User-Driven Service Lifecycle Management – Adopting Internet Paradigms in Telecom Services

Juan C. Yelmo[1], Rubén Trapero[1], José M. del Álamo[1], Juergen Sienel[2], Marc Drewniok[2], Isabel Ordás[3], and Kathleen McCallum[3]

[1] DIT, Universidad Politécnica de Madrid, Ciudad Universitaria s/n, 28040 Madrid, Spain
{jcyelmo, rubentb, jmdela}@dit.upm.es
http://www.dit.upm.es

[2] Alcatel-Lucent Deutschland AG, 70435 Suttgart, Germany
{Juergen.Sienel, Marc.Drewniok}@alcatel-lucent.de
www.alcatel-lucent.com

[3] Telefónica I+D, Emilio Vargas 6, 28043 Madrid, Spain
{ioa, kmc352}@tid.es
http://www.tid.es

Abstract. The user-centric service creation paradigm set out in Web 2.0 technologies on the Internet allows users to define and share their new content and applications. Open services and interfaces provided by Google et al can be used to build easily, and quickly deploy exciting applications. User-centric service creation provides a cheap solution in contrast with the huge engineering effort that has to be spent both on development and marketing in order to get a new telecom service running and deployed in the market. Adopting Internet paradigms in telecom services requires major flexibility and dynamicity in managing service lifecycle compared to current service management systems.

This paper proposes an approach to user-centric service lifecycle management in telecom-oriented platforms. It allows users to drive their services' lifecycle, e.g. when and for how long they must be available, as well as automating the process between the creation and the execution of the services.

1 Introduction

In an IP world, new competitors such as mobile virtual network operators (MVNO) or Internet companies threaten the traditional business models of telecom operators by providing their services directly to the operators' customers. They use the operator's network as a kind of bit pipe [1], without returning any benefit to it in exchange for the use of those services.

Moreover, end users are also putting on pressure by increasingly requiring innovative and attractive new services: They would like to have the advanced model they use on the Internet which allows users to define new contents and applications (mashups) using open services and interfaces that could be quickly and easily built and deployed e.g. Yahoo Pipes [2].

User-centric service creation refers to this process. It enables end-users (not necessarily the very technically skilled) to create their own services and manage the lifecycle of those services autonomously. It also allows users to share these services within a community which will promote the most interesting ones at a minimum cost (viral marketing[1]).

In order to support the aforementioned approach, it has become imperative for telcos to change their rigid business and provisioning models, replacing them with much more agile processes. This could be accomplished by identifying the operators' assets that can be provided only in the core network such as end-user location and presence information, and then abstract and offer them through well defined interfaces. Users may use these resources to create new or personalized services, thus generating a powerful and self-increasing ecosystem around the telecom operators' core business - their networks.

Service Oriented Architecture (SOA) is the main approach to opening up these network resources with initiatives such as ParlayX [3] or Open Mobile Alliance (OMA) enablers [4], which converge in the use of Web Services as the middleware allowing third parties to access and control network resources.

Nonetheless, operators still have to cope with some difficulties that arise from the openness of their networks and the reduction of the time to market in the lifecycle of new services. Furthermore, in order to apply the user-centric model telcos must also deal with a huge set of short-lived user-generated services each one having its own user-driven lifecycle and orchestrating a subset of telecom based services.

The Open Platform for User-centric service Creation and Execution (OPUCE) [5] allows users to create, manage and use their own telecom-based services in an Internet style. In this article we introduce some of the early results within the OPUCE project, focusing on the lifecycle of these services. We pay special attention to user-driven lifecycle management of user-centric services as well as the automation of the related processes.

The following sections introduce the fundamentals of the OPUCE project stressing the importance of having a structured way of describing user-generated services and their lifecycles in such an open service ecosystem. The paper continues by setting out the model that supports user-driven service lifecycles in a user-centric telecom environment. Section 5 concludes the paper.

2 The OPUCE Project

The OPUCE project is a research project within the European Union Sixth Framework Programme for Research and Technological Development. OPUCE aims to bridge advances in networking, communication and information technology

[1] Viral marketing refers to marketing techniques that use pre-existing social networks to produce increases in brand awareness, through self-replicating viral processes, analogous to the spread of pathological and computer viruses. It can be word-of-mouth delivered or enhanced by the network effects of the Internet. [Source: Wikipedia].

services towards a unique service environment where personalized services are dynamically created and provisioned by the end-users themselves.

The general objective of OPUCE is to leverage the creation of a user-centric service ecosystem giving users the chance to create their own personalized services as is currently done on the Internet.

Within this approach, service concepts are redefined. In OPUCE, services are envisioned as short-lived telecom services that end-users will create by orchestrating simpler services called base services. Base services are considered as functional units deployed by the operator or authorized third parties, available at the OPUCE platform and offered to end users through Web Services interfaces.[2]

Figure 1 introduces a detailed diagram of the OPUCE architecture. Its main elements are:

- A *Service Creation Environment* with a set of tools to be used by people or third parties to dynamically create services [6]. It can be seen as a portal through which users can create, deploy and share services. It consists on two portals: a user portal, to manage social networks, service subscriptions and configurations, etc; and a service portal, to manage the service edition, test, simulation, monitoring, etc. The Service Creation Environment also includes other general functions (access control, registration) and administration tools.
- A *Context Awareness* module to manage the adaptation and customization of services to the users' ambience conditions. In OPUCE two types of context aware adaptations are supported: explicit, when it is the service creator who specifies the service behavior taking into account user context information; and implicit, when the platform itself analyzes the service and adapts dynamically the execution.
- A *User Information Management* module to control the user's personal information (agenda, buddy list, presence information, device capabilities, potential use of certain services, etc.) identity management and AAA.
- A *Subscription Management* module to keep control of the user subscriptions to services. The information that this module stores is mainly consumed by other OPUCE modules (such as the context awareness or the user information modules).
- A *Service Lifecycle Manager* module which manages the lifecycle of all services created within the OPUCE platform. Section 4 describes in depth this module.
- A *Service Execution Environment* which manages the execution of services and orchestrates base services following the logic created by the users when composing them. A BPEL (Business Process Execution Language) [7] engine is used to orchestrate them, thus it is necessary to wrap those base services with Web Services interfaces. This module also supports events, managed by an event handler. Those events are generated by other OPUCE modules, such as the context awareness module.

[2] This definition is still coherent with the extended idea of service: Services are autonomous, platform-independent computational elements that can be described, published, discovered, orchestrated and programmed using standard protocols for the purpose of building networks of collaborating applications distributed within and across organizational boundaries. [Source: ICSOC 2005].

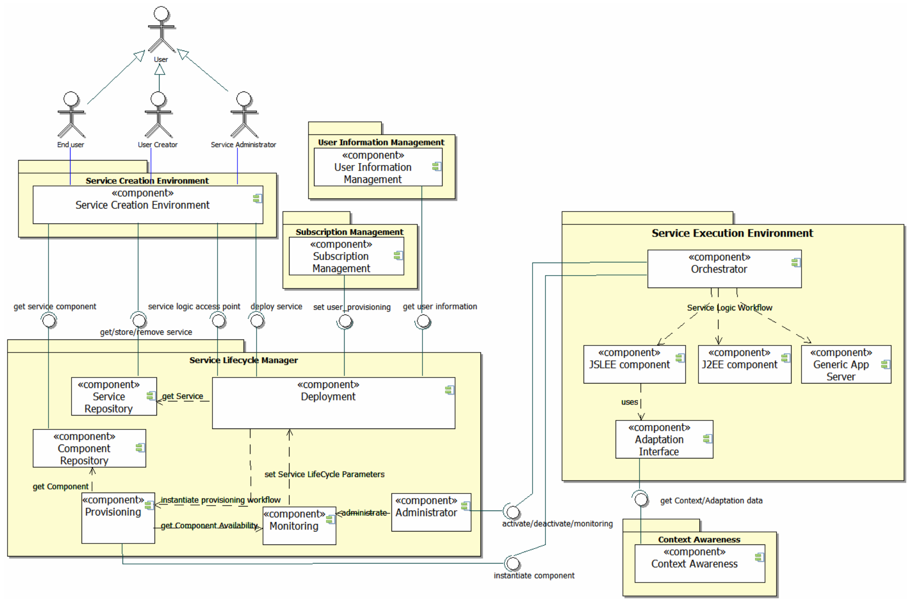
User
End user
User Creator
Service Administrator
Service Creation Environment
«component» Service Creation Environment
User Information Management
«component» User Information Management
Subscription Management
«component» Subscription Management
Service Execution Environment
«component» Orchestrator
Service Logic Workflow
«component» JSLEE component
«component» J2EE component
«component» Generic App Server
uses
«component» Adaptation Interface
get service component
service logic access point
deploy service
set user provisioning
get user information
get/store/remove service
Service Lifecycle Manager
«component» Service Repository
get Service
«component» Deployment
«component» Component Repository
set Service LifeCycle Parameters
get Component
«component» Provisioning
instantiate provisioning workflow
«component» Monitoring
get Component Availability
administrate
«component» Administrator
activate/deactivate/monitoring
instantiate component
get Context/Adaptation data
Context Awareness
«component» Context Awareness

Fig. 1. The OPUCE architecture

This architecture has been validated with the creation of an intelligent email service. It forwards an incoming email according to the situation of the user and the device he has connected to the platform. For example, if the user is driving the service will set-up a call to read the email. If the user is not logged in his office PC, the service may also send an SMS containing the email subject and sender. The base services combined by the user creator are an email service, an SMS service, a text-to-speech call service and a user-context service.

Since this paper is focused on the service lifecycle of these user-centric services, we concentrate on both the Service Lifecycle Manager and the Service Execution Environment. The following sections will detail how user's can indirectly interact with these platform modules when creating services.

3 Supporting User-Centric Services: Service Description

A user-centric based service ecosystem requires major flexibility and dynamicity in managing service lifecycle compared to current service management systems. In order to automate the process between the creation and the execution of the services, the OPUCE platform needs a common way to completely describe the services.

The concept of service description has already been addressed in projects such as SPICE [8] and SeCSE [9]. Some of the results of these projects have been considered in OPUCE to create a service specification model completely. More precisely, the faceted approach [10] of the service description defined by the SeCSE project has been extended to support the user-centric features of OPUCE services, including the user-driven lifecycle management of user-centric services. Figure 2 depicts the faceted specification followed in OPUCE.

Therefore, in OPUCE, services are described using a service specification which contains all aspects of a service. Each aspect, called a facet, is further described in a separate XML-based specification. With a view to all the functionalities available in the OPUCE platform we have considered three sets of facets:

- *Functional facets*, which include service logic facets, service interface facets, service semantic information facet, etc.
- *Non-functional facets*, which include service level agreement (SLA) facets, quality of service facets, etc.
- *Management facets*, which include service lifecycle schedule and deployment facets.

In this article we focus on the management facets and describe how their contents allow users to interact with various aspects of the lifecycle of services, such as the service scheduling, and how to automate the deployment of services.

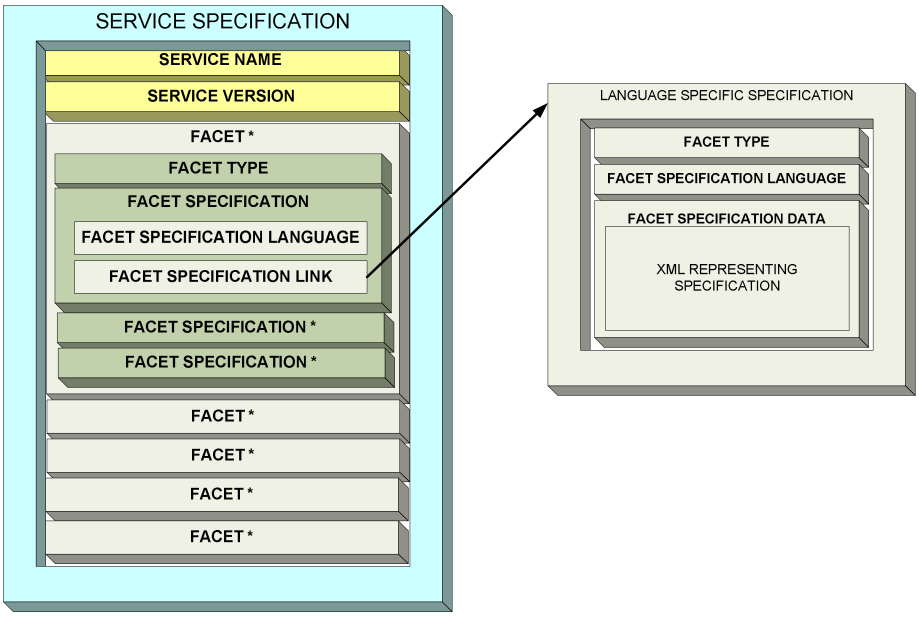

Fig. 2. Faceted approach used to describe services

4 Service Lifecycle Management in User-Centric Telecom Platforms

One of the major constraints in today's telecommunication services is the huge engineering effort that has to be spent both on development and marketing, in order to get a service running and deployed in the market. On the other hand, the Internet model provides a quite different and cheaper approach through user-centric service creation, e.g. mashups and viral marketing.

In order to transfer the advantages of the Internet model to telecom environments the platform must be able to cope with a huge set of user generated services each one orchestrating a subset of telecom-oriented base services. The requirements for the lifecycle management of these new services are:

1. to allow users to decide when and for how long their services must be available i.e. the lifecycle schedule. Since we encourage end-users to create services, the lifetime of services may be very short including even a one-time usage of a service. This means the overheads for users and platform administrators to deploy the services must be limited to an absolute minimum, which brings us to the second requirement;
2. to be able to provision base services and platform elements automatically, and to register and publish new services in order to be available to end-users.

As each service is compound of a different subset of base services and has its own dynamicity, lifecycle management processes in user-centric telecom platforms cannot be completely set beforehand and will depend both on the base services used and the user needs. However, there is at least one set of common activities a service always goes through in a telecom platform: creation, deployment, maintenance, and withdrawal.

- *Service Creation.* End users compose their own services by orchestrating a set of telecom base services. At the end of this process a service description is generated and stored in a service repository.
- *Service Deployment.* This is a platform supported process to make a service available in the communications environment. It usually includes physical installation, provisioning, registration, publication and activation. These tasks must be carried out in a given order. Whenever something fails, the steps already carried out must be undone. Deployment finishes when the service is up and running and ready to be subscribed to by end users.
- *Service Maintenance.* After the service has been properly deployed, this step will help the service provider to analyze the service execution status. This information will help to improve the service based on monitoring and usage statistics, to optimise the resource usage spent on that service and to identify errors and other runtime problems.
- *Service Withdrawal.* If the service is going to be substituted or evolved, or will not be used for a while, it must be stopped. Then, when it is no longer needed it must be cleared up from the platform. Service withdrawal consists of undoing the steps carried out during deployment phase; i.e. deactivation, withdraw the publication (unpublishing), deregistration, unprovisioning and physical uninstallation. Again, these tasks must be carried out in a given order.

The whole process a service goes through during its lifecycle can be formalized in a UML state diagram (Figure 3). Within OPUCE we consider that the execution platform exists and provides enough resources to provision all components and deploy the service. For this reason the physical installation and uninstallation will not be taken into account. For later stages of the project we will provide virtualized resources which can be set-up on demand with appropriate functionality. This virtualisation will provide not only the means to adapt the platform dynamically to the specific needs of the services running on top, but also provide a secured sand-box for testing and running services in a separate environment.

The activities associated to the other sub-states within the service deployment and service withdrawal depend on the base services used and the one that is composed: reservation of network resources, provision of base services, deployment of the new service, and so on. This information is implicitly obtained during the creation process. Therefore once the platform knows the set of base services that are used, and how the creator orchestrates them, it automatically generates the description of the activities associated to the management of the service lifecycle. Finally this information is stored as a set of facets within the service description: deployment and provisioning facets.

On the other hand, in order to allow users to drive the lifecycle of their new services, the Service Creation Environment explicitly collects information about the desired activation and deactivation events. This information, which allows the schedule of the service to be known, is also stored as a facet within the service description: lifecycle schedule facet.

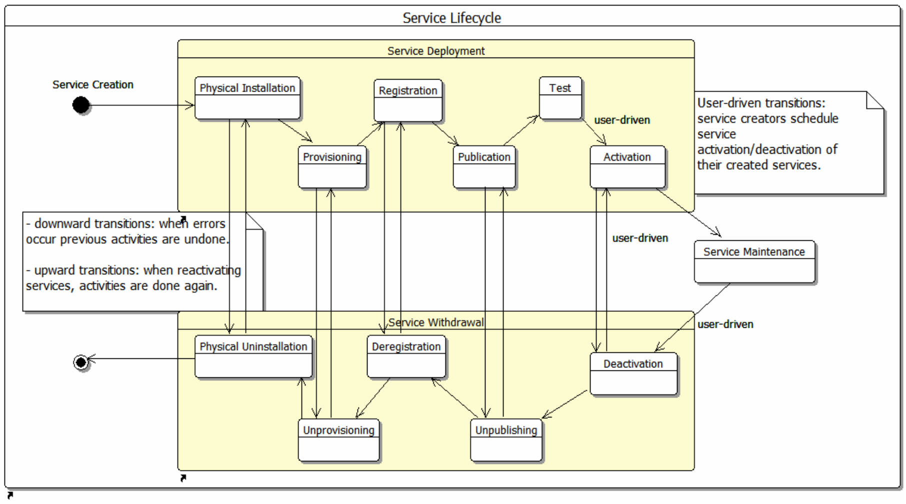

Fig. 3. The Service Lifecycle in OPUCE

Therefore, the service description includes all the information needed to carry out automated user-driven service lifecycle in its facets.

The following subsections detail this service lifecycle. First we explain how service creators can modify some aspects of the service lifecycle, scheduling the activation and deactivation of services. Then, the deployment process is described and how the service description contributes to making this process automatic. Finally, we close with the description of the lifecycle detailing the service withdrawal.

4.1 User-Driven Service Lifecycle

Within a user-centric service lifecycle model, users would like to decide, depending on their own preferences, when, where and how their services must be active. For example, an SMS service sending messages of the results of the local football team should only be available every weekend from Friday afternoon to Sunday evening, or a personal friend finder will be activated based on the location context.

Since activation in telecom platforms means the real allocation of resources, and therefore expenses to the platform providers, the deployment activities must be planned based on the initial schedule that the creator has decided for the service. On the other hand, at the end of the lifecycle the operator needs to be sure that all allocated resources are released and can be re-used for other services. Therefore withdrawal activities must be carried out after the service has been deactivated, whether the service is going to be activated again (following the user schedule) or if it is going to be definitely removed.

Within the OPUCE architecture the entity that controls and monitors the lifecycle activities is the Service Lifecycle Manager. This subsystem knows the current state of a service lifecycle, how to make a transition, and how to carry out the activities inside each transition. This entity also knows the user-driven transitions, monitors their occurrence, and triggers the related events. This entity must also set the policy for the other events within the lifecycle.

4.2 Service Deployment

Automatic deployment is nothing new in the Telco world, however the traditional deployment, automatic or not, has always been instigated and determined by operators. In OPUCE we go a step further; letting the service creators (the users) take control of some aspects of the deployment, such as the service scheduling.

There are more aspects that make the difference between OPUCE deployment concepts and traditional ones: the nature of the services. It is the user himself who triggers the deployment of the services until it is activated and finally available to users. In OPUCE, services are also created from outside the operator's administration domain, either from a computer, PDA or cellular phone and from different places such as the users' living room, a taxi or from their workplace.

Therefore, in OPUCE there is no human intervention from the operator either when triggering the service deployment process or during the deployment itself. Thus the service description must contain all the data that is necessary to deploy the service in each of the activities that make up the service deployment. As we will see, with the faceted approach the information needed to be handled by each activity can be clearly separated.

In the intelligent email service, once the creator has finished the composition the service must be deployed and provisioned to the platform. The Service Creation Environment will generate the service description including the facets needed for the deployment, i.e. service logic description, service deployment facet, service provisioning facet and the service lifecycle schedule. The deployment process is initialized and the following steps are executed in the OPUCE Platform:

- *Service Provisioning*. In OPUCE service provisioning is carried out automatically contrary to traditional provisioning which often requires a manual participation to create accounts for users, reserve resources, etc. We have identified three provisioning tasks, each one affecting different elements of the OPUCE architecture.
  - *Component Provisioning*, which considers the provisioning tasks to be done in those base services that the user creator has combined. These activities include the reservation of resources or configuration of permissions to use the base services. In the intelligent mail service, one of the component provisioning tasks is the creation of an email account for the user in the email base service.
  - *Platform Provisioning*, which considers the provisioning tasks to be carried out in the OPUCE platform components such as updating billing systems, updating service repository. These tasks are common for all the services created.
  - *User Provisioning*, which considers the provisioning tasks to be carried out on the user side at subscription time, such as installing a dedicated application in the user's mobile device. In our example, the user context base service requires that a module for generating the location information is provisioned on the client device. This provisioning task also involves the Subscription Management module to control subscriptions of end-users to services.

Each type of provisioning is a compound of different tasks. Thus separate facets for each type of provisioning are nested within the provisioning facet so as to be easily separated and distributed to the provisioning module within the Service Lifecycle Manager. Figure 4 depicts the service description structure for this facet and the relationship of each type of provisioning with the corresponding OPUCE architecture module.

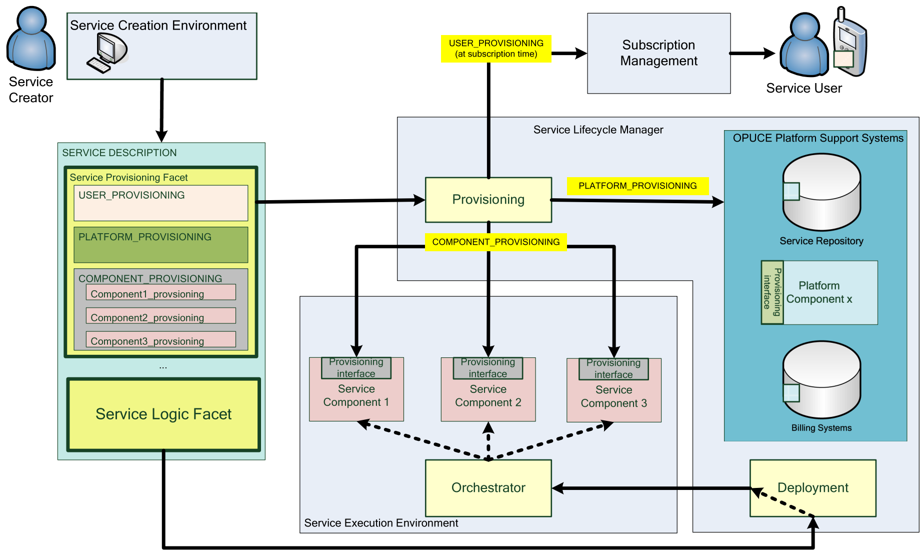

Fig. 4. Service provisioning and deployment in OPUCE

- *Service Registration.* In this activity the platform registers all the information needed to access the service once activated, such as endpoints if it is a Web Services-based access. In our service prototype, it consists of registering the email address as the way to access to the service.
- *Service Publication.* In this activity the service is officially published including all associated attributes, e.g. service type, descriptions, terms and conditions of use, etc, making service discovery easier. Some of these data are taken from the service description, such as the service name, the semantic facet, etc. Publication is done via a service advertiser to which other users can subscribe specific keywords describing the service instance like "intelligent email", "email forward" or "email reading".
- *Service Test.* The aim of this activity is to ensure that a new service is ready to be subscribed to and consumed by end-users. Nonetheless, this activity has not been considered in the first stages of the OPUCE project. It might be included in the second iteration of the project.
- *Service Activation.* This is the last step of the service deployment process. Once the service has been activated it becomes publicly available and ready for subscription. This activity is triggered by using the information included within the service lifecycle schedule facet of the service description. This is done by connecting the email system to forward the emails to our receiving component and thus initializing the service logic flow. So it can be configured that the service is only active Monday to Friday between 8 am and 8 pm.

4.3 Service Withdrawal

Basically, the service removal consists of undoing the steps carried out while deploying the service (except for testing the service). Most of them are automatic, and there is no intervention from the user. As well as the service description containing information on the correct deployment and provisioning of a service, it also contains information about the undeployment and unprovisioning activities. This information is automatically included in the service description by the service creation environment.

On the other hand, the user-centricity capabilities of OPUCE services imply that the service description must also include some information about when the users would like the services to be deactivated. Just as with the activation activity, this information is explicitly introduced by the service creator when scheduling the service from the service creation environment and is stored in the lifecycle schedule facet.

5 Conclusions

The user-centricity of services is gaining momentum in the current opening up of telecom service ecosystems. Within this approach, services are created and managed by the end-users themselves, even if they are not technically skilled.

The OPUCE project aims to create a complete platform to support the creation, management and execution of user-centric services. Its initial results have been summarized in this paper, especially those focused on the service lifecycle management of user-centric services and the architecture that supports it. This service lifecycle is also driven by users since service creators can interact with some of its aspects: they trigger the deployment process and can schedule the activation and deactivation of the services they have created.

The role of the service description in such an automatic and customizable lifecycle has also been described. We have used a faceted service specification to describe different aspects of the service: the deployment tasks, the scheduling of services, etc. Each one is used at a certain stage of the lifecycle and by a specific architecture module of the platform.

In order to validate the suitability of the service description in conjunction with the user-driven service lifecycle considered in OPUCE, we have developed a simple, but realistic and relevant, prototype. It consists of a simple service which orchestrates several base services (an email service, messaging service, a text to speech service and a user context service). It allows users to receive the content of an incoming email either as an SMS or as a voice message, depending on user's situation. This prototype validates the service description and some stages of the service lifecycle, such as the user-driven automatic deployment.

Further work considers a potential contribution to the OMA Service Provider Environment specification [11], which is currently under development, with some aspects of this user-driven lifecycle of user-centric services.

Acknowledgments

This work is framed within the IST European Integrated Project OPUCE (*Open Platform for User-centric service Creation and Execution*), 6th Framework Programme, Contract No. 34101. We thank all our partners in the project for their valuable comments and proposals aiming at improving the conceptual model.

References

1. Cuevas, A., Einsiedler, H., Moreno, J.I., Vidales, P.: The IMS Service Platform: A Solution for Next-Generation Networks Operators to Be More than Bit Pipes. IEEE Commun. Mag. 44(8), 75–81 (2006)
2. Yahoo Pipes Website (2007), http://pipes.yahoo.com/pipes
3. ETSI Standard: Open Service Access (OSA); Parlay X Web Services; Part 1: Common (Parlay X 3). ES 202 504-01 (2007)
4. OMA Specification: OMA Web Services Enabler (OWSER): Core Specification. Version 1.1 (2006)
5. OPUCE Website (2007), http://www.opuce.eu
6. Caetano, J., et al.: Introducing the user to the service creation world: concepts for user centric creation, personalization and notification. International Workshop on User centricity – state of the art. Budapest, Hungary (2007)
7. Andrews, T., et al.: Business Process Execution Language for Web Services. Version 1.1 (2003)
8. SPICE Website (2007), http://www.ist-spice.org
9. SeCSE Website (2007), http://secse.eng.it
10. Sawyer, P., Hutchison, J., Walkerdine, J., Sommerville, I.: Faceted Service Specification. In: Proceedings of Workshop on Service-Oriented Computing Requirements, Paris, France (2005)
11. OMA Specification: OMA Service Provider Environment Requirements. Candidate Version 1.0 (2005)