Bridging the Gaps towards Structured Mobile SOA

Apostolos Papageorgiou
Technische Universität Darmstadt
Multimedia Communications Lab-KOM
Merckstr. 25, 64283
Darmstadt, Germany
Apostolos.Papageorgiou@kom.tu-darmstadt.de

Bastian Leferink
B2M Software AG
Technologiepark Karlsruhe
Emmy-Noether-Str. 9, 76131
Karlsruhe, Germany
b.leferink@b2m-software.de

Julian Eckert
Technische Universität Darmstadt
Multimedia Communications Lab-KOM
Merckstr. 25, 64283
Darmstadt, Germany
Julian.Eckert@kom.tu-darmstadt.de

Nicolas Repp
Technische Universität Darmstadt
Multimedia Communications Lab-KOM
Merckstr. 25, 64283
Darmstadt, Germany
Nicolas.Repp@kom.tu-darmstadt.de

Ralf Steinmetz
Technische Universität Darmstadt
Multimedia Communications Lab-KOM
Merckstr. 25, 64283
Darmstadt, Germany
Ralf.Steinmetz@kom.tu-darmstadt.de

ABSTRACT

As the principle of service-orientation is gaining ground in a significant number of emerging solutions, the definition of standards coupled with the appearance of new architectures bring a new era in software development. New technologies enable the usage and composition of services in mobile environments, such as PDAs or common cell phones, thus paving the way for “mobile SOA”. However, the lack of middleware and technical solutions and the limitation of SOA appliance only within enterprise systems have forced mobile SOA to remain primitive. Mobile services are being rather used with mashup techniques.

In this paper, we investigate in three steps how to move from “unregulated mobile service mashing-up” to “structured mobile SOA”. First, we give a roadmap, analyzing critical mashup issues. Then, as a proof of concept, we describe a novel application, named “Services To Go!”, which, based on the proposed roadmap, provides a mobile client with various capabilities, such as map-based services, location-based services, routing services, social networking services, and more. In the third and last step, we use the “Services to Go!” experience to introduce a proposal for bridging the gaps between current mashing-up and structured mobile SOA.

Keywords

1. INTRODUCTION

A wide range of issues have to be investigated before mobile, resource-constrained devices become part of promising and efficient Service-Oriented Architectures (SOA). However, the special features (mobility, localization) and the people-centric usage of such devices in everyday life present totally new perspectives in innovative application development.

It may have been because of some standards, specifications and prototypes (as they are described in [15]) that mobile devices became SOA-capable, but parallel research on the fields of service mashups and location-based services is also important in order to drive mobile SOA to the most compelling functionality attainable. Concerning the aforementioned issues, [5] can give an overview of aspects of service mashups that concern us, while [2] covers important issues of location-based services and their future in mobile application development.

Mashing-up is sometimes conceived as an alternative solution to SOA. However, as argued in [14], Web 2.0 concepts such as mashups can be perfectly converged with SOA concepts towards the vision of an *Internet of Services*. The meaning of service mashup as we understand it, and as implicitly defined in [5], supports the understanding of it as a part of an SOA.

Section 2 refers to related work on the different issues that we are concerned with and discusses our contributions. In Section 3 we focus on mashup issues that are critical for mobile SOA and in Section 4 we examine further issues that make mobile SOA special, efficient, and promising. The Sections 3 and 4 together comprise a roadmap for the usage of mashups in mobile SOA, which in turn is the basis of the roadmap for identifying the gaps towards structured mobile SOA. Based on these analyses, our reference application is described in Section 5, with the description being limited to the architecture, exploitation of services and capabilities. Based on the insights gained in Sections 3, 4 and 5, Section 6 concretizes particular gaps in mobile SOA enabling technologies and describes our vision and our first steps towards bridging them. We conclude our paper with a summary in Section 7.

2. RELATED WORK AND OUR CONTRIBUTION

A good description of what a Web service mashup is and how it differs from Web service compositions can be found in [5]. According to it, service mashups differ in that they alleviate the burden of service composition with strict regulations and patterns and they are therefore more easily implemented by less specialized developers. Interesting mashup modeling approaches and execution platforms are investigated in the mentioned paper, as well as in [4]. A contribution of our current paper to such research work is the identification of mashup issues that are critical when it comes to mobile SOA, and the formulation of a corresponding roadmap.

Concerning mobile SOA, [15] gives an overview of developments that are necessary in order to enable Web Service consumption in mobile environments. To this end, we offer a roadmap application and an example of usage of the corresponding J2ME specifications (JSR-172) mentioned in [15] and [8]. Lightweight mobile SOA architectures [8] have become more and more important, with keeping traffic and hardware costs to a minimum being a critical issue. We contribute by describing key aspects of our fully-functional, efficient, map-based, multiple-service mobile application, which requires less than 300 kilobytes of memory.

Still, our main contribution is the identification of gaps in the mobile SOA world and our proposal for bridging them in a way that we can smoothly move from the mashing-up of mobile services to really structured mobile SOA, without losing the advantages of current techniques. The details of this vision become clearer in the last sections. The conclusions made from this research work led to the kick-off of a new research project, goal of which is to further investigate the identified gaps and design/implement solutions for them. More related work concerning such gaps will be referenced in Section 6.

3. SERVICE MASHUP TECHNIQUES

The term *Service Mashup* is normally used to indicate the result of the gathering of a set of services, when this result has been achieved after the application of a variety of design and implementation decisions and techniques. Mashups do not imply the existence of standard protocols or procedures, as *Service Compositions* [12] do. Still, the research on relevant design procedures, implementation architectures or even tools (as in [4] and [5]) can help transform mashing-up into a powerful, structured approach.

Critical issues of service mashups are presented now in such an order and in such a way, so that they can form:

- A roadmap for mobile SOA application development, (together with the issues of Section 4) and most importantly
- A basis for the identification of gaps that new mobile SOA middleware should fill in order to support or enable more structured mobile SOA development.

3.1 Principles and Design

Although service-orientation offers the possibility to flexibly add, replace, or subtract services at any time, a different and careful design phase is still very necessary. In addition to traditional application design tasks, special consideration of the steps for the design of the service mashup, as well as of some basic SOA design principles are needed.

As *basic SOA design principles*, we refer to tasks such as trade-off valuations and agreements with service providers, guidelines for the creation of new services, selection of a governance approach and more. Having applied a careful design process in our reference application, we keep it here short and do not go into details of each phase. The selection of a careful design process among the numerous proposed approaches must match the application needs. Showing the way, [9] and [12] offer interesting analyses of SOA design and governance.

As *service mashup design steps*, we refer to a chain of decisions such as the following, where each step must be well considered before heading for the next.

- Identify an abstract set of necessary services
- Find, examine, evaluate and test existing services that can be used to serve the application purposes.
- Select services that can be used, either independently or as parts for the composition of new services.
- Identify which new services and which compositions of new or existing services must be implemented.
- Identify and design all service relations, possibly by creating a mashup diagram or picture.
- Make sure that all needs of the initially abstract set are now fulfilled and when they are, proceed to implementing the designed mashup.

The goal of the overall procedure is to lead to the optimal co-application of the techniques that will be described in 3.2, 3.3 and 3.4.

3.2 New and Existing Services

Re-use of existing services is one of the most important benefits of the service-oriented paradigm. It is supposed to reduce the workload of software development to the composition of existing implementations. However, the vision is far from being achieved, especially when it comes to mobile SOA, and application-specific services must almost always be developed.

Coarsely categorizing, we can separate the services that we had to implement into those that fulfill new tasks and those that are related to existing services, by serving their composition, by being combined with them, or even enhancing them.

3.3 Service Compositions

The technique of service composition is more or less known and it is considered as one of the most important characteristics of SOA. Service compositions are part of the mashup but, as the “black box” principle (i.e. the loose coupling rule) is applied recursively, composite services can be depicted as entities inside a mashup. Then, they should also be separately depicted by means of the modules that compose them. At this point, it must be stated that compliance to the “black box” principle is unfortunately not always guaranteed. Experience has proved that it is not always easy to achieve, and that non-compliance to it can have disastrous consequences in the flexibility of a service-oriented architecture.

For example, a typical service composition in our reference application will be the usage of a routing information service (provided by [17]), a public transport information service (provided by [13]), and a city information service (provided by our application server thanks to data from many different sources) to compose a service which in turn takes advantage of location and time data (cf. Section 4) in order to offer intelligent routing or suggestions.

3.4 Service Wrapping

This is a less standard technique but has an upgraded importance in service-oriented systems. This happens because service-oriented systems often benefit from the existence of services that are older and may not comply with the newest needs or standards. Service wrapping may affect either single services or even more commonly composite services, and is engaged mostly because of one of two reasons:

- *Protocol wrapping* is the effort to either provide existing software as a (web) service or to take an existing service and provide it with a different protocol, e.g., SOAP from HTTP.
- *Functionality wrapping* takes place when existing services or compositions of services fit our needs but not in an optimal way or not with the exact functionality that we need.

Where and why the above described techniques can be applied and in what context they have been applied for the needs of our application may become clearer during the description of our system in Section 5.

4. ENHANCING SERVICES IN MOBILE ENVIRONMENTS

As mentioned in the introduction and indicated with an example in 3.3, mobile applications can take advantage of the nature of their hosting devices in order to use some powerful features. The term *people-centric computing*, more analytically dealt with in [1], is more often used in the context of sensor network applications to mean the exploitation of data that spring from the human user. This way, the application can serve needs that are really narrowly related to its user. Because of the nature of modern mobile applications, we believe that they can have many features that make them people-centric, if they are effectively exploited. The capabilities offered by such features, which will be discussed in this section, are in a way similar to the capabilities offered by sensed data. Thus, they could be further enhanced by the existence of sensed data, providing a complete people-centricity. But for now, the involvement of sensed data in our solution remains a subject of future work.

Summing up the mentioned features, mobile devices accompany the user and contain data that are to be used personally (personalization). They follow and possibly record the actions of the user or events in his/her environment (context-awareness) and they also provide location data (localization). [1] is also a good study of such features. In the following, we explain how they can be exploited giving examples on how they are part of our reference application.

4.1 Personalization

The term *personalization* is mainly used to refer to the ability of the user to set preferences in applications. When it comes to mobile environments, it has an even wider meaning and a close relation to personal data as well as ambient- and context-awareness, as [1] also explains. In cases that the application includes services that take advantage of personal data and location data, personalization is not only a matter of user preferences, but mainly a matter of privacy.

Typical examples in our application are options like activation/deactivation of location services, tracking services, social networking services, and services that use other personal data (e.g. calendar). In addition to user preferences, we also regard as personalization the relationship of the application with user's personal actions. For example, our application enables services such as direct calling or message sending to phones of contacts or places (hotels, restaurants etc.). Further thoughts include services for booking etc.

4.2 Context-awareness

Mobile applications can track user actions and environmental states in a different, more enhanced way than traditional applications can do. This can enhance the functionality of web services consumed within a mobile environment. For example, a routing web service can obviously be enhanced by location information of the consumer, while an information web service can be enhanced by taking into account previous choices, actions or preferences of the user. Such techniques, along with a route tracking service, are typical examples of context-awareness within our application. Such solutions can lead to improved or even intelligent services, stemming from the enhancement of common Web services - new or existing.

Web services that could take advantage of (or be enhanced by) sensed data should also be considered in this category, although our application leaves this as an open issue for future work. The vision of mobile devices as the companions of humans, especially of elderly people, has already initiated much research work. With approaches such as ours, Web services will play an important role towards fulfilling this vision e.g., for transferring sensed data through the web and performing remote calculations or information retrieval triggered by the sensed data.

4.3 Localization

Location-based services are considered separately because of their great importance and the vast research effort placed on them by the community, although they could be included in the context-awareness issue. Numerous studies ([2], [16]) deal with location-based services and platforms for their exploitation, while relevant applications have already appeared in scientific approaches and commercial products.

Another reason for special consideration is the high level of privacy that is normally dictated by the existence of such services. The best example of all, which is also present in our application, is the people-tracking case. Our application uses location data to enhance a social networking service, showing the current location of contacts on a map (if they have activated the corresponding service). Still, location data can enhance Web services in a more seamless and less concerning way. Automated calls of routing or information Web services and automated tips based on the user's current location are such cases.

5. OUR REFERENCE MOBILE SOA APPLICATION

This section presents our reference application and puts it all together, by showing how the discussed issues and techniques are applied in mobile environments with existing technologies and currently feasible architectures. We present a high-quality mobile client requiring very few kilobytes that exploits mashups made out of a variety of services. The "Services To Go!" client is an application that was designed to consume service mashups of heterogeneous services in order to provide the user with location-based, context-driven capabilities. Accordingly, the client includes numerous – though homogeneous – user interfaces that show these capabilities and constitute a multi-purpose companion application. Working on all these features is important for gaining insights that lead to the proposals of the next section.

Most of the functionalities are based on third party services that provide information ranging from map data [10] and routing details [17] to city information and restaurant reviews from a city magazine, as well as a social networking service. Depending on the desired functionalities and the available protocols, the services are either directly consumed or wrapped from our server, according to the techniques described in Sections 3.3 and 3.4. Specific application-oriented services are implemented and hosted exclusively on our server (cf. Section 3.2). Before we list the main categories of the exploited Web services (cf. Section 5.2) and present some more details about the client (cf. Section 5.3), we give a quick overview of the architecture in Section 5.1. In its generalized form, this can be seen as a roadmap architectural approach for mobile SOA solutions that are based on existing middleware and technologies.

5.1 Services To Go! Overview & Architecture

As emphasized in [15], the latest specifications defined for Java Microedition (J2ME) – namely the JSR-172 specification (J2ME API for Web Services) – lay the foundations for service-orientation in mobile applications. Exploiting JSR-172 capabilities along with capabilities of JSR-179 (J2ME Location API), we have pursued the enhancements described in Section 4. We have tested solutions both on devices that implement these specifications as well as on other devices. Of course, as stated in [15], the further enabling of service-orientation for mobile applications is a hot research issue and subject of future work.

Figure 1 is a coarse representation of the architecture. Our client is based on J2ME and uses the specifications mentioned earlier, as well as a mobile device database system (Record Management Store - RMS) and the device's Personal Information Management (PIM) system. At the bottom right, we see the connections to our Application Server, to the Birdseye Server (provider of the social networking service) and to the Third-Party Service Providers. We distinguish the Birdseye Server in order to introduce the notion of *internal* and *external* services. Although the Birdseye Service is not oriented to the purposes of our application, it is characterized as internal. This means that it lies inside the domain of the project, in the sense that the access to it reaches further than simple service consumption. The distinction between internal and external services plays an important role when it comes to the procedures of mashup design ([4], [5]) and SOA Governance [9]. Components of minor importance for the understanding of the architecture are not depicted.

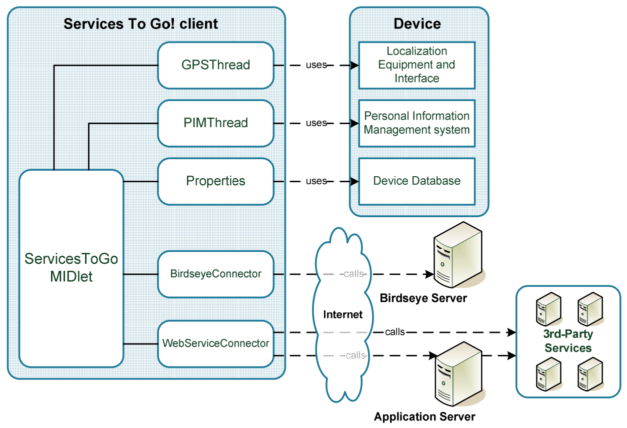

Figure 1. "Services To Go!" architectural overview

5.2 Web Services

Altogether, we can distinguish 3 different categories of services involved in our client:

- Navigation Services (Routing, Public Transport etc.)
- Information Services (Hotel, Restaurant, Sightseeing, Tours, Stations etc.)
- Social Networking Service (Birdseye)

In the following, we comment on them shortly, trying to give insight into how the techniques and ideas presented in previous sections can be exploited.

5.2.1 Navigation/Routing Services

Our Routing Service uses the wrapping and composition techniques in order to merge pedestrian routes with public transport connections. At the same time, it incorporates a context-driven logic in order to provide up-to-date and smart proposals and alternatives. While the current GPS position of the client is the only information stored on the device, requests for navigation services are sent based on geo-coordinates and handled as follows: The Application Server asks a third-party service [17] for geo-location information (country, city, street, number, descriptions, etc.). Then, context-based decisions are made combining this information with data that are either available or provided by other services, e.g., distance to nearest stations, and user preferences (personalization). If they have to be included, the corresponding (time-aware) public transport connections are sought through another provider [13], while pedestrian routing details are then offered again by a different provider [17].

5.2.2 Information Services

Information services can either be used in combination with other services (as described earlier), or they can directly serve user needs. Whatever the type of information and whoever the provider is, consistent representations and common attributes are used for their storage within the client. So, homogeneous (i.e. regardless of their type), user-friendly actions can be performed upon them, for example with the help of the map-based GUI (cf. Section 5.3).

5.2.3 Social Networking Services
The involvement of social networking services in mobile companions is a hot issue and our reference application was among the first applications to offer this novel capability and promote the research on this field. The Birdseye Server is currently under development by SAP Research. This server provides a Web API for geo-position tracking, message exchange, and contact position tracking. Every time XML-based messages for location updates, contact location requests or message submission/reception are exchanged with the Birdseye Server, status values (amount of unread messages etc.) are updated in the device memory, so that the social networking utility can be up-to-date and personalized. Additional types of requests are possible, for example, our contacts positions by a geo-range and/or activity within a given amount of time. Of course, for reasons of privacy and personalization, all localization features can be very easily turned off. Again, we will see in the next Section how these aspects can be brought further.

5.3 Client Implementation and Interfaces
Typical examples of the interfaces, indicating the location-based and context-driven nature, are the map screen and the next steps screen. The next steps interface allows for selecting services filtered by context values, like the current position or past user actions. Movement tracking and message utilities are two more key features.

5.3.1 Map-based Interface
The map is the basic interface of the client when it comes to exploitation of services from the end user. All main actions, known from existing navigating- and information-systems, are available using this interface. Namely, current location of the user (and of its contacts, if desired) is shown there. Any objects retrieved from heterogeneous services appear homogeneously on the map, while various user actions and functions like “route to”, “call”, “show details”, etc., are available regardless of the object’s type or “origin” (see also 5.2.2). Our map source is OpenStreetMap [10]. A sample screenshot can be seen in Figure 2.

However, our mobile SOA approach carries some great advantages in comparison to integrated navigators or personal companion applications:

- All data and information (including the maps) are retrieved from services and are not locally stored. This means very little need for memory and up-to-date information
- There is no single-purpose logic, which would limit extensibility. Our application already integrates many different types of services and it can easily integrate many more. Moreover, service updates or changes may not affect the client software.

5.3.2 Next Steps
The next steps interface provides the user with context-based options and proposals. While the location data are being updated, it is seamlessly being detected if the user is in “known” places, which can be predefined or user-tracked. Special rules then determine which options (i.e., services) will be proposed to the user each time.

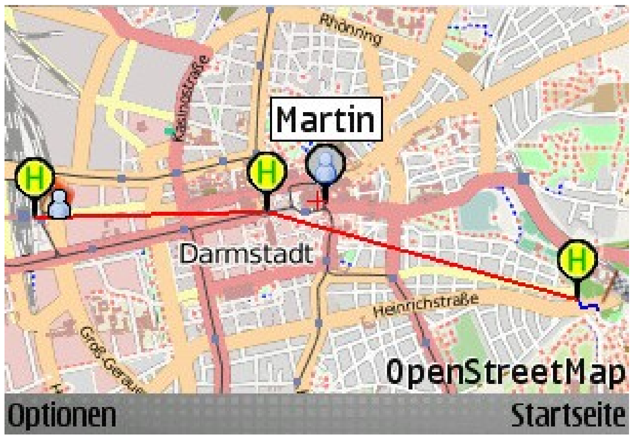

Figure 2. Sample screenshot of the Map-based interface

5.3.3 Messages
Services To Go! can send and receive messages of the social networking utility not only from within the Map-based interface, but also with a classical message interface.

5.3.4 Tracking
When the tracking utility is activated, positions and upcoming routing events are tracked. A use case is the combination of a tracking utility with routing services. The tracking utility can provide the user with alerts and reminders, for example, when entering a specific geographic zone or is running late for an event.

6. FACILITATING MOBILE SOA
After many recent “success stories” and detailed analyses in research and industry, service-oriented architectures have already proved to offer flexibility, scalability and reusability, due to the independency, interoperability and loose-coupling of the services [6].

Furthermore, the current state of the SOA world indicates that its evolution will lead to the so-called “Internet of Services” (IoS) with the help of service marketplaces. In this scenario, extended functionalities and capabilities of the supporting layers will let services be more accurately searched among all providers, automatically chosen with respect to quality and “matching degree”, and more easily and flexibly bound to the clients.

6.1 Identified Gaps
All these advantageous features and capabilities (of the current state, as well as of the IoS) are also wished for mobile SOA. However, a “trip” along the roadmap and the applied scenario that we presented in the previous sections, shows that existing technologies cannot save the developers from the following obstacles on their way towards structured, fully-capable mobile SOA:

- Services have to be manually bound to mobile clients, requiring varying effort with respect to the protocols that are supported at both client- and provider-side. Developers of mobile apps have to bother a lot about the communication protocols of the used services and often implement complicated parsing of the used message formats, because the libraries for service consumption at the client side are primitive. Even for Web Services, the most common enabling technology, their consumption from mobile clients is much more difficult than from workstations, as there are no standard packages, e.g. in J2ME, to facilitate this further. Even worse, many devices do not implement required JSRs for Web Services, making it much more difficult and possibly bandwidth consuming to communicate with the providers. A study of [15] can support the understanding of this particular issue. Another consequence of this manual binding is that, in practice, mobile services cannot be replaced by equivalent services once they have been bound. In "Services To Go!" we had to wrap many services (see Sec. 3.4) and then manually bind them, requiring great effort. We also had to reject certain devices, because of missing capabilities.

- Enhanced context-awareness for mobile applications cannot be guaranteed and implemented without complex or application-driven structures or modules. In our example, the social networking utility (Birdseye) and its interfaces had to be extended, and they can still offer limited context-awareness. For the capturing and the exchange of other context-information, much manual work was required. Such "heavy" modules, which often reduce scalability, have to be implemented in almost all solutions (e.g. [7]) that want to support enhanced context-awareness, i.e. mutli-user time- and location-based personalized information.
- The lack of ESB-capabilities (Enterprise Service Bus) and the importance of "value-added services" on mobile applications dictate new solutions for workflow handling. The modeling and the consumption of workflows (with possibly dynamic service selection) is currently not possible and even if the ongoing research on workflow engines for mobile devices [11] has the desired results, a lot needs to be done in order to have workflows that are being resolved optimally for mobile applications and are getting automatically enriched with context-information ("value-addition"). We have described how composition or mashing-up is now manually performed, still directly addressing fixed service providers and requiring additional effort for providing the resulting composition or mashup with an interface compatible with the mobile client.

6.2 The Mobility Mediation Layer

Although some research initiatives (e.g. [11], [15]) may lead to results that support mobile SOA, we are still far from letting developers easily exploit all SOA capabilities, as well as the Internet of Services, within their mobile applications. We present the Mobility Mediation Layer (MML), believing it is the most complete conceptual approach for supporting the participation of mobile clients in the Internet of Services.

The MML and its complementing environment is not simply a middleware to which a mobile application has to be bound. It is itself service-oriented and the services it offers aim to accompany and complement the existing solutions of SOA middleware and service marketplaces in order to make the latter exploitable in mobile environments and enriched with guaranteed context-awareness. In the following, we briefly describe its main parts, which are depicted on Figure 3.

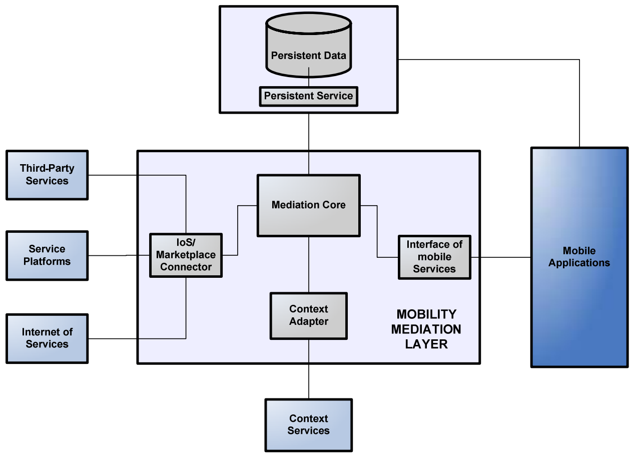

Figure 3. Concept of the Mobility Mediation Layer

The MML defines new specifications for its mobile interface, i.e. for the way it can be used from mobile clients so that the latter can flexibly "ask for" workflows or services that will be dynamically composed by accessing service marketplaces. This access will be provided by IoS/Marketplace Connectors. With respect to personalized features and other application-related data that will be available from many users and applications thanks to the Persistence Services, the dynamically selected services (or composed workflows) will be potentially enriched by being combined with Context Services. Context Adapters will be deciding when and how to perform this "intelligent" enrichment. Further aspects of these parts become clear in the following, where we describe how the problems of 6.1 can be overcome.

6.3 Bridging the Gaps

With respect to the identified gaps, we explain how the proposed layer faces the respective problems.

- For the consumption of services that either have during development unknown providers/endpoints (i.e. have to be dynamically bound), or are offered with protocols incompatible with the mobile client, or should be wrapped (see sec. 3.4) or dynamically enriched with context, the MML offers a new mobile interface. Corresponding J2ME libraries that simplify the usage of this interface will reside on the client-side. The development of these libraries will accompany the specification of the MML's mobile interface. Unlike the existing standards for messaging in SOA communication, these specifications will be designed particularly for mobile clients, giving a new opportunity to allow for lightweight communication, which is important for resource-constrained devices. So, for mobile applications on devices that do not support the JSRs for Web Service consumption and for mobile applications that cannot or do not want to bind particular services, the mentioned capabilities of the MML will allow for an easy development that lets MML undertake the difficult tasks. This does not mean, of course, that an MML should mediate every interaction of the application with the used services.
- As we also mentioned for the case of Birdseye, the handling of enhanced context-information is difficult and even after its implementation, it can cost a lot in terms of inter-

application communication. The support of context services as depicted in Fig. 3 has a vision that is in accordance with the IoS. Context-information like the location of things or people, events reported by users of various applications etc., will be available through independent services, which will in turn be easily accessible through the MML. So, the developer will be able to create context-aware mobile applications without caring much about the handling of the context (simple usage of existing context services) or even without bothering at all (automatic enrichment of mediated services with context-information).

- To avoid the selection of fixed service providers at design-time and to allow for easy workflow-handling, the MML will include workflow descriptions in the specification of its mobile interface and the client-side libraries will ease the programming of such workflows. So, the capability of workflow execution will not be transferred to the mobile device, but it will rather remain a "server-side" task. Still, the consumption of workflows at the mobile side will be eased by the mentioned specifications and libraries, while the workflow execution will take place at an environment such as this of the MML, so that every step of it will have extra capabilities, like the access to persistence services, context services and other MML mechanisms.

All in all, the MML can be seen as an extension or accompanying concept of the platforms that will comprise the basis for the IoS. As the MML is currently under specification and development, evaluations of its mechanisms are subject of future work. The contribution of this work is limited to presenting the results of the project that led to this new proposal and to a new project, funded by the German Federal Ministry of Economy and Technology, aiming to extend the vision of the Internet of Services.

7. SUMMARY

We performed a detailed study to find ways to bridge the gaps towards a more structured form of mobile SOA, which can exploit all the capabilities of the future Internet of Services. More specifically, we referred to the new specifications that are being developed to support mobile SOA, we handled the issues that distinguish current mobile SOA applications, we explained how these can be transformed from impediments to enhancing features, we presented a (novel) reference mobile application that integrates various services and SOA features based on existing technologies and we closed the work with our conclusions and our corresponding proposals for new solutions. A first insight into these new solutions was given, while their exact specification and evaluation is subject of future work.

8. ACKNOWLEDGMENTS

The project was funded by means of the German Federal Ministry of Economy and Technology under the promotional reference "01MQ07012". The authors take the responsibility for the contents. We would also like to thank all of our service providers, and in particular SAP AG and YellowMap AG for their support.

9. REFERENCES

[1] Arbanowski, S., Ballon, P., David, K., Droegehorn, O., Eertink, H., Kellerer, W., van Kranenburg, H., Raatikainen, K., Popescu-Zeletin, R. "I-centric Communications: Personalization, Ambient Awareness, and Adaptability for Future Mobile Services". IEEE Communications Magazine, vol. 42, issue 9, September 2004, pp. 63-69.

[2] Bellavista, P., Kupper, A., Helal, S. "Location-based Services: Back to the Future". IEEE Pervasive Computing, vol. 7, issue 2, April/June 2008, pp. 85-89.

[3] Berbner, R., Spahn, M., Repp, N., Heckmann, O., Steinmetz, R. "Heuristics for QoS-aware Web Service Composition". IEEE International Conference on Web Services (ICWS) 2006, July 2006, pp. 72-82.

[4] Blake, M.B., Nowlan, M.F. "Predicting Service Mashup Candidates Using Enhanced Syntactical Message Management". IEEE International Conference on Services Computing (ICSC) 2008, July 2008, pp. 229-236.

[5] Braga, D., Ceri, S., Daniel F., Martinenghi, D. "Mashing up Search Services". IEEE Internet Computing, vol. 12, issue 5, September/October 2008, pp. 16-23.

[6] Erl, T. "Service-Oriented Architecture: Concepts, Technology, and Design". Prentice Hall PTR, August 2005, ISBN: 0-13-185858-0.

[7] Hofer T., Schwinger W., Pichler M., Leonhartsberger G., Altmann J. "Context-Awareness on Mobile Devices - the Hydrogen Approach". International Hawaiian Conference on System Science, 2003.

[8] Kim, Y.S., Lee, K.H. "A Light-weight Framework for Hosting Web Services on Mobile Devices". European Conference on Web Services (ECOWS) 2007, November 2007, pp. 255-263.

[9] Niemann, M., Eckert, J., Repp, N., Steinmetz, R. "Towards a Generic Governance Model for Service-oriented Architectures". Americas Conference on Information Systems (AMCIS) 2008, Association for Information Systems, Toronto, ON, Canada, August 2008.

[10] OpenStreetMap Project, www.openstreetmap.org

[11] Pajunen, L., Chande, S. "Developing Workflow Engine for Mobile Devices". 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC) 2007, pp.279-286.

[12] Papazoglou, M.P., van den Heuvel, W.J. "Service oriented architectures: approaches, technologies and research issues". The International Journal on Very Large Data Bases (VLDB), 16(3), 2007, pp. 389–415.

[13] Rhein-Main-Verkehrsverbund (RMV), www.rmv.de

[14] Schroth, C., Janner, T. "Web 2.0 and SOA: Converging Concepts Enabling the Internet of Services". IT Professional, vol. 9, no. 3, May/June 2007, pp. 36-41.

[15] Tergujeff, R., Haajanen, J., Leppanen, J., Toivonen, S. "Mobile SOA: Service Orientation on Lightweight Mobile Devices". IEEE International Conference on Web Services (ICWS) 2007, July 2007, pp. 1224-1225.

[16] Xia, Y., Bae, H.Y. "General Platform of Location based Services in Ubiquitous Environment". International Conference on Multimedia and Ubiquitous Engineering (MUA) 2007, April 2007, pp. 791-795.

[17] YellowMap AG, Karlsruhe, www.yellowmap.com