End-User-Oriented Telco Mashups: The OMELETTE Approach

Olexiy Chudnovskyy
Chemnitz University of Technology
olexiy.chudnovskyy@cs.tu-chemnitz.de

Tobias Nestler
SAP Research
tobias.nestler@sap.com

Martin Gaedke
Chemnitz University of Technology
martin.gaedke@cs.tu-chemnitz.de

Florian Daniel
University of Trento
daniel@disi.unitn.it

José Ignacio Fernández-Villamor
Universidad Politécnica de Madrid
jifv@gsi.dit.upm.es

Vadim Chepegin
TIE Holding R&D
vadim.chepegin@tieholding.com

ABSTRACT

With the success of Web 2.0 we are witnessing a growing number of services and APIs exposed by Telecom, IT and content providers. Targeting the Web community and, in particular, Web application developers, service providers expose capabilities of their infrastructures and applications in order to open new markets and to reach new customer groups. However, due to the complexity of the underlying technologies, the last step, i.e., the consumption and integration of the offered services, is a non-trivial and time-consuming task that is still a prerogative of expert developers. Although many approaches to lower the entry barriers for end users exist, little success has been achieved so far. In this paper, we introduce the OMELETTE[1] project and show how it addresses end-user-oriented telco mashup development. We present the goals of the project, describe its contributions, summarize current results, and describe current and future work.

Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications; D.2.11 [**Software Engineering**]: Software Architectures—*Service-oriented architecture (SOA)*; C.2.4 [**Distributed Systems**]: Distributed applications

General Terms

Theory

Keywords

Mashup, Telco Services, EUD

[1]Open Mashup Enterprise service platform for LinkEd data in The TElco domain

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

1. INTRODUCTION

Over the last few years, a large number of novel telephony and messaging services have become available to the Web community. These so-called telco services provide unique capabilities regarding voice, video and data transfer [5]. From the usage point of view, telco services are invoked using provider-specific Web APIs and shield the developer from the complexity and protocol mismatch of operator networks. Although several initiatives that aim to standardize cross-provider service interfaces[2] exist, the integration of telco services into Web applications is still challenging. First, data streaming, real time communication and conferencing still lack suitable technologies and integration techniques. Second, without appropriate tools and platforms, the deployment, management and execution of converged services remains a very complex and costly task. Finally, current state-of-the-art platforms do not support end users adequately, so the users are hindered in building their solutions without specific programming skills.

To meet these challenges the FP7 project OMELETTE specifically focuses on the needs of the telco domain and end user development. Based on the technologies and lessons learned from Web mashups, the project investigates into innovative ways of letting end users create their own collaboration platforms. In particular, OMELETTE contributes to the state of the art as follows:

1. OMELETTE provides a set of tools and components to support the development and execution of telco mashups. OMELETTE introduces an open extensible architecture based on several open-source projects. Special attention goes to interoperability among mashups environments, which is achieved by standardizing the models of produced artifacts.

2. OMELETTE assists end-users and developers in their development task by means of a recommendation and an automatic composition engine. The recommendation engine is able to interactively provide help during manual development; the automated composition engine is able to take over some of the development work.

[2]http://oneapi.gsmworld.com/

Both instruments guide developers/end users, allowing them to expand their development capabilities and to develop faster.

3. OMELETTE contributes to several open source projects (Apache Wookie and Apache Rave) as well as emerging W3C standards (W3C Widgets and WebID). Apache Rave will be extended with the capability to discover widgets from remote sources, share workspace models with other environments, deliver development recommendations, and compose workspaces out of user goals. Apache Wookie will support extended widget logic, in particular, inter-widget communication.

2. MOTIVATING SCENARIO

To motivate the research on telco service integration and dedicated mashup tools, consider the following scenario.

Mike is working for a small company in Portugal that produces wooden toys for kids. He is responsible for the launch of a new product that will be distributed via large chain stores all over Europe. To discuss the latest orders from the stores and decide on a possible discount, Mike needs to organize a meeting among him, a representative of that chain who is located in Paris and his colleague from the sales department who is currently on a business trip. To hold a live meeting among all geographically distributed participants, Mike would like to have an adaptable web-platform that provides configurable access to data of the existing enterprise systems (e.g. CRM, ERP) and a seamless integration of collaboration and communication functionalities. He wants to create a view on the data required for the next meeting and share it with others. Before making decisions, Mike asks colleagues to provide feedback. As such, if no or only limited Internet access is available, participants should use other communication channels, such as telco operator network. The internal web portal of Mike's company seems to be a promising candidate to become such a platform. It provides a dashboard with widgets from an integrated catalog. However, the capabilities of the portal are very limited in terms of extensibility and communication support. In addition, most of the existing widgets provide only very generic data and cannot be combined or connected.

The presented scenario reveals the following challenges:

- Native integration of telco capabilities into existing web applications and portals to avoid extra software solutions and minimize distraction by switching tools and contexts.
- Easy extensibility and inter-widget communication for creating collaboration spaces of widgets.
- Lower the barrier for end-users to create and integrate widgets, i.e. creating mashups that serve users' needs.

3. THE OMELETTE APPROACH

Instead of proposing yet another mashup editor similar to Yahoo! Pipes, the idea behind the OMELETTE approach is to leverage as much as possible on existing mashup environments. The goal is to come up with an integrated environment that is able to provide each role in the mashup development life cycle with the right tool matching the role's individual skills. The idea is in line with the spirit of mashups themselves, i.e., reuse, and aims at leveraging on previous experiences and results, also stemming from other projects. OMELETTE creates an architecture based on open standards and interfaces, where components can be replaced or integrated in a seamless way. Figure 1 gives an overview of the OMELETTE toolkit and shows how end-users interact with different mashup environments.

3.1 The Live Environment

End users are provided with a composition environment for widgets (the so-called live environment), which can be placed on a canvas, synchronized among each other, and shared with other users. Widgets provide user interface to complex application logic, aggregated data services or telco functionality. The composition paradigm is of low complexity and can be seen as the fusion of the current portal and mashup technologies. The key innovation of the live environment - which also gives it its name - is its capability to support a "live" composition approach, in which widgets, once placed on the composition area (the so-called workspace), are immediately executed, and users can inspect them and interact with them. Furthermore, the live environment enables widgets to communicate with each other, providing added value to the otherwise static set of widgets [7]. This is fundamentally different from current practice, in which composition is typically based on explicit composition models [3]. Instead of manipulating abstract modeling constructs (which are relatively intuitive to experts, but may not have any meaning to end users) representing software artifacts, the live environment allows end users to directly manipulate the software artifacts (the widgets). The Live Environment is based on the Apache Rave and Apache Wookie projects and makes use of open Web standards (HTML5, advanced CSS features, W3C Widgets, Device APIs). The Live Environment runs in both regular and mobile web browsers, making use of available device capabilities. Apache Rave provides a lightweight Java platform to host, serve and aggregate widgets (OpenSocial Gadgets and W3C Widgets). It manages workspace instances and orchestrates background services to build a complete workspace for a given user. OMELETTE extends Apache Rave with an inter-widget choreography service, a dynamic loader and interfaces toward the recommendation and automatic composition engines. Apache Wookie acts as a widget repository, where widgets are published so that they are available for the Live Environment. It manages widget instances and supports inter-widget choreography by providing additional messaging facilities.

3.2 Recommendation and Automatic Composition Engine

OMELETTE recognizes that it is not enough to simplify technology in order to enable end users to develop mashups. End users (and partly also developers) need to be assisted in their development task, either because they are not yet acquainted with their mashup environment or because they simply are not able to find a good solution on their own (e.g., the right components or the right combination thereof).

OMELETTE therefore equips its mashup environments with two novel features: a recommendation engine that is able to interactively provide end users or developers with help during manual development, and an automated composition engine that is able to take over some of the development work (Figure 1). The goal of these two instruments is

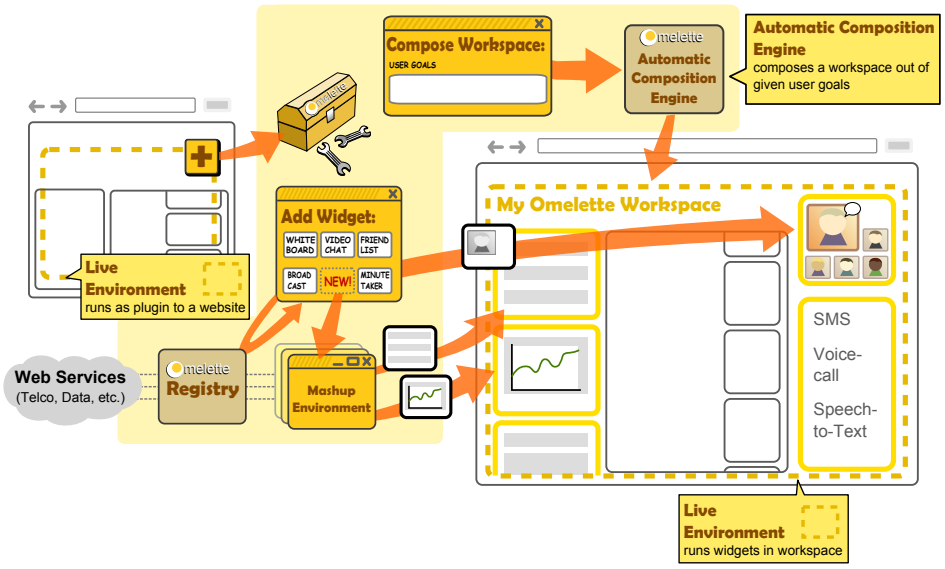

Figure 1: The OMELETTE approach overview

to guide developers and end users, allowing them to expand their development capabilities and to develop faster.

The Recommendation engine, suggests users with the set of composition patterns, e.g., recommendations in the form of possible widgets to add or configuration options for widgets, which help users to define their workspaces [1]. Based on the partial workspace model, the Recommendation engine queries a pattern knowledge base, which contains composition patterns obtained by mining the repository of workspace models. The knowledge base is used to compute a ranked and contextualized list of composition patterns aimed at helping the user complete his/her workspace under development. Retrieved patterns are shown to the end users in a dedicated recommendation panel next to the workspace. Once the user has selected a recommendation, the chosen pattern is weaved into the current workspace model, i.e., it is connected to the existing workspace model.

The automatic composition engine allows users to focus on what (the goal) the mashup is supposed to do instead of on how it will do so (in terms of implementation). It provides a dialog-based interface to specify user goals and finally suggests a mashup, taking into account possible user preferences. Based on the user profile settings and preferences, the automatic composition engine personalizes the workspace and configures widgets to take into account the user's location, his/her preferred language, etc.

3.3 Mashup Environments

Since mashups can be very complex applications, complexity must however reside somewhere. OMELETTE recognizes this, and moves complexity inside the widgets. That is, instead of exposing to end users how to synchronize a widget with an underlying web service, e.g., providing updated stock prices, we shield the users from these details and hide them inside the widgets. Developing widgets becomes therefore a crucial aspect in the OMELETTE ecosystem, a fact that OMELETTE takes into account by fostering the use of dedicated mashup environments for skilled developers, aiding them in the development of also complex application logic. The idea, hence, is to provide developers with another, more expressive and more complex mashup environment, which allows them to compose generic web resources (e.g., SOAP/WSDL web services, RESTful services, RSS/Atom feeds, or reusable UI components) and - more importantly - to publish the result as a widget that can then be used by the end users in their workspaces (Figure 1). Given the complexity of developing widgets, this kind of mashup environment will be based on models, rather than on a live development paradigm.

OMELETTE approaches this problem by extending two existing mashup environments, in order (i) to take into account the peculiarities of telco services and (ii) to be able to produce mashups in output that comply with the W3C widget family of specifications. The first one, the ServFace Builder [6] is a web-based authoring environment that aims to support skilled web users and developers in the creation of widgets based on annotated web services (SOAP/WSDL and OData/CSDL). It follows the concept of service composition at the presentation layer, a design paradigm developed in the course of the EU FP7 project ServFace[3]. The other editor, MyCocktail, was developed in the context of the EU FP7 project Romulus[4]. MyCocktail focuses on the development of mashups that integrate RESTful services and UIs. It allows users to combine information obtained from different services, to modify them with suitable operators, and to present them with a wide variety of UI renderers.

Although both the tools have a different focus, the output is in both cases a reusable and interoperable artifact - W3C widgets - which plays the important role of mediator between the experienced developer (mashup environment) and the less skilled user (live environment).

[3]http://www.servface.eu
[4]http://www.ict-romulus.eu

3.4 Mashup Registry and Automatic Discovery Engine

The Web is a continuously evolving and growing ecosystem with a huge variety of possible resources of interest to mashup developers and end users. While this is in general a benefit, it however also implies that the developers and users may simply not be aware of the novel possibilities offered, e.g., by a new messaging services deployed on the Web. OMELETTE provides a (semi)automatic discoverer for the identification of such resources, in order to bring them to the attention of the OMELETTE users. Specifically, an expert can evaluate their suitability for inclusion into the OMELETTE registry and, hence, decide whether to make OMELETTE aware of them or not.

The discovery process is performed on the basis of the Scraping Ontology [4] and the open source screen scraper called Scrappy[5], which crawls HTML-based descriptions of services, data mashups and widgets and converts them into semantically enriched ones. All component types become ready for semantic search queries and build the basis for an automated composition process. Up to now, we have crawled more than 10000 component descriptions, coming from different online repositories, such as ProgrammableWeb, Opera Widgets and Yahoo! Pipes. The OMELETTE registry is implemented on top of the WebComposition Data Grid Service [2], a generic data access component with primary focus on content and meta-data management. It uses a common component model that is employed to describe various component types. For this purpose, the information provided by different component repositories has been combined into a unified schema. The model includes technical, business and trust aspects about the components, as well as a categorization scheme for additional selection.

4. DEMONSTRATION

During the demo poster session, we plan to present how a simple telco mashup for the online meeting scenario can be developed using the OMELETTE toolkit. In particular, we will show how an end user can create a new workspace using the OMELETTE Live Environment. The starting point will be the OMELETTE mashup library containing a set of components deployed together on the OMELETTE platform. Jointly with the audience, we will assemble and configure several telco widgets, implementing advanced communication and collaboration functionalities (SMS, telephone conference and telephone-based voting widgets). Using MyCocktail we will show, how telco and data services can be combined to create new widgets: one to visualize product data and share it with other workspace participants and another one to display user contacts on a map and establish a telephone connection between them. As a result we will get a mashup that is not only able to combine services and UIs but also to establish a telephone connection between geographically distributed participants. After adding further widgets, such as a shared dashboard and a chat widget, the mashup will provide even more facilities to work collaboratively and to achieve the meeting's goals faster.

5. CONCLUSIONS AND OUTLOOK

In this paper, we presented the OMELETTE approach to end-user-oriented telco mashups. We showed how telco services can be integrated with other Web sources in a seamless way that can be mastered even by end users. The project is still ongoing, but several prototypes are already ready for demonstration[6][7][8]. Currently, we are working on the integration of all tools and components into one holistic ecosystem, but the demonstration of the individual tools that are already available already gives good insight into where OMELETTE is headed at. Besides some technical aspects that need to be finalized, we also plan to conduct extensive user studies with end users (we partly already started this task), in order to elicit a complete picture of end user requirements and to fine-tune the OMELETTE instruments. In this context, we see the demonstration of OMELETTE at WWW 2012 as particularly valuable.

6. ACKNOWLEDGMENTS

This work was supported by funds from the European Commission (project OMELETTE, contract no. 257635)

7. ADDITIONAL AUTHORS

Additional authors: José Angel Fornas (Logica Spain, email: jose.angel.fornas@logica.com), Scott Wilson (University of Bolton, email: scott.bradley.wilson@gmail.com), Christoph Kögler (T-Systems MMS, email: Christoph.Koegler@t-systems.com) and Heng Chang (HUAWEI Technologies Co.LTD., email: changheng@huawei.com).

8. REFERENCES

[1] S. R. Chowdhury, F. Daniel, and F. Casati. Efficient, Interactive Recommendation of Mashup Composition Knowledge. *ICSOC 2011*, pages 374–388.

[2] O. Chudnovskyy and M. Gaedke. Development of Web 2.0 Applications using WebComposition / Data Grid Service. *Service Computation 2010*, pages 55–61.

[3] A. De Angeli, A. Battocchi, et al. End-User Requirements for Wisdom-Aware EUD. *IS-EUD 2011*, pages 245–250.

[4] J. I. Fernández-Villamor, C. A. Iglesias, and M. Garijo. A Semantic Scraping Model for Web Resources – Applying Linked Data to Web Page Screen Scraping. *ICAART (2) 2011*, pages 451–456.

[5] H. Gebhardt, M. Gaedke, F. Daniel, et al. From Mashups to Telco Mashups: A Survey. *IEEE Internet Computing, 2011. http://doi.ieeecomputersociety.org/10.1109/MIC.2012.19.*

[6] T. Nestler, M. Feldmann, et al. The ServFace Builder - A WYSIWYG Approach for Building Service-Based Applications. *ICWE 2010*, pages 498–501.

[7] S. Wilson, F. Daniel, U. Jugel, and S. Soi. Orchestrated User Interface Mashups using W3C Widgets. *ICWE Workshops 2011*, pages 49–61.

[5] http://github.com/josei/scrappy

[6] http://demo.ict-omelette.eu/portal

[7] http://demo.ict-omelette.eu/MyCocktail/

[8] http://demo.ict-omelette.eu/wookie