2009 Eighth International Conference on Mobile Business

A User-centric Mobile Service Creation Approach Converging Telco and IT Services

Jian Yu[1], Paolo Falcarin[2], José M. del Álamo[3], Juergen Sienel[4], Quan Z. Sheng[1], and José F. Mejia[2]

[1]School of Computer Science
The University of Adelaide
Adelaide, Australia
jian.yu01@adelaide.edu.au
quanzheng.sheng@adelaide.edu.au

[2]Dep. Automation & Information
Politecnico di Torino
Torino, Italy
paolo.falcarin@polito.it
jose.mejiabernal@polito.it

[3]Universidad Politécnica
de Madrid
Madrid, Spain
jmdela@dit.upm.es

[4]Alcatel-Lucent
Deutschland AG
Stuttgart, Germany
juergen.sienel@
alcatel-lucent.de

Abstract—**While new competitors are threatening the traditional business models of Telecommunications operators by providing their services directly to the customer, user-centric service creation paradigm brings new opportunities for operators to deliver diverse, attractive, and profitable services directly to the end-user. This paper discusses the service creation model, architecture and implementation in the European Union sponsored research project OPUCE (Open Platform for User-Centric Service Creation and Execution), which aims at enabling end-users to use their smart mobile devices for both creating and consuming personalized services.**

Keywords-User-Generated Services, Service Creation, Service Delivery Platforms.

I. INTRODUCTION

The convergence of telecommunications networks and information technology has led to a dramatic impact on the business models of traditional operators. Especially the broad adoption of Internet technologies like the Session Initiation Protocol in the telco world changes the behavior of customers and the provisioning of telecommunications services to end-users. Companies like Skype (http://www.skype.com) have proven that they can cheaply provide telecommunications services directly to their customers, using the operators' network only as a kind of bit pipe [1]. To maintain their position as service provider, telecommunications operators need to climb up the value chain and take a more active part in service delivery [2].

Next generation network architectures like the IP Multimedia Subsystem [3] are maybe only an inadequate answer to the challenges that need to be addressed in an all-IP world, because it basically does not change the typical restrictive way that handles the usage of services and networking resources.

Looking at the Internet, technologies enabling the Web 2.0 [4] have created a tremendous boost of innovation, because end-users can now not only passively invoke available services but creatively combine and personalize them in a new fashion, offering and sharing their own services back to communities. Such a user-centric service creation paradigm enables even non-technically skilled people to create, manage, and share their own personalized services fitting their needs. It has gained a momentum on the Internet, with the release of several Web content and application mash-up tools such as Yahoo Pipes (http://pipes.yahoo.com) and Microsoft Popfly (http://www.popfly.com). But up to now a unique integration between the telecommunications and the Internet services, which fosters the creation of a personal communication space across terminals, networks and domain boundaries, is still missing. User-centric service creation as supported by the platform being developed by the OPUCE project (http://www.opuce.eu) does not only mean a faster and cheaper way of service delivery, but also offers new business models to promote the use of services such as messaging, location, and presence residing in the telco core networks.

In this paper, we report our user-centric service creation approach of the OPUCE project. Elementary communications services such as telephony, SMS, presence, and location and common IT services, either Internet-based or stand alone, are unified under the OPUCE Base-Service model, which form the building blocks for end-users to create their personalized and ubiquitous services that are directly delivered to their mobile devices. It is worth noting that end-users can also use smart handheld devices to create, deploy and run services on the move in a similar way.

The rest of the paper is organized as follows: Section 2 briefly introduces the OPUCE platform, including its user-centricity features and the architecture. Section 3 presents the details of OPUCE service model, including the Base-Service model and the Composite-Service model. Section 4 introduces the validation initiatives. Finally Section 5 discusses and concludes the paper.

II. THE OPUCE PLATFORM

User-centricity is the dominant feature of the OPUCE platform, which is reflected in the whole life-cycle of the service provisioning process. At the client side, diverse telecommunications and IT resources are encapsulated as visualized OPUCE Base-Services. And the end-user, even technically inexperienced one, can play with these Base-Services and link them into its desired services using an intuitive event-action composition pattern (e.g. When I

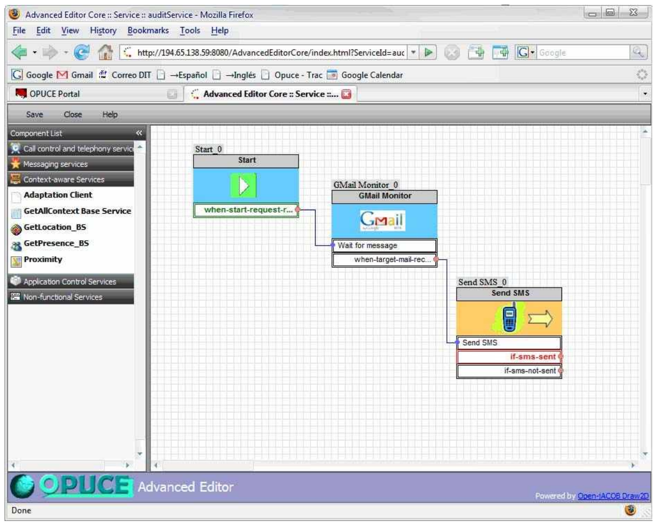

Figure 1. OPUCE Advanced Editor interface.

receive an email, send me a SMS with the content). Figure 1 is a snapshot of the user interface of OPUCE Advanced Editor, which illustrates an OUPCE *Composite-Service* (or *OPUCE service* for short) containing three Base-Services: a stereotyped *Start* Base-Service used to initiate a service process, a *Gmail Monitor* Base-Service used to check a specific Gmail account, and a *Send SMS* Base-Service. The links between Base-Services reflects the following logic: *When receiving start request from the user, monitoring the Gmail account for incoming emails; when a target email arrive, send SMS*. In parallel with the Advanced Editor, there is also a mobile version that runs on smart handheld devices. Figure 2 gives a snapshot of the OPUCE Mobile Editor.

At the server side, OPUCE supports a user-centric service management strategy, which means end-users are granted the freedom to manage the lifecycle of their own services, from deploying, activating, publishing, to unpublishing, deactivating, undeploying the services.

Next we briefly describe the general architecture of the OPUCE platform. As illustrated in Figure 3, the core sub-systems include:

- *Portal* – It integrates the graphical user interfaces of different modules through which end-users and administrators can perform management and service creation tasks. The web-based Advanced Service Editor is the main working space for end-users to create OPUCE services.
- *Mobile/Basic Service Editor* – It enables end-users to compose OPUCE services on smart handheld devices such as PDA. It is a useful complement to the Advanced Service Editor that must run on a computer. Although Mobile Service Editor does not support complex control-flow constructs, it does have a wizard helping to chain OPUCE Base-Services automatically.
- *Service Lifecycle Management* – It manages the entire lifecycle of all services, including deployment, provisioning, and monitoring. It also hosts the Service Description Translator (*SD Translator*), which is responsible for translating the high-level OPUCE composite service description into executable BPEL [6] scripts.
- *Service Execution Environment* – It hosts and runs the real executable code of both OPUCE Base-Services and OPUCE services. OPUCE Base-Services can be implemented using different technologies such as JSLEE, J2EE, .NET or even legacy technologies as long as they are encapsulated as WSDL-interfaced Web services. The main components of the Service Execution Environment are the Event Gateway and the BPEL Engine. The Event Gateway reflects the event-driven nature of the telecommunications applications. It is the endpoint for all the event notifications generated by the Base-Services and will forward these notifications to the BPEL Engine which serves as the service logic execution engine.
- The other components in the OPUCE architecture include:
  User Information Manager – It stores information about users. Specifically, five groups of user information, including User Profile, User Context (such as location, presence, device capability, network condition), Service Usage, Device Usage, and User Preferences are kept to be used by the Service Advertising and Context-Awareness components.
  Service Advertising – It recommends services to end-users based on both explicit user subscription to service categories or keywords, and intelligent matching of user profiles with service descriptions.
  Service Repository – It stores the descriptions of both the OPUCE Base-Services and OPUCE services The repository comes with a powerful search capability supporting both keyword-based and semantic search.
  Context-Awareness – It allows the dynamic adaptation of services according to the information retrieved from the profiles within the User Information Manager. It performs the necessary changes in the service logic and/or the data handled

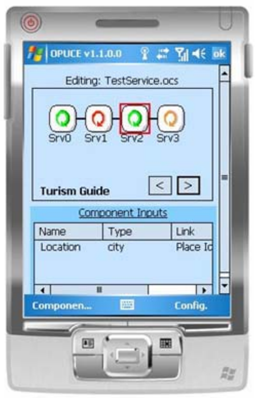

Figure 2. OPUCE Mobile Editor interface.

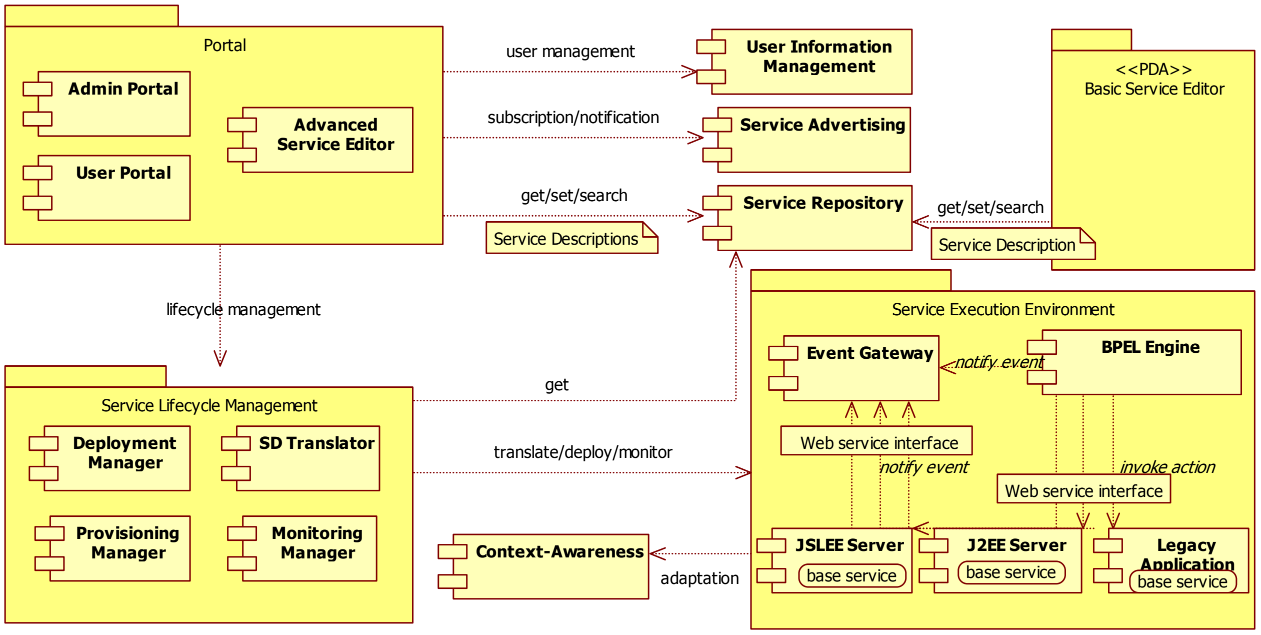

Figure 3. OPUCE platform architecture.

in order to adapt the service to the context of each user.

In the next section, we describe the models of OPUCE Base-Services and OPUCE services in detail.

III. OPUCE Service Models

A. *OPUCE Base-Services*

Figure 4 illustrates the OPUCE Base-Service model specified in UML class diagram with an example describing the *SendSMS* base service. The rationale behind this model is a simple event-driven paradigm: every base service can receive event requests and data inputs from the external and the event will trigger an action and the action will also generate new events and data outputs. With this model, event-based asynchronous Telecommunications services can be naturally described; call-return type synchronous services can also be accommodated by treating the call and return as special events.

As we can see in Figure 4, input and output parameters can be specified using the concept Property. A property uses access attribute to indicate that it is an input parameter (with *read access*), output (with *write access*), or both input and output parameter (with *read/write access*). Referring to Yahoo Pipes, our type system includes the following types: *String, Boolean, Number, URI,* and *Location*. For example, *SendSMS* has a *To* property with *URI* type (suppose we use SIP URI to identify a user), a *Content* property with *String* type, and an *UnlimitedContent* property with *Boolean* type. Finally, every property can have a constraint, e.g., the content of a SMS should not be empty.

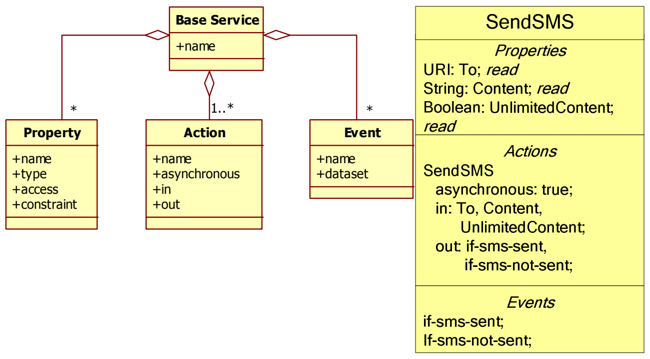

Figure 4. OPUCE Base-Service model.

A Base-Service should at least have one *action* to do. The action can be *asynchronous* or *synchronous*, with input data and also events that may be generated during the execution of the action. If there is any output, it is contained in the corresponding event with *dataset* attribute. For example, *SendSMS* has an asynchronous action also called *SendSMS*, which uses *To*, *Content*, and *UnlimitedContent* as inputs and may generate *if-sms-sent* event or *if-sms-not-sent* event.

As we mentioned in the previous section, OPUCE Base-Services can be implemented using diverse technologies as long as it is encapsulated as Web services.

B. *OPUCE Composite-Services*

The OPUCE Composite-Services/services are based on a simple and intuitive event-action composition model. That is, a start event will trigger an action of a Base-Service, and the new events generated in the action will be used to trigger new actions in other Base-Services, and so on. Finally, a final event will cause the termination of the service. For example in Figure 5, a *Start* event triggers the

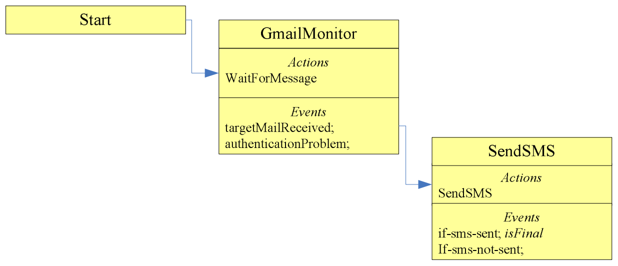

Figure 5. OPUCE event-action composition model example.

WaitForMessage action of the *GmailMonitor* Base-Service, and the *targetMailReceived* event generated by the action is used to trigger the *SendSMS* action, and the whole composite service will terminate when the final event, *if-sms-sent*, is generated.

Other features of the OPUCE service model include: implicit dataflow and control-flow patterns.

Implicit Dataflow – As we can find in the Base-Service model, the data generated by the action is encapsulated in the corresponding event. For example, *WaitforMessage* action keeps the information about the received mail in the *targetMailReceived* event. So when this event is sent to the *SendSMS* action, it will automatically get the inputs from the event without explicit data copy. Implicit dataflow is very helpful for the novice users, which enables them to build simple OPUCE services just by a few event-action connecting steps. It is worth noting that Microsoft Popfly also uses the implicit dataflow approach.

Control-flow Patterns – Except for the above mentioned simple event-action composition model, two kinds of control-flow blocks are introduced to define complex OPUCE services:

- *IF block* – This block will fire one of two mutually exclusive events when an event occurs, based on a condition set by the user.
- *JOIN block* – This block implements a logical AND operator between events, and will fire an event as soon as all its actions have been invoked.

With these two control-flow blocks, the user can define composite services base on the following patterns:

- *On-event-if-then-else pattern* – One event is connected to an IF block; the IF block has two outgoing connections to actions. For each event invoking the action of the IF block, only one outgoing event is fired, depending on the condition.
- *On-multiple-events-action pattern* – Two or more events are connected to the actions of a JOIN block.
- *On-event-multiple-actions pattern* – One event is directly connected to more than one action.

To execute an OPUCE service created following the above composition model, we need to translate the service description into a BPEL script that interacts with the event gateway. The execution model of an OPUCE service is based on the following algorithm [7]:

1. Wait for an event notification;
2. If this event is final, terminate the service;
3. Uniquely identify the action or control-flow block associated with the incoming event;
4. Perform the required action or block;
5. Go back to step 1.

A flow chart of the execution algorithm is shown in Figure 6.

At the first step, the algorithm waits for any incoming event notification, originating from a Base Service Instance. During this listening phase, the Service Execution Environment does not actually do anything other than waiting for a notification; however, the whole OPUCE service could be considered running: in fact, some of the base service instances could be performing their functionalities without firing any event. At some time, however, one of the running behaviors of the Base-Service Instances could generate an event which must be handled by the service logic. This means that, in the high-level service logic description, that particular event has an outgoing connection (or is marked as final). At this point the event is notified to the Service Execution Environment, which will proceed to the second step.

As soon as an event is notified, the algorithm identifies the action or control-flow block associated to the received event and delegates its execution to the BPEL engine. As to the translation between control-flow blocks and BPEL, we map the action call to the BPEL *invoke* activity, map the *On-event-if-then-else* pattern to the *switch* activity, and map the *On-event-multiple-actions* pattern together with the *On-multiple-events-action* pattern to the *flow* activity in BPEL.

IV. VALIDATION

OPUCE is supported by an underlying infrastructure where services are deployed and executed. In its early stages this infrastructure was realized as an open source implementation of IMS (http://www.openimscore.org). Recently, and due to the success of our proposal, OPUCE was launched as a beta version integrated with Open movilforum (http://opuceportal.movilforum.com). The integration has been achieved by wrapping the APIs offered by Open movilforum as Web Services, so that they can be integrated in OPUCE as Base-Services. For example, our initial beta pilot includes APIs for locating mobile devices based on CellID information, for using a network-hosted agenda that synchronizes with the mobile device local agenda, for sending SMS, and even for monitoring the reception of new SMS; as well as others from the Internet world for sending email messages, monitoring e-mail accounts, reading content from an RSS feed, etc. The network operator Telefónica Spain (http://www.telefonica.com) has assigned a premium number to our OPUCE test-bed, which, leveraging on the SMS monitoring Base-Service, results specially useful for creating services “triggered” by an SMS message sent by an end-user. For instance, the EuroNewsSMS service, which is activated when a user sends an SMS with the word “EuroNews”, sends back an SMS containing the news that EuroNews (a European TV) publishes on its RSS.

Before you begin to format your paper, first write and save the content as a separate text file. Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

V. Discussion and Conclusion

The approach presented in this paper applies the notion of enabling non-technically skilled end-users to create, manage and share personalized mobile services. Several research projects in Europe are also aiming at providing tools and platforms to facilitate the delivery of mobile services but with different focuses: the IST-FP6 project SPICE (http://www.ist-spice.org) provides a full-fledged service platform from service creation to service management and execution with focus on providing a seamless way to deliver services over heterogeneous execution platforms, network and terminals; IST-FP6 project MAGNET (http://www.telecom.ece.ntua.gr/magnet) aims at developing the concept of flexible Personal Network (PN) that supports resource-efficient, robust, ubiquitous service provisioning in a secure, heterogeneous networking environment for nomadic users; IST-FP6 project PLASTIC (http://www.ist-plastic.org) provides a layered middleware leveraging B3G networking capabilities (e.g. multi radio, customized protocols for B3G, multicast, multi-network routing); IST-FP6 project SMS (http://www.ist-sms.org) provides tools to simplify the discovery, use, trust and setup issues around mobile services.

From the perspective of programming models, user centric Web application authoring tools including Taverna [8], a Web services workflow environment in the bioinformatics community, and Yahoo Pipes both explicitly use data dependencies to connect operations, reflecting the resource-centric nature of many Web and bioinformatics applications. Microsoft Popfly and Bite [9] use control-flow to connect operations, but the output of its source operation can be implicitly used as part of the input of its target operation, which simplifies the definition and handling of data in common programming languages. As for OPUCE, we use an event-driven programming model to fit to the asynchronous nature of telecommunications services and applications, and also adopt the implicit dataflow mechanism.

In this paper, we have presented the user-centric mobile service creation approach of the OPUCE platform. Especially we discuss the architecture, service model and validation aspects the platform. Currently, the OPUCE platform has been integrated into the mobile open source community Open movilforum, and a usability testing plan is underway.

Acknowledgment

This work is funded by research project OPUCE, under Information Society Technologies (IST) priority of the 6th Framework Program of the European Community, Contract No. 34101. We thank all our partners in the project for their valuable comments to this paper.

References

[1] A. Cuevas, H. Einsiedler, J.I. Moreno, and P. Vidales, “The IMS service platform: A solution for Next-Generation Networks operators to be more than bit pipes”, IEEE Communications Magazine, 44(8), Aug 2006, pp. 75-81.

[2] R. Chen, V. Shen, T. Wrobel and c. Lin, “Applying SOA and Web 2.0 to telecom: Legacy and IMS next-generation architectures”, in 2008 IEEE International Conference on e-Business Engineering *(ICEBE'08)*, Xian, China, Oct. 2008, pp. 374-379.

[3] G. Camarillo and M.A. García-Martín, The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the cellular worlds, John Wiley & Sons, 2004.

[4] T. O'Reilly, What is Web 2.0: Design patterns and business models for the next generation of software, 2005, available online at http://www.oreilly.com/pub/a/oreilly/tim/ news/2005/09/30/what-is-web-20.html.

[5] J.C. Yelmo, R. Trapero, J.M. Del Álamo, J. Sienel, M. Drewniok, I. Ordas, K. Mccallum, “User-driven service lifecycle management: Adopting Internet paradigms in telecom services", in 5th International Conference on Service Oriented Computing (ICSOC’07), Vienna, Austria, Sept. 2007, LNCS 4749, pp. 342-352.

[6] A. Alves, et al (Ed.), Web Services Business Process Execution Language Version 2.0, OASIS WSBPEL TC, Jan 2007, available online at http://docs.oasis-open.org/ wsbpel/2.0/.

[7] D. Cipolla, F. Cosso, M. Demartini, M. Drewniok, F. Moggia, P. Renditore, J. Sienel, Web Service based asynchronous service execution environment, in 1st International Workshop on Telecom Service Oriented Architectures, Vienna, Austria, Sept. 2007.

[8] T. Oinn, M. Addis, J. Ferris, D. Marvin, A. Wipat, P. Li, T. Carver, Delivering Web Service coordination capability to users, in 13th International World Wide Web Conference (WWW2004), New York, USA, May 2004, pp. 438-439.

[9] F. Curbera, M. Duftler,R. Khalaf, D. Lovell, Bite: Workflow composition for the Web. In 5th International Conference on Service-Oriented Computing (ICSOC’07), Vienna, Austria, Sept. 2007, LNCS 4749, pp. 94-106.