's output may be propagated to a number of *sinks,* which may be possibly other transformers. Thus, transformers may be chained to perform data in a more advanced or to reuse the existing transformers performing parts of the needed transformations. There are several aspects of the service contents that can be transformed: *format and encoding* (e.g. H.263 or MPEG1 streaming formats in case of the video contents), *type* (e.g. video, audio and text), and *structure* (e.g. XML and HTML).

The transformation framework is extendable so that it allows for inserting new transformers that perform some specific data transformation. An example transformer that was developed for the purposes of the application presented in Section 5 is concerned with processing video streams. This transformer is responsible for (1) on-the-fly redirecting the video contents ordered by the user to a given display and (2) modifying the stream quality of service parameters. The redirection of the video stream is done out-of-bound so that the transformer does not have to be deployed on one of the intermediate nodes between the video server and video player; instead the transformer uses a control protocol to make the server send the video to the given receiver. The display is assumed to be capable of receiving video streams and playing them back so that the play back can be remotely controlled. With regard to the other transformer functionality, the video stream QoS is modified so that the video stream can be optimized to match the hardware constraints of the target display, like the maximum screen resolution or supported number of colors in bits per pixel, and the video processing capabilities of the software controlling the display.

5 Example Application Scenario

The text below describes the application scenario developed using the Mercury middleware. The core of the application is a video-on-demand service that is leveraged by the context-aware adaptation supported by the underlying middleware.

A passenger enters an airport lounge before boarding a flight. Attracted by personal video booths in the lounge, she decides to watch her favorite movie in one of them. Unfortunately, all of booths are occupied and she decides to start watching the movie on her mobile device until a booth is available to her. After accessing the video streaming service in the lounge service menu and choosing the movies she likes, the system offers to reserve a booth for her to watch the movie in a more comfortable environment. A video player appears on the display of her mobile device and she starts watching the video. A notification about an available booth draws her attention while watching the video, and so she suspends the video and gets into the booth. After she makes herself comfortable in the booth, she resumes the video on her mobile device. Now the video is projected onto the big display in the booth and the movie sound comes out of the hi-fi sound boxes that are automatically selected by the context-aware system according to her preferences. The passenger may continue watching the movie until a notification pops up to remind her that boarding gate closes soon. She suspends the video again and proceeds to a boarding gate.

5.1 Events Occurring in Application Scenario

Figure 5 shows an overview of major events occurring in the middleware. Resources and services are detected considering user's context or preferences. The detected services are negotiated between the user and their providers through the intermediation of a coordinator agent. After reaching agreement, resources are adapted or personalized according to the service agreement, user's context or preferences.

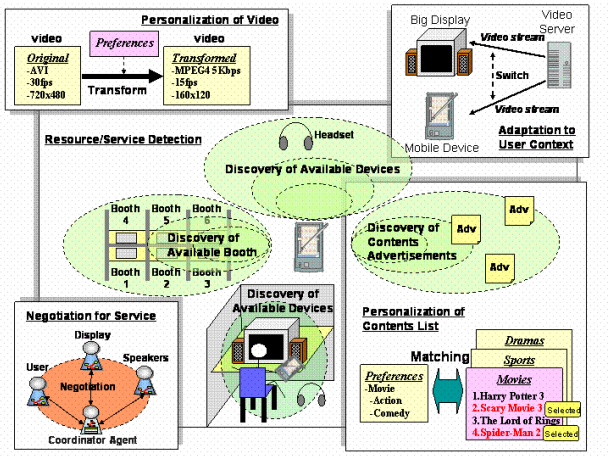

Fig. 5. An overview of the events occurring in the application scenario

5.1.1 Negotiation for Services

In the scenario presented above, a passenger wants to watch some video content. In order to watch the movie, user and provider need to negotiate about the usage of the content service according to the Dynamic Service Delivery Pattern as introduced in Section 2.2.

1. A video content provider issues an advertisement of the video contents, i.e. the movies, he offers. The advertisement includes the video content description, the proposed video cost as well as some information on how to contact the provider for further negotiation. On the other hand, the software acting on behalf the passenger, the user agent, looks for video with some characteristics determined by the user preferences (e.g. movie genre, favorite actor) and possibly discovers a matching advertisement sent by one of the video content providers. If there are many different contents available and matching the user criteria, the passenger chooses one and a negotiation with corresponding video providers may start.
2. The user agent and the video providers, in the role of participants, interact in a negotiation dialog, exchanging negotiation schemes, credentials, proposals according to the agreed negotiation scheme until the negotiation is complete (the participant agree on proposals of others) and a service agreements can be established.

Compared to the information in the initial advertisement, the proposals add some details on the quality of the contents to provide (in this case the maximal video size, language variant) or just modify the initial values of the data, e.g. the final price for the movie. The resulting service agreement includes some information needed to start and use the service, like the access data to the movie.

3. When the service delivery starts (here requested by the passenger), all the participants are requested to be ready. In case of the video provider, it means that the content in question has to be made accessible to the user according to the terms in the service agreement.

5.1.2 Adaptation to User Context

In the application, the user wants to watch the movie at the best possible quality, which means here using the best available display and with best quality supported by this display. The adaptation subsystem includes a strategy which allows to detect available displays around the user and to redirect the video stream so that it is displayed at the currently best display with a video quality optimized to that display. This strategy is used in this scenario so that it subscribes for the availability information of device nearby (context information) and it installs the specialized video transformer on the PDA used by the user. To perform the latter, the user has to provide some execution resources on the PDA so that the transformer code can be deployed there; this resource is one of the requirements of the adaptation strategy announced in the negotiation process. The transformer allows for controlling the local display on the PDA or any other intelligent display that can be controlled remotely using an RTSP-like protocol. It is deployed on the user's PDA to efficiently intercept the control commands for the player coming from the GUI operated by the user on the PDA. Whenever the adaptation detects a better device, the transformer is reconfigured to redirect the video stream and modify the quality of video.

6 Conclusions

In future, mobile service provisioning will include not only services that are offered by a central server system but also services that have to work in the current users' environment, independent from the facilities available in the infrastructure. Furthermore, future mobile environments will be characterized by co-existence of heterogeneity of devices and networks, including also resource constrained and intermittently connected devices and by rapidly changing context. To deal with such ubiquitous services, designing of a flexible and smart support for service adaptation is a major challenge.

The Mercury middleware proposes some solutions to this challenge. Its design benefits from the peer-to-peer communication paradigm and autonomously interacting software agents, which address the issues of decentralization, loose-coupling and making use of intermittent connectivity. These features are complemented with a flexible interaction paradigm, called Dynamic Service Delivery Pattern, which allows for secure and dynamic interworking of 3rd party software components. The Mercury

prototype has been implemented on top of a mobile agent platform extended by peer-to-peer communication mechanisms of JXTA.

Furthermore, the Mercury middleware provides strong user support for personalized, context aware service provisioning. In a heterogeneous mobile environment adaptability is a key feature for the success of such system. A flexible service adaptation is achieved by using specialized subsystems dealing with adaptation logic, context management, user profiling, and transformation mechanisms as illustrated in this paper.

We believe that the Mercury middleware system is an essential contribution to next generation mobile service provisioning, since it addresses the needs for advanced personalization and the identified requirements on service provisioning in a ubiquitous mobile environment.

References

1. M. Weiser: The Computer for the 21st Century. Scientific American, September 1991, pp. 94-104
2. H. Yumiba, K.Imai, M.Yabusaki: IP-Based IMT Network Platform. IEEE Personal Communications Magazine, October, 2001
3. H. Shao, W. Zhu, Y. Zhang: A New Framework for Adaptive Multimedia over the Next Generation Internet
4. I. Sygkouna, S. Vrontis, M. Chantzara, M. Anagnostou, E. Sylas: Context-Aware Service Provisioning on Top of Active Technologies. Mobile Agents for Telecommunication Applications (MATA), Marakech, Morocco, October 2003
5. M. Metso, J. Sauvola: The Media Wrapper in the Adaptation of Multimedia Content for Mobile Environments. Proceedings SPIE Vol. 4209, Multimedia Systems and Applications III, 132-139, Boston, MA
6. W3C: Composite Capabilities/Preference Profiles: Structure and Vocabularies 1.0. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/
7. C. Noda, A. Tarlano, L. Strick, M. Solarski, S. Rehfeldt, H. Honjo, K. Motonaga and I.Tanaka: Distributed Middleware for User Centric System. WWRF#9 conference, Zurich, July 2003
8. WWRF, Wireless World Research Forum WG2, Service Infrastructure of the Wireless World. http://www.wireless-world-research.org/
9. IKV++ Technologies AG: enago Mobile Agent Platform. http://www.ikv.de/content/Produkte/enago_mobile.htm
10. T. Lemlouma, N.Layaïda: Context-Aware Adaptation for Mobile Devices, IEEE International Conference on Mobile Data Management, Berkeley, CA, USA, January 2004, pp. 106-111
11. Wai Yip Lum and Francis C.M. Lau: A Context-Aware Decision Engine for Content Adaptation, IEEE Pervasive Computing, July-Sept. 2002, p.41-49.
12. Brian D. Noble, M. Satyanarayanan, Dushyanth Narayanan, James Eric Tilton, Jason Flinn, Kevin R. Walker: Agile Application-Aware Adaptation for Mobility, Sixteen ACM Symposium on Operating Systems Principles, 1997
13. C. Bartolini, C. Preist, N.Jennings: A Genric Softwrae Framework for Automated Negotiation, AAMAS'02, Bologna, Italy, July 2002