A MIDDLEWARE INFRASTRUCTURE FOR MOBILE SERVICES BASED ON AN ENTERPRISE SERVICE BUS

Mobile Service Provisioning as SOA-Integration Problem

Michael Decker and Rebecca Bulander
Institute AIFB, University of Karlsruhe, Englerstr. 11, 76128 Karlsruhe, Germany

Keywords: Mobile and wireless Computing/Services, Service Oriented Architecture, Enterprise Service Bus.

Abstract: A middleware platform especially designed for the provisioning of data services for handheld computers using wireless data communication (e.g. smartphones, PDAs) has to offer a variety of different features. Some of these features have to be provided by external parties, e.g. billing or content syndication. The integration of all these features while satisfying mobile-specific challenges is a demanding task. In the article at hand we thus describe a middleware platform for mobile services which follows the idea of an Enterprise Service Bus (ESB). An ESB is a communication backbone for a Service Oriented Architecture (SOA) based on asynchronous communication with integration capabilities like transformation or intelligent routing. Because of its specific characteristics an ESB is an appropriate fundament for mobile service provisioning.

1 INTRODUCTION

In the last ten years we witnessed the extraordinary success of the wired internet. So far this success couldn't be repeated in the wireless world despite the high penetration rates of mobile devices and the availability of wireless data communication provided by cellular networks (e.g. GPRS, EDGE, UMTS).

One of the reasons for the lack of success of mobile data services is that the large variety of services which drove the wired internet to success isn't available in the wireless world. This originates from the fact that offering a data service for mobile devices is much more difficult than offering a service in the wired internet: there is a much bigger heterogeneity with regard to mobile devices' capabilities, e.g. what kind of data formats they can process or the quality of the displays. Due to the limited size of mobile devices they have only small displays and data input is cumbersome; that's why it is necessary to make use of context information like current location, time or profile information to support the user when interacting with the mobile device. Features like detecting the device's position, sending push messages (SMS/MMS), identifying the user or the kind of device used and charging fees (billing) can only be supplied by the respective mobile network operators (MNO). But MNOs won't grant access to these functions to everyone. Because of these hurdles only large firms with telecommunication know-how are able to offer mobile data services nowadays.

In literature many descriptions of technical solutions like frameworks or middleware for making it easier to develop and operate mobile services can be found. The novel approach of the middleware presented in the paper at hand is that the provisioning of mobile services will be considered as integration problem within a Service Oriented Architecture (SOA). The basic idea of SOA is to loosely couple the components of a distributed system (Keen et al., 2004; 37ff; Kaye, 2003): some components of the system provide functionalities called "services" and other components consume these services. The service provider publishes an interface description for each service offered. The most prominent approach for the realization of SOA is the use of web services. But SOA is also a management concept which postulates not to try to build monolithic business applications but a distributed system whose components offer reusable services which can be easily plugged together to implement the IT-support for business processes.

To integrate all the services required for the provisioning of mobile services we resort to the concept of a so called Enterprise Service Bus (ESB). An ESB is a special middleware for SOA that is based

Decker M. and Bulander R. (2007).
A MIDDLEWARE INFRASTRUCTURE FOR MOBILE SERVICES BASED ON AN ENTERPRISE SERVICE BUS - Mobile Service Provisioning as SOA-Integration Problem.
In *Proceedings of the Second International Conference on e-Business*, pages 122-129
DOI: 10.5220/0002112501220129

on asynchronous messaging. We argue that the features provided by an ESB help to fulfil essential requirements for a "mobile middleware". Afterwards we describe an ESB-based architecture for mobile service provisioning which we are currently realizing.

This article mainly deals with architectural considerations and does *not* address the problem of service discovery, namely how end users can find appropriate services based on contextual information or semantic techniques. There are other groups working on these subjects (e.g. Chakraborty, 2001; Hamdy & König-Ries, 2006).

The remainder of this article is structured as follows: in the second chapter we introduce our comprehension of some basic concepts required for the understanding of this article; afterwards we discuss the advantages of messaging oriented middleware for gluing together the infrastructure required for mobile services. In chapter three we describe features that should be provided by a middleware platform for mobile services. Chapter four describes a mobile-specific ESB which is exemplified by an example given in chapter five. We conclude with a brief discussion of related work in chapter six and give a summary and outlook in chapter seven.

2 BASIC CONCEPTS

2.1 Platforms for Mobile Services

In figure 1 we depict a simple reference model for mobile service platforms: we make the assumption that there is some kind of client software on the mobile device, e.g. thin client (only rendering function, e.g. web-browser for XHTML/WML/ cHTML-based services), rich client (browser capable of some business logic like validation of user entries), medium client (major part of business logic but only data caching) or fat client (autonomous data management). This assumption is a weak one since even the reception of SMS messages requires some kind of client software embedded in the mobile device's firmware. The other three parties involved in the provisioning of mobile services are stationary:

Service providers: They use the middleware to offer the actual services (end-user services), e.g. a location-aware newsticker.

Third party providers: Their services may be required by the end-user service providers. Third party services are "raw" and are not intended to be directly accessed by end users, e.g. a service to query weather data in machine-readable form.

Middleware: Middleware is software that mediates between the different software-components in a distributed system. The paper at hand deals mainly with architectural considerations of this component and which features it should provide and integrate.

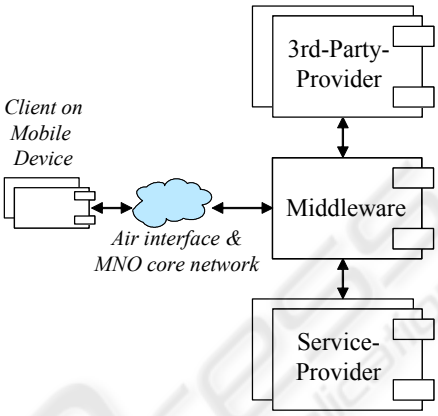

Figure 1: Reference model for mobile service provisioning.

Services provided by the middleware itself are called internal services; services provided by third parties or services for end-users are external services. All communication between the mobile device and the service provider is routed over the middleware.

2.2 Messaging Oriented Middleware

One can distinguish two fundamental paradigms to implement interaction with a remote service: synchronous and asynchronous communication. Using synchronous communication a request is sent to the remote service which tries to send back a response as soon as possible. This might be a convenient way from a developer's point of view since it mimics the way local methods/functions/procedures are invoked in a programming language but is not always adequate for the realization of a SOA: The caller has to wait (block) till the response arrives. If the server fails to deliver a response within a certain time span (e.g. temporary failure of wireless communication, high load of requests) this may even impair the calling process.

The asynchronous approach of messaging oriented middleware (MOM) is different: using this communication model a *producer* formulates a so called *message* which is a self-containing collection of arbitrary data and passes it to the MOM along with a specification of a *queue* (point-to-point messaging) or a *topic* (publish-subscribe messaging). For point-to-point-messaging each message can only be read by exactly one consumer, for publish-

subscribe each message can be consumed by an arbitrary number of consumers.

Asynchronous messaging is an adequate fundament for a middleware for mobile services because it has the following features:

Push capability: A service can proactively deliver information to a consumer without being directly requested (polled) to do so. This enables the implementation of push services at user level, that means services that can proactively delivery (maybe time critical) information.

Scalability: By looking at the current penetration rates of mobile devices it is obvious to conclude that the market for mobile services is a mass market. A middleware for mobile services thus should be able to handle large amounts of service requests. MOM makes it easy to fulfil this requirement: for services with high load levels one can simply add another service provider that helps to process the messages in the respective queue.

Information syndication: The publish-subscribe mechanism is well suited for the distribution of one piece of information (message) to a lot of different consumers; this problem with regard to mobile middleware has to be solved for the syndication of content or public context information. The content provider's server hasn't to be a strong machine because the MOM takes the burden of distributing the information to a potentially big number of consumers.

Reliability: Asynchronous communication helps to make the system more reliable: if a service provider or a (wireless) network connection fails the MOM stores the messages and retries to deliver them some time later.

Loose coupling: MOM constitutes a way to loosely couple the different parties involved in a SOA. This is important because the participating service providers will often change.

There are various vendors offering MOM products based on proprietary standards. In the Java-World the Java Messaging Service (JMS) defines a standardized API for MOM and several implementations are available, e.g. OpenJMS or JORAM.

2.3 Enterprise Service Bus

The term "Enterprise Service Bus" (ESB) was originally introduced by a consulting firm but meanwhile it is used for a certain kind of integration software based on MOM (Chappel, 2004; Keen et al., 2004; 74ff). With the Java Business Integration specification (JBI, see Java Specification Request No. 208) there is a standardization attempt for ESBs in the Java world.

In first generation SOA implementations consumer and providers of services communicate in a peer-to-peer fashion but this leads to a "brittle" system, because when one provider fails or has to be replaced all other components depending on him have to be altered. In second generation SOA approaches an ESB is used as hub between service consumers and providers. The ESB is not just a hub but it also provides features necessary for integration of different services in the sense of a SOA. The minimum functions an ESB has to provide are transformation of messages, intelligent routing (e.g. based on its contents a message is sent to a certain service provider) and adapters to connect services via different protocols like web services, Corba, RMI, SMTP/POP3, FTP or even proprietary protocols. Additional features are authentication or orchestration of services.

Technically an ESB consists of federated MOM-servers which share configuration and control. There are internal services (e.g. transformation, routing, and authentication) and the external services of the SOA which are connected using the adapters. From the MOM's point of view internal as well as external services are so called endpoints.

It is also possible to incrementally extend an ESB by adding segments. For the case of an ESB as "mobile middleware" this might be interesting if a company wants to operate mobile services that are only accessible by employees, e.g. access to groupware or ERP for mobile sales force.

2.4 Document-Style versus RPC-Style

Synchronous middleware communication typically (but not necessarily) employs Remote Procedure Calls (RPC): the caller's request is targeted at a certain address/procedure name and contains optional parameters; the response may contain return values. RPC-style might be sufficient for simple communication between a service consumer and a service provider but has the following drawbacks:

- If the required communication pattern in a distributed system can not be mapped to simple "request-response" RPC-style is not appropriate; this is the case when three or more entities have to collaborate.

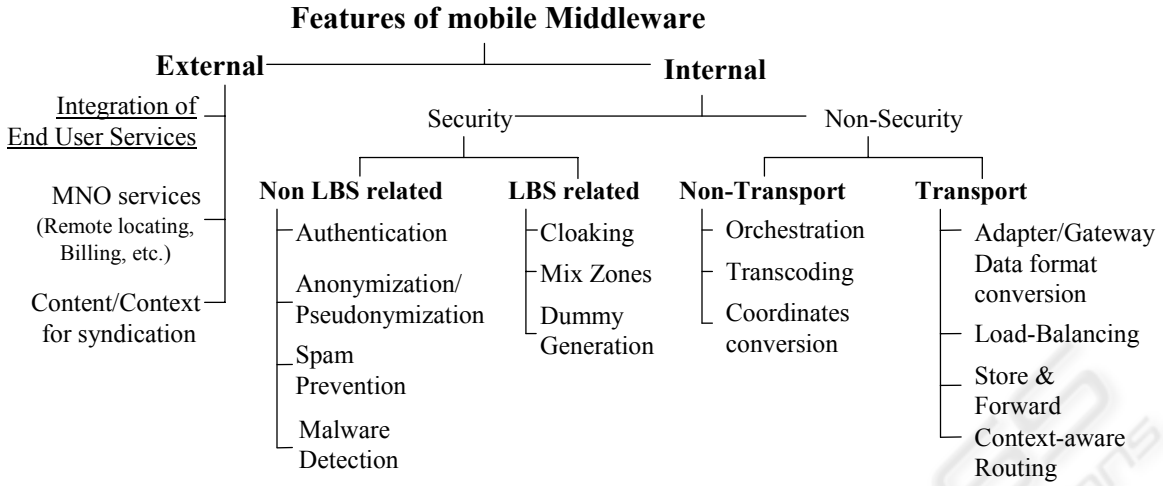

Figure 2: Features required for mobile services provisioning.

- When an entity receives a RPC request it has to find out to which session the request belongs to.
- One cannot use the full versatility of XML while using RPC.

Therefore asynchronous middleware typically makes use of document-style messaging (Kaye, 2003): in this mode the data is passed in form of self-contained messages between the collaborating entities. Each entity only looks at the parts of the message that it needs for its duty and has only to understand these.

3 FEATURES

In this chapter we describe the features that a middleware for mobile services should provide and sketch how they are supported by the ESB. Not all of these features are required for each service. We categorize these features into two main groups (figure 2): those which can be provided by the middleware itself (internal services, operated in the same administrative domain like the ESB) and those which have to be provided by external parties (external services). The internal services can be further divided into security-related and non-security related; for the external services we can distinguish end-user services and third-party services.

3.1 External Services

The most important type of external services to be integrated by the ESB are the actual end-user services of course, e.g. a firm that wants to offer a location-based information service. For the provisioning of these services it might be necessary to resort to external services provided by 3rd-parties. These 3rd-party services aren't directly used by the end users but are needed as "building-blocks" for the actual end-user services. One group of such 3rd-party services are services that have to do with functionalities controlled by the MNO. We call these services "MNO services". Examples are:

Billing means to pay for a telecommunication-related service over a telephone bill.

Remote Locating: When mobile devices are not equipped with a self locating module (e.g. GPS-receiver) it may be necessary to resort to remote locating techniques (e.g. CellID, TDOA[1]) offered by external providers. These techniques offer sufficient precision for many (but not all) thinkable services, e.g. localized news or weather forecast.

Querying Device's Profile: The MNO knows about some of the characteristics of the mobile device, e.g. display size and supported data formats.

Call control: The article at hand is about data services but some data services might need the ability to initiate voice telephony sessions (e.g. "call that hotel"-button in a location-aware hotel-finder service).

There are standardized gateways and APIs (e.g. OSA Parlay(X), see *www.parlay.org*) which are intended to enable MNOs to offer access to such functionalities to independent service providers but

[1] CellID: the location of the mobile device is determined by the coordinates of the base station used; TDOA: Time Difference Of Arrival, location of device is determined by runtime difference of radio signal between several base stations.

MNOs can't or don't want to negotiate with the large number of service providers interested in offering mobile services.

Content/Context syndication: The ESB also integrates third party services that offer content for syndication; this content is used by the client services to enrich their services, e.g. news or weather data. If the content is machine-understandable (e.g. temperature in a certain city) it can be used to realize context-aware services. According to our comprehension Context (in the sense of mobile computing) is information that can be used to support the user when interacting with a mobile device or service. One way to support a user in this sense is to display only data deemed as relevant by evaluating context information, e.g. a location-aware tourist guide shouldn't recommend a visit to the nearest swimming pool when it is raining all day long.

3.2 Security Related Features

We first discuss the following security features that are not mobile specific:

Authentication: The ESB has to ensure that every message it transports originates indeed from the party that is stated inside the message. To realize this each endpoint connecting to an external service has to require authentication from the service provider before accepting or delivering messages to/from that endpoint.

Anonymization/Pseudonymization: For many services it isn't necessary that the service provider knows the identity of the end user or can correlate different request, e.g. for general information services. To implement this feature the transformation capability of the ESB can be used: the ID-token in the request-message is encrypted before forwarding the message to external parties and decrypted just before the ESB has to find out to whom he has to dispatch the message. In contrast to anonymization for pseudonymization the ciphertext of a given ID has to be the same between several requests so the service provider can recognize users by the pseudonym.

Spam prevention: "Spam" denotes unsolicited push-messages. Because users carry their mobile device with them most of the day we can realize many useful proactive services dealing with time-critical information like reminders or alerts (e.g. stock quotes, new messages, failure of a technical facility, inventory level) using push messages. But an unsolicited push-message would be much more annoying on a mobile device than on an ordinary desktop computer. So there is the great fear that the spam wave known from the email-system spills over to mobile services. Therefore the ESB should help to avoid spam: it could provide a spam filter like those for emails (e.g. Bayesian filtering) and check a push-message before delivery; if the message indicates evidence of being unsolicited (e.g. by looking for ominous words) it would be blocked. Since filters are an essential component of ESBs the integration of such a feature is trivial but these kinds of filters are prone to produce false positives and thus relevant messages could get lost. Therefore it's preferable that the ESB adds a special token to a request message before forwarding it to the service provider. The service provider has to include this token if he wants to publish a push message to the end user over the ESB. The ESB will only dispatch a push message to an end user if a valid spam token is included.

Malware filter: The term "malware" subsumes software that is intended to cause harm to a computer system, e.g. viruses, worms, Trojan horses or spy ware. Meanwhile malware targeted especially at mobile devices was discovered, e.g. *Cabir*, *Pbstealer.A* or *Brador*. Therefore we use the ESB's filtering capability to check messages for such malicious content before they are delivered to the end user.

There are also security requirements that are specific to mobile services that make use of location awareness. An external service provider receiving a request could find out about the identity of the originator by analysing the included location information: e.g. it might be the location of a private estate so he could infer that the requestor is the owner of that estate; it might also be possible to correlate requests from different sessions that came from the same place or involved the same spatial-temporal sequence (Bettini et al., 2005). In many countries there are a lot of cameras to supervise public places (and some of them are even equipped with face recognition capability); if we learned that Alice was at a certain place and at the same time a service request containing the coordinates of that place was received by a service provider we can uncover the pseudonym used by Alice (observation attack). In literature several approaches to ensure location privacy can be found and these can be integrated by the ESB:

Cloaking means to deliberately reduce the precision of the location information. (Gedik & Liu, 2005).

Mix zone: If a user is within a defined mix zone the ESB won't forward any location information and will assign a new pseudonym so a service provider

will lose the user's trace (Beresford & Stajano, 2003).

Dummy request: There is also the idea of generating dummy requests with changing locations to confuse a potential attacker (Kido et al., 2005).

3.3 Non-security Related Internal Features

We identified the following features an ESB can provide by itself that don't have to do with security issues. The first group of such features has to do with the transportation of messages to or from an endpoint:

Adapter/Gateways: The external parties to be integrated may use different protocols and data formats. ESBs are an advancement of integration brokers and therefore the endpoints can host adapters or gateways to communicate with external parties using various protocols and data formats.

Load-Balancing: The ESB can use its routing capability to realize some kind of load balancing if for a certain service there are a lot of requests and there are several physical external service providers. Since the market for mobile consumer services is a mass market (more than one billion cellular phones worldwide) this is an essential feature.

Store & Forward: A typical problem of mobile services is that one has to assume temporary connection dropouts or switched off devices. This requirement is covered by the underlying MOM's store & forward capability.

Context aware routing: Depending on the user's current context it might be reasonable to route a service request to a different service provider. One example is a user interested in special offers for clothes: if he is in city X the ESB can forward the request to a service provider offering such a service for city X. Another example of context aware routing are service providers offering event guides targeting at different age groups; the ESB can forward the request to the most appropriate service provider based on knowledge of profile information.

The non-transportation related features are the following:

Orchestration: For each service request the ESB has to know which internal and external services in which order he has to invoke. Example: the first request in a session might require the message routed through a remote locating service for obtaining the requestor's location. Therefore the ESB should possess a simple workflow module as internal service which can decide to which endpoint a given message has to be routed next; this workflow module should be stateless (all state information is included in the message) so we can easily operate more than one instance if a lot of requests have to be handled.

Transcoding: Because of the great heterogeneity of mobile devices it might be necessary to transcode the content of a message before delivering it to the end user, e.g. to reduce size and/or colour depth of images or to transcode documents formulated in generic markup languages to specific formats like cHTML or XHTML-MP. This transcoding can be easily realized as filter feature within an ESB.

Coordinate conversion: there are many different ways to express the location of a user's device, e.g. Cell-ID, latitude/longitude with Bessel or WGS84 ellipsoid, Cartesian or ellipsoidal coordinate systems. Therefore services for coordinate conversion are required.

4 AN ESB AS MIDDLEWARE FOR MOBILE SERVICES

As argued in chapter two an ESB has many characteristics that make it a good choice as communication backbone for a mobile service infrastructure and all the features mentioned in chapter three can be easily integrated. In this chapter we therefore sketch an ESB-based infrastructure for the provisioning of mobile services (see figure 3).

To connect internal or external services with the ESB one has to make use of a logical endpoint. A logical endpoint may contain a service container with an internal service like pseudonymization or a connector to communicate with an external service.

The client component on the mobile devices interacts over gateway endpoints with the ESB: this might be an endpoint for a medium or fat client application using web services or another way of communication with the ESB. For thin client services the gateway has to act as web server receiving requests from a mobile browser application and sending responses in form of markup documents (e.g. XHTML-MP, WML). There are also gateways to send push messages, e.g. SMS/MMS or TCP/IP push channel.

An ESB uses MOM with document-style messaging as communication backbone: for each logical endpoint there is one queue for incoming messages. When a service component has to send a message to another logical endpoint or finished the processing of a messaging it puts the result message in the

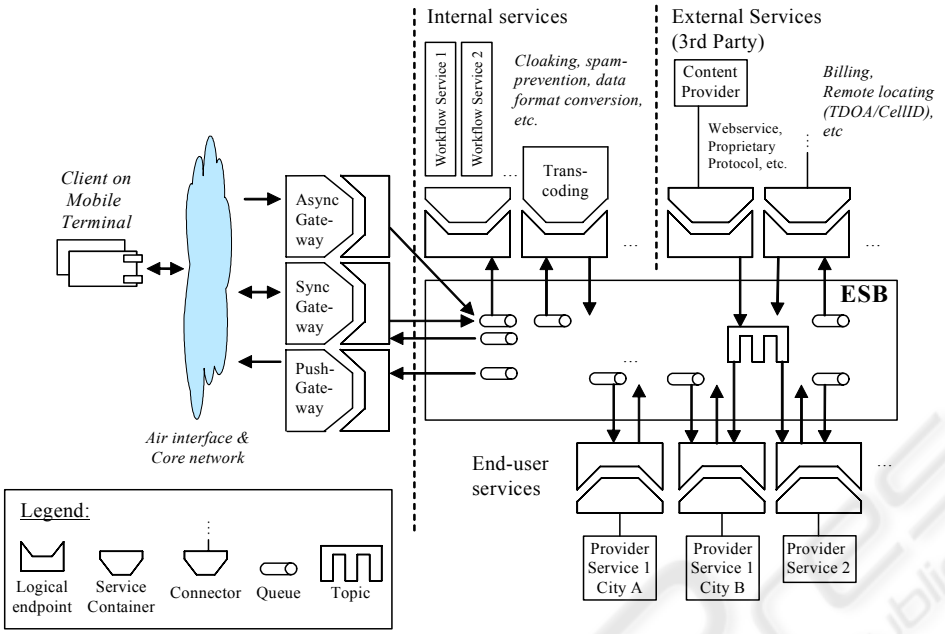

Figure 3: Enterprise Service Bus for mobile services.

queue for the workflow module which will decide to which abstract endpoint the message has to be forwarded. If there is information that has to be distributed to a lot of endpoints (e.g. content syndication) a public/subscribe topic is used. For communication between internal services the messages shouldn't be serialized to XML since serializing and deserializing is an expensive operation.

5 EXAMPLE SERVICE

To exemplify the concept of the ESB-based infrastructure we consider "location based news delivery" as example service: we assume a client component on the mobile devices that sends an SOAP-message to the ESB's gateway to start a subscription concerning bargain offers. Before the ESB can forward this request to the respective service provider it has to be routed through several other internal services:

- The message has to be validated and authenticated.
- The user has to be billed.
- Relevant information from the user profile (e.g. age, preferred language) and the position obtained from a remote locating provider have to be added to the message.
- The User-ID has to be replaced by a unique pseudonym.
- To prevent spamming an internal service adds some "spam tokens" to the message.

Now the message can be dispatched to the external provider which offers the service for the region where the user currently roams. If this provider some time later gets aware of an interesting offer matching the user's profile he formulates a message including one of the spam tokens and submits it to the ESB. The ESB again has to route this message through some of its services before it can be delivered to the end user's device eventually:

- The presence of a valid spam token has to be checked otherwise the message will be discarded.
- The content of the message has to be transcoded according to the device's capabilities, e.g. size of images.
- The pseudonym has to be resolved so that the ESB can find out to whom the message has to be dispatched.
- For thin client services it might be necessary to convert the message to an appropriate markup language, e.g. WML, cHTML or XHTML-MP.

6 RELATED WORK

There is some work that deals with SOA in mobile computing: Duda, Aleksy & Butter (2005) discuss how discovery, extraction, choice and invocation of services can be distributed between fixed and mobile

components when integrating mobile devices into a SOA. Others publications address the problem of how to discover appropriate services in mobile scenarios or the peer-to-peer-style collaboration of mobile devices (see the overview article by Hamdy & König-Ries, 2006). However, as far as we know there is no work dedicated especially to the backend integration problem of a "mobile SOA".

Since the most prominent implementation approach for SOA is the use of web services there are articles which propose solutions how to consume web services with mobile devices despite their limited resources, e.g. Sanchez-Nielsen et al. (2006). Adacal & Bener (2006) discuss several approaches to tackle performance issues that typically occur when accessing web services with mobile devices because web services make extensive use of XML and XML is notorious for being bloated. Their solution approach is based on the use of software agents; other approaches include the use of proxies or compressed XML.

The appropriateness of using asynchronous middleware for mobile services is a well-accepted notion; there is even a JMS-implementation for mobile devices (see *www.jtom.de*). But in our approach we don't assume that mobile devices necessarily consume/produce the middleware's asynchronous messages directly; rather we concentrate on asynchronous message exchange between the backend services required for mobile service provisioning.

There are many hosting solutions for mobile service provisioning: ASPF (Karlich et al., 2004) and WASP (Koolwaaij & Strating, 2003) provide many features but they don't describe a particular integration approach.

7 SUMMARY AND OUTLOOK

We argued that building a comprehensive middleware platform for mobile service provisioning leads to an integration problem: a lot of different external and internal services have to work together. To "glue" all these services together in a flexible or loosely coupled way we proposed a middleware infrastructure in the form of an Enterprise Service Bus (ESB). An ESB is a sophisticated communication backbone based on asynchronous messaging to connect the participants in a Service Oriented Architecture. It was explained why an ESB is an appropriate choice for the implementation of a middleware for mobile services. Afterwards we sketched an infrastructure for the provisioning of mobile services based on an ESB.

We are currently engaged in the implementation of a proof-of-concept prototype of the ESB presented in this paper using J2EE-technologies.

REFERENCES

Adacal, M. & Bener, A., 2006. Mobile Web Services: A new Agent-Based Framework. *IEEE Internet Computing*. 10(3), 58-65.

Beresford, A., & Stajano, F., 2003. Location Privacy in Pervasive Computing. *Pervasive Computing*. IEEE, 2(1), 46-55.

Bettini, C., Wang, X.S., & Jojodia, S., 2005. Protecting Privacy against Location-Based Personal Identification. In *Proceedings of the Conference on Secure Data Management (SDM '05)*, Trondheim, Norway, 185-199.

Chakraborty, D., Perich, F., Avancha, S., Joshi, A., 2001. DReggie: Semantic Service Discovery for M-Commerce-Applications. In *Workshop on Reliable and Secure Applications in Mobile Environment*. New Orleans, LA, USA.

Chappell, D. A., 2004. *Enterprise Service Bus*. O'Reilly Media, Sebastopol, CA, USA.

Duda, I., Aleksy, M. & Butter, T., 2005. Architectures For Mobile Device Integration into Service-Oriented Architectures. In *Proceedings of the International Conference on Mobile Business (ICMB '05)*, IEEE, Sydney, Australia.

Gedik, B., & Liu, L., 2005. A Customizable k-Anonymity Model for Protecting Location Privacy. In *Proceedings of the 25th International IEEE Conference on Distributed Computing Systems (ICDCS '05)*, Columbus, Ohio, USA, 620-629.

Hamdy, M., & König-Ries, B., 2006. Service-Orientation in Mobile Computing: An Overview. In *Proceedings of the MDM '06, IEEE, Washington*, D.C., USA.

Karlich, S., et al., 2004. A Self-Adaptive Service Provisioning Framework for 3G+/4G Mobile Applications. *Wireless Communications*, IEEE, October 2004, 48-56.

Kaye, D., 2003. *Loosely Coupled — The Missing Pieces of Web Services*. RDS Press, California, USA.

Keen, M. et al., 2004. *Implementing an SOA Using an Enterprise Service Bus*. IBM Redbooks, 2004.

Kido, H., Yanagisawa, Y., & Satoh, T., 2005. An Anonymous Communication Technique using Dummies for Location-based Services. In *Proceedings of the IEEE International conference on Pervasive Service 2005 (ICPS '05)*, Santorini, Greece, 88-97.

Koolwaaij, J., & Strating, P., 2003. Service Frameworks for Mobile Context-aware Applications. *Proceedings of the Conference on e-Challenges*, Bologna, Italy.

Sanchez-Nielsen, E., Martin-Ruiz, S. & Rodriguez-Pedrianes, J., 2006. Service-Oriented Architecture to Mobile Phones. In *Proceedings of the International Conference on E-Business (ICE-B 2006)*, INSTICC Press, Setúbal, Portugal, 57-62.