Having Services "YourWay!": Towards User-Centric Composition of Mobile Services*

Raman Kazhamiakin[1], Piergiorgio Bertoli[1], Massimo Paolucci[2], Marco Pistore[1], and Matthias Wagner[2]

[1] FBK-Irst, via Sommarive 18, 38050, Trento, Italy
{bertoli,raman,pistore,traverso}@fbk.eu
[2] DoCoMo Euro-Labs, Landsberger Strasse 312, 80687 Munich, Germany
{paolucci,wagner}@docomolab-euro.com

Abstract. Mobile phones are becoming an essential tool in our life. They not only act as phones and media players, but more fundamentally they give us access to variety of services that we use in everyday life, including social networking, personal assistance, entertainment, travelling, and so on. Unfortunately, while the number of services increases, each service is narrowly directed to solve a specific user task with no attention to how the user may utilize these services in combination. At this stage, the combination of services and the integration of their information flows must be managed by the user on his own, in a handcrafted way.

To support the user in the composition of services and applications, we propose to organize the services around a small set of resources - time, location, social relations, money - which model the essential user assets handled by mobile services, and which guide the data integration and service composition.

In the paper, we discuss our current realization of such resource-based, user-centric service composition approach, detailing the underlying conceptual architecture and discussing the actual execution of the platform on a set of practical scenarios.

1 Introduction

Mobile phones are becoming an essential tool in our life. They not only act as phones and media players, but more fundamentally they give us access to variety of services that we need in everyday life. A simple catalogue of such services includes ones for travel activities (e.g., navigation and map services, ticket booking via mobile, SMS notifications of flight delays), social networking, personal assistance, and entertainment.

While the number of services increases, each service is narrowly directed to solve a specific need of the user with no attention to how services may work

* The research leading to these results has been partially funded by the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 213339 (ALLOW).

J. Domingue, D. Fensel, and P. Traverso (Eds.): FIS 2008, LNCS 5468, pp. 94–106, 2009.

together nor to how the user may utilize these services in combination. For example, even now there exist near-field services that allow us to get information about movies shown in a local theatre, Web services for booking movie tickets, and TelCo services for payment; we can store movie event in the agenda and set up a reminder for it; we can share the information about the movie with the people in our contact list using TelCo services, and finally use map services to route us to the theatre. In spite of the availability of these complementary functionalities, the user has to face the problem of their integration on his own. First, the user has to deal with different services, and consequently, with their specific interfaces, formats, and protocols. Second, the user has to manage by hand the composition of these services and information flow across them. Third, the user has to continuously ensure the consistency of the information used by different services: if the user is not able to go to the movie, he has to take care of propagating the effects of this decision by means of relevant services, i.e., removing the event from the agenda, sending an alert to the friends, cancelling the ticket on-line, and so on.

Ultimately, what is missing is a platform that enables user-centric delivery and composition of mobile services by satisfying the following requirements:

1. facilitate the user activities proactively identifying, interpreting, and addressing the user needs and constraints;
2. support and automatize the information integration so that the user does not have to manually manage and keep track of the data flow between the applications and services involved;
3. provide continuous support for the information consistency through coordinated integration and use of relevant operations and services;
4. foster the user control on the performed activities, requiring that no critical information is transmitted without involving the user, and no actions that are redundant or even harmful for the user are initiated by the platform autonomously;
5. abstract away the heterogeneity of the service implementations making transparent the differences between the protocols and data formats.

Working with the platform, the user activates various services and applications and makes decisions on the information to use and actions to perform in reaction to the results of this work. The role of the platform is to keep track of the data managed by different services and to guarantee its consistency. In this way, the platform has a coherent representation of the undergoing user's tasks and can react to changes in a coherent way by discovering, organizing, and representing the services in a manner understandable by the user, so that the user can make his decisions. In our example, the user decides when to book a ticket and which friends to invite, while the platform has to integrate and pass the relevant data to the payment, agenda, communication, and navigation services. Similarly, the user may decide to remove the event from his agenda, and the platform will take care of cancelling the ticket, notifying the friends, etc.

The key challenge here is how to relate and integrate different applications and services: they are created independently by different parties, represent different

application domains, and deal with particular aspects of the user activities. Crucial issue is to identify a set of concepts that are suitable for expressing both the user intentions and the service capabilities, and therefore to form the basis for their integration.

In this paper we present an approach to the user-centric composition of mobile services, where these concepts correspond to the core user assets, or "resources", namely time, location, money and social context. The user activities deal with allocation of such resources, and the key organizational issue for the user is to manage his activities in a way to make a consistent use of his own resources. In practice, most activities have an impact on the user's availability of time and on the requirements on his location; a very relevant set of activities involve monetary exchanges and social networking. Moreover, these resources are already associated with a lot of widely used applications and services, that are operated by the user during his everyday life. Therefore, considering these four resources, we can achieve a major impact on the way the user organizes his activities, providing strong support for the data integration and service composition, while focusing on an agile and manageable set of concepts which can be described and handled in practical ways.

We also present a platform that supports the user in his activities. Based on the notion of resources, the platform enables the information integration and service composition, coherently propagates the use of these resources by the applications, and continuously monitors their evolution in the user context. Finally, we present and discuss the preliminary implementation of our approach on a mobile platform, and show how it may be applied to a set of scenarios.

The paper is structured as follows. In Sect. 2 we present a set of scenarios describing the problems and requirements to the user-centric composition of mobile services. Section 3 presents our resource-driven composition approach, describe the conceptual architecture and prototype implementation of the underlying platform. Section 4 concludes with the discussion on the related works.

2 Motivating Scenarios

We illustrate the problems of providing the user-centric mobile service composition using three scenarios covering different aspects of typical user activities. In particular, the scenarios aim at demonstrating how the information is integrated and maintained across different services, how the user intentions are operationalized through service compositions, and how the dynamically changing context of the user affects and drives the user activities. These scenarios rely on a set of services and applications – agenda, communications, map and navigation, context tracking, payment – that refer to different domains (e.g., travelling, personal activity management, entertainment).

2.1 Data Integration: Booking a Movie Ticket

Passing an interactive poster of the cinema [1], the user decides to go to cinema with his friend. Using the corresponding interaction means (e.g., NFC), the user

downloads to his phone an information about the movie show. In order to organize the cinema visit, the user has to perform a set of activities, such as access the booking service (indicated in the above information) to book the tickets, activate the payment procedure, save the movie show in the agenda, share the event with a friend using communication application, add the cinema location to the navigator or map application.

Even if all the above activities are centered around the management of the same object (i.e., movie show), they are not directly related neither to each other nor to this particular object. First, this forces the user to repeatedly input the same data to different applications and services. Second, the actions are performed manually and independently; the associations between the services and objects are not persistent. As a result, if in the future the user decides to cancel the movie show, he has to perform the integration activities again, repeating the data input, and keeping in mind all the relations.

Therefore, the target platform should address the following essential requirements. First, it should provide a means for relating and integrating information entities and services from different domains. Here an important challenge is to achieve a balance between the spectrum of concepts to cover and the simplicity and flexibility of integration provision. Given the diversity and dynamics of the information operated through the mobile device, it is crucial to be able to deal with a compact but rather expressive and pragmatic set of concepts. Second, it should be able to maintain the integration over time so that the related services and objects are managed in a consistent way. Such relations, for instance, would allow one to identify and initiate a new service composition when the user decides to cancel or to change the date of the cinema visit.

2.2 Service Composition: Business Trip

The user receives an invitation to an international event and saves the event in his agenda. If the event takes place in a different city or country, the user has also to organize a trip, since otherwise his future plans become inconsistent with respect to the location of the user. This activity includes several actions (plan an itinerary, search for transportation means, book and pay the travel tickets and a hotel) and involves various services (Web, TelCo, local) that are integrated on purpose of resolving such inconsistency.

In order to support the user in such situations, it is crucial not only to associate and relate different services and information objects, but also to take into account personal information of the user, his requirements and constraints (such as the above inconsistency). That is, the system should be able to identify these requirements, to associate them to the relevant services and applications, and to provide the user with the composed procedures that aim at satisfying the constraints.

A similar situation takes place when the user has to manage conflicting (overlapping) personal activities. For instance, the trip may be in conflict with the movie show, and therefore the latter should be moved or cancelled. Again, the cancellation is driven by the necessity to resolve the inconsistency in personal

information of the user, and therefore the platform should be able to detect this and to provide an appropriate solution.

2.3 Monitoring: Trip Cancellation

In many cases the activities are initiated by external events and changes in the user context. For instance, Lufthansa provides a service that sends notification to the subscribed passengers when their flights are being delayed or cancelled. Given the above scenario, when the user is notified about the flight delay, he has to check whether the delay is still safe with respect to the meeting time, and if not, to reorganize or to cancel it. The latter may include removing the trip and the meeting from the agenda, cancelling flight booking, notifying involved people about the problem, etc. Again, the user should take care of the dependencies between the flight and meeting, time constraints, necessity to cancel the booking in order to get money back, making the involved people aware, and so on.

An important functionality that the target platform must provide is the ability to continuously observe the user operational environment and context and to react to the changes and relevant events proactively. In the above scenario, this includes the need to react to the notification about the flight modification; in other situations this might require to track the location of the user through the GPS navigator, etc.

3 Resource-Driven Composition

The presented scenarios give rise to a set of important problems and requirements to the integration and provision of mobile services to users. In order to address these requirements we propose a novel user-centric service composition approach, which is based on the notion of "resources". These resources correspond to the core user assets that he operates and changes during his everyday activities, namely user agenda, location, finances, social network. This compact set of concepts reflects very pragmatic and minimalistic approach to modelling the relevant information: on the one hand it allows for describing a wide range of services, events, and information entities, and on the other hand does not require an exhaustive and very sophisticated models of the real world.

Figure 1(a) represents our resource-driven information and application integration approach, where the resources play fundamental role and perform the following functions.

First, the resources are used to relate different independent application domains: information entities and actions from these domains are interpreted in terms of resources and resource-critical actions (e.g., ticket payment is related to finances, while the show is related to time and location). This provides the basis for the integration of mobile applications and services.

Second, the resources are used to consider and interpret the state of essential user assets in order to align the resource-critical actions with the actual user needs and constraints. In this way, for instance, the approach is able to relate

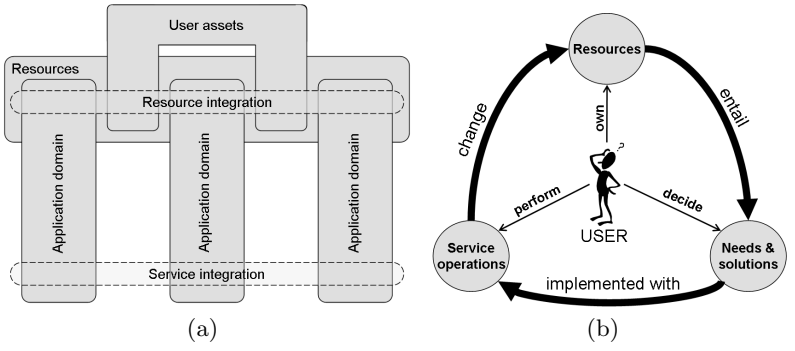


Fig. 1. Resource-driven approach and composition life-cycle

the location of a business meeting with the expected user location, to identify the location problem, and the necessity to integrate various applications in order to organize a corresponding trip.

Finally, the integration performed at the resource level entails the corresponding integration at the level of the actual applications and services. Here the abstract resource-related information and actions are automatically grounded and operationalized using available services, as well as the interactions with the user.

The life-cycle of our resource-driven approach is conceptually represented in Fig. 1(b). The service invocations and the context changes performed by the user are associated with the effects on the owned core resources. These effects are integrated with other resource-critical information, potentially leading to the inconsistencies in the state of resources, and therefore may require new resource-critical activities. The activities are operationalized with the set of available services and applications, and the user decides which of them are relevant and have to be performed.

In the following we present the conceptual architecture and its reference implementation, which supports the presented approach providing the following functionalities:

- represent and associate entities and services from different domains in terms of resources and resource-related actions;
- represent and manage the user resources and their evolution;
- propagate the effect of the actions and operations to the user resources;
- propagate the resource inconsistencies and problems to the user and applications in order to operationalize their resolution;
- continuously monitor the resource-related context of the user in order to synchronize the resource-critical information.

3.1 Platform Architecture

The architecture of the platform (Fig. 2) consists of three main components, namely the *Resource Layer*, the *Control Layer*, and the *Application Layer*. This

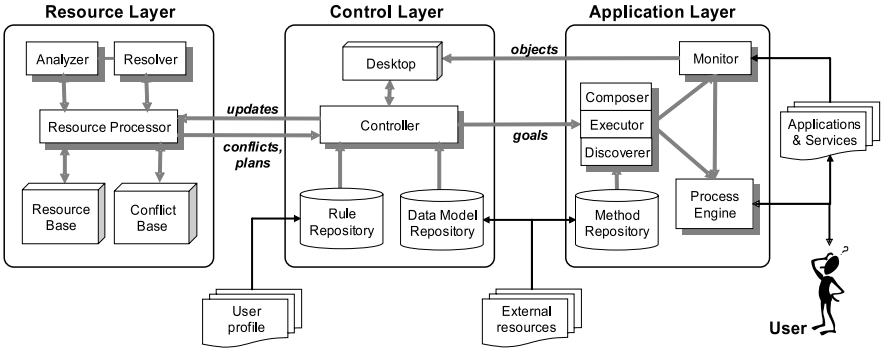


Fig. 2. Platform architecture

decomposition provides a clear separation of concerns in the user-centric operational environment.

Resource Layer. The *Resource Layer* (RL) component keeps track of the use of the core resources and the relations between different resource actions, identifies and analyzes potential inconsistencies, and finds ways of resolving these problems.

When the user resources are modified, the system updates the RL component with the information on the modification: the *Resource Processor* updates the current state of resources (*Resource Base*), analyzes its consistency with respect to a set of resource-specific constraints (*Analyzer*), identifies and stores the resource problems (*Conflict Base*), and looks for the resolution actions on resources (*Resolver*). The information about identified conflicts, as well as the actions needed to resolve them, are returned to the system. This includes the type of the problem (e.g., location inconsistency), its parameters, and a set of abstract, domain-independent resource-related operations needed to make the user assets consistent.

Control Layer. The *Control Layer* (CL) component is used to integrate and harmonize the information from different applications and services according to the model of user resources and their manipulation.

The component continuously processes the information events generated by the applications and by the user context; it integrates various information entities and relates them to the resources and to the resource updates. The information events deliver information about relevant objects as instances of various data structures (e.g., XML document describing the movie ticket) and about their evolution (e.g., change of entity state or modification of relevant information). For this purpose shared data model (*Data Model Repository*) is used. It describes the corresponding data structures from different application domains and their relation to resources.

Information events are published and stored in the *Desktop* and are processed by the *Controller*. Such processing consists of applying a set of associated ECA

(event-condition-action) rules stored in *Rule Repository*, which describe the operations and procedures to be executed in reaction to a specific event. This may include invocation of services, execution of service compositions, interaction with the user, etc. The rules may be configured according to the user preferences (*User Profile*); new rules may be installed during the platform evolution.

An important functionality of the CL component includes the interaction with the Resource Layer. It is used to update the RL component with new information regarding user resources and to handle the inconsistencies identified at the level of resources. The inconsistencies and proposed resource actions are related to the concrete objects, and then propagated to the Application Layer, where they are operationalized and executed using available services and applications.

Application Layer. The functionality of the *Application Layer* includes the identification, composition, and execution of the applications and services operated through the mobile phone; interactions with the user and managing his activities and decisions; monitoring and publishing relevant events from applications and user context.

The *Executor* module instantiates and controls the execution of actions specified by the CL component. It validates the availability of known services and find new ones (using *Discoverer* module), creates and adapts service compositions according to the new actions and context changes (*Composer*), and executes these actions and compositions (*Process Engine*). The set of services and applications known to the platform are defined in the *Method Repository*, which defines the service interfaces, protocols/parameters to use, data structure manipulated, etc. This repository, as well as the Data Model Repository, which defines the data structures and ontologies, operated by the platform, are shared among different layers. The repository can be extended with the new information from external sources.

In the AL component a set of *Monitor* plug-ins are installed, in order to track and signal external events, such as the SMS messages received, contextual changes, etc. The events are then forwarded to the CL component for processing.

3.2 System Execution

We illustrate the functionality of the platform using the scenarios presented above. We show how the events are processed, how the objects are related to the resources, and how the resource-driven composition is performed.

Realization of the Movie Ticket Processing. The platform is activated when the movie show information is downloaded and delivered to Desktop (Fig. 3). Using the data model, the Controller identifies the relation of the movie show to a set of resources, namely time (time of the show), location (address of the cinema), money (price of the ticket), and social (public event). These relations trigger the necessity to "allocate" the corresponding resources using available applications and services. The goal is operationalized in AL component by identifying and composing a set of relevant services. This involves booking

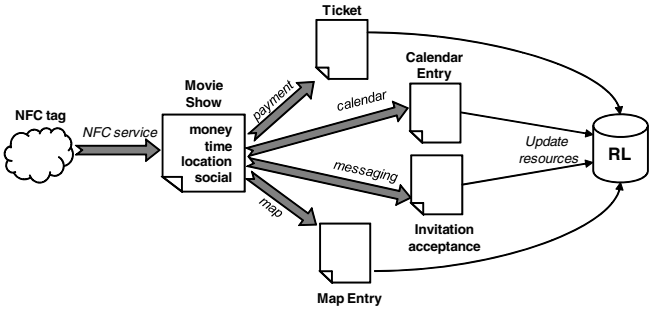


Fig. 3. Movie ticket processing

a ticket, paying it with the appropriate payment service, sending notification to a friend, adding the cinema address to the map, and storing event in the agenda. We remark that these operations are not just atomic service invocations, but complex procedures that include data transformations, interactions with the user, etc. In particular, payment procedure requires (*a*) asking the user to accept the payment request, (*b*) creating the payment request object, (*c*) invoking associated payment service, (*d*) informing the RL component about the action on the user finances. In a similar way, the update of the calendar and the event invitation/acceptance are followed by the updates of the corresponding resources.

When the procedure is activated (upon the decision of the user or automatically), the composition is executed, and the results of the executions are also propagated to Desktop. In this way, the platform observes the information integration, and relates the different objects and events. These relations are made persistent in order to continuously guarantee the consistency of the managed information entities. For instance, the payment confirmation, the agenda and map entries are associated to the movie ticket: if later on the user decide to cancel the visit to the cinema, the platform will also be able to cancel the payment, notify the friend, and even remove an entry from the map.

Realization of the Business Trip Management. The scenario flow is represented in Fig. 4. When the business meeting is detected, a procedure similar to the one described above is engaged. Besides this, the RL component performs the analysis of the user resources with respect to the meeting information (time and location), and detects the corresponding location inconsistency. The problem is reported to the Control Layer that uses the AL component to operationalize the resolution. Depending on the parameters of the problem, different resolution variants may be defined. In our scenario the resolution includes finding and selecting a flight trip, paying the chosen option, and updating the agenda. As before, the procedure is constructed by the Composer module, and is executed by the Process Engine. In case of positive outcome, the results of the execution will be published in the Desktop, propagated to the Resource Layer, and the user resources will be updated to a consistent state.

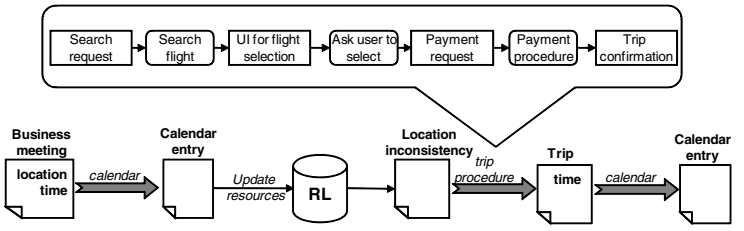


Fig. 4. Managing business trip

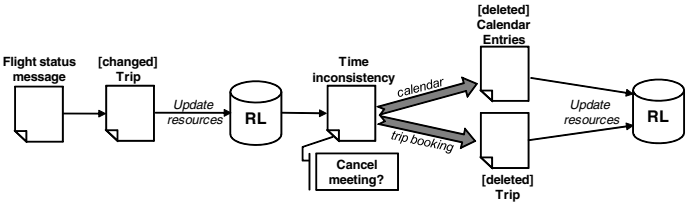


Fig. 5. Flight cancellation

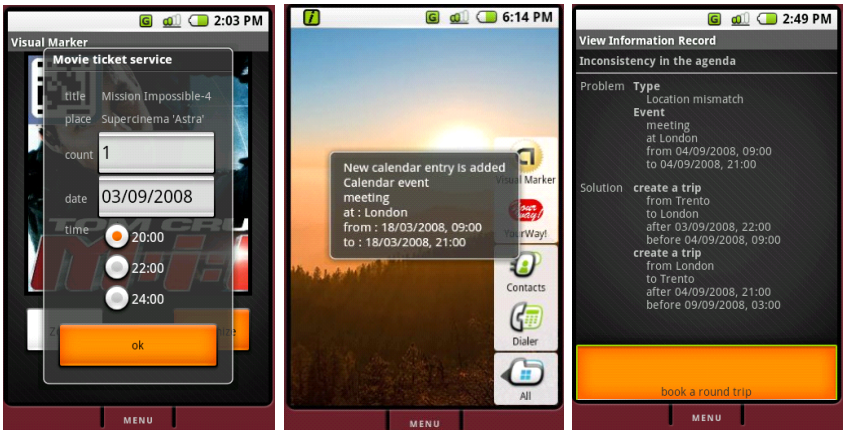
Realization of the Flight Cancellation. The scenario flow is represented in Fig. 5. When the message with the delay information is detected by the monitor, it is published as the flight trip modification event, in particular, its time modification. This modification is propagated to the Resource Layer, and the inconsistency is detected. This inconsistency requires that either the meeting (time) is cancelled or another trip is organized. Both options are operationalized and offered to the user.

We remark that both the proposed solutions include management of other related objects. In case of meeting cancellation, this includes cancelling the trip using the trip booking service, removing the entries corresponding to the meeting and the trip from the user agenda (calendar), etc.

3.3 Platform Implementation

We have implemented a prototype of the conceptual platform for the resources-driven user-centric composition of mobile services. While the prototype is still preliminary, it has already made possible the definition, integration, and delivery of the discussed functionalities on a mobile platform. In the future we plan to extend and enrich the proposed platform with new services and functionalities.

The prototype is designed as a distributed software platform, in which some components are located on a mobile phone, and the others (that require expensive computations and persistence) are accessed remotely. In particular, the RL, CL, and in part AL components are implemented as a Web application, that interacts with a mobile part via a specific protocol. The mobile part contains the platform front-end and the corresponding UI components, which inform the user



(a) (b) (c)

Fig. 6. Platform UI

about various conflicts and important objects, offer different actions to execute, and query additional information required for invocation of services and applications. The mobile part of the platform is implemented on top of the Android platform [2], an open java-based operating environment for mobile phones.

Apart from the applications and facilities provided by the platform (e.g., UI components), the implementation makes an extensive use of remote services and applications. In particular, the Google Calendar application is used in order to manage the user agenda, and the Google Mail application is used to monitor the e-mail messages and notifications from other users. Also, a set of mock-up Web services were implemented for testing/demonstration purposes, such as payment service, movie ticket service, flight search, etc.

Figure 6 shows some screenshots of the platform interface. In particular, Fig. 6(a) represents a dialog window of the movie ticket service. Figure 6(b) shows the notification about the result of the calendar update with the information about the business meeting. Finally, Fig. 6(c) demonstrates the location inconsistency problem identified by the platform, its parameters, necessary actions, and the reference to the trip booking procedure corresponding to these actions.

4 Related Works and Conclusions

We have presented an approach to the problem of user-centric data integration and composition of mobile services. The approach relies on the notion of resources, i.e., a compact set of concepts used to model and represent the important aspects of the everyday activities of the user. This allows for expressive and pragmatic integration of independent application domains, and enables the automated composition of mobile services. The approach also incorporates the management and evolution

of the associated user assets, thus providing a way to continuously adapt the service provision and execution to the user activities and needs. We also presented a conceptual architecture for the supporting platform, and discussed its prototype implementation.

So far, research in service composition has focused on the problem of task-centric composition, i.e., on creating a composed process that performs a desired task defined by the composition requirements and interacts with a set of services. The existing approaches target the problem both at the functional level (composition of atomic services, [3,4,5]), and at the process level (take into account stateful behavior of components, [6,7,8]). Our approach is radically different: it starts from the simple requirements, where resources play key role, and from the service descriptions, whose semantics takes resources into account; it considers and integrates all the services related from the resource point of view, even if they perform completely unrelated tasks; it is continuous and is dynamically performed at run-time, when the context and conditions are changing. Our approach also differs from the approaches to the composition of Semantic Web services that take into account user preferences and constraints [9,10]. Our user-centric approach composes services through a set of core resources, which play the crucial role for all the users in general. User preferences may be specified and integrated on top of it.

The way the data is represented and integrated is also rather different from the approaches adopted in Semantic Web services, where the description of services and service data relies on expressive and complex languages and models in order to target automated inference, discovery, and integration of generic services and domains [11,12,13]. In our approach we focus on a compact set of resource-related concepts, targeting a pragmatic and minimalistic description and reasoning on the service and data models. The same holds for the context-related information, where we focus only on the resource-related aspects. This is different from the traditional approaches to the context awareness that define the context in a very unbounded way [14].

References

1. Paolucci, M., Broll, G., Hamard, J., Rukzio, E., Wagner, M., Schmidt, A.: Bringing Semantic Services to Real-World Objects. IJSWIS 4 (2008)
2. Open Handset Alliance: Android mobile phone platform
3. Sheshagiri, M., des Jardins, M., Finin, T.: A Planner for Composing Services Described in DAML-S. In: Proceedings of the Second International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS (2003)
4. Narayanan, S., McIlraith, S.: Simulation, Verification and Automated Composition of Web Services. In: Proceedings of the 11th international conference on World Wide Web, WWW (2002)
5. Wu, D., Parsia, B., Sirin, E., Hendler, J., Nau, D.: Automating DAML-S Web Services Composition using SHOP2. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 195–210. Springer, Heidelberg (2003)

6. Pistore, M., Traverso, P., Bertoli, P., Marconi, A.: Automated Synthesis of Composite BPEL4WS Web Services. In: Proceedings of the IEEE International Conference on Web Services, ICWS (2005)
7. Khalaf, R., Mukhi, N., Weerawarana, S.: Service Oriented Composition in BPEL4WS. In: Proceedings of the 12th international conference on World Wide Web, WWW (2003)
8. Berardi, D., Calvanese, D., Giacomo, G.D., Mecella, M.: Composition of Services with Nondeterministic Observable Behaviour. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 520–526. Springer, Heidelberg (2005)
9. Baier, J., Bacchus, F., McIlraith, S.: A Heuristic Search Approach to Planning with Temporally Extended Preferences. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI (2007)
10. McIlraith, S., Son, S.: Adapting Golog for Composition of Semantic Web Services. In: Proceedings of the Eighth International Conference on Principles and Knowledge Representation and Reasoning, KR (2002)
11. OWL Service Coalition. OWL-S: Semantic Markup for Web Services (2003)
12. Roman, D., Lausen, H., Keller, U., Oren, E., Bussler, C., Kifer, M., Fensel, D.: Web Service Modeling Ontology (WSMO): WSMO Working Draft (2004), `http://www.wsmo.org/2004/d2/v1.0/`
13. Noll, J., Kileng, F., Hinz, R., Roman, D., Pilarski, M., Lillevold, E.: Semantic Service Delivery for Mobile Users. In: Proceedings of the WWRF N17 meeting, WG2 (2006)
14. Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. In: Proceedings of the Internatonal Symposium on Handheld and Ubiquitous Computing (2006)