Editor: Gustavo Rossi • gustavo@lifia.info.unlp.edu.ar

From Mashups to Telco Mashups: A Survey

Hendrik Gebhardt and Martin Gaedke • *Chemnitz University of Technology*

Florian Daniel, Stefano Soi, and Fabio Casati • *University of Trento*

Carlos A. Iglesias • *Universidad Politécnica de Madrid*

Scott Wilson • *University of Bolton*

Given their increasing popularity and novel requirements and characteristics, telco mashups deserve an analysis that goes beyond what's available for mashups in general. Here, the authors cluster telco services into different types, analyze their features, derive a telco mashup reference architecture, and survey how well existing mashup tools can respond to these mashups' novel needs.

Web mashups integrate data, application logic, and parts of UIs sourced from the Web to create new, composite applications.[1] A typical example is the housingmaps.com application, which integrates housing offers from craigslist.com with a Google map. Although mashups are often coded manually, so-called mashup tools or platforms aim at development paradigms that don't require programming skills and, hence, target end users. However, the scope of the instruments conceived so far is both broad and technology-centric, which limits these tools' ability to cater to domain-specific features and needs when it comes to developing concrete applications.

For instance, interconnecting people, possibly via different channels such as voice, video, or IM, in both fixed and mobile settings is still a difficult and time-consuming endeavor — if it's feasible at all. In fact, the peculiarities of the telecommunications (telco) domain, which specifically focuses on transmitting data to enable communication and collaboration among people, haven't percolated into existing mashup tools. Features such as multidevice deployment, audio and video (A/V) streaming, distributed session management, and live collaboration aren't supported in an integrated fashion and are thus unavailable to the general public. The same holds for quality of service (QoS), the key nonfunctional requirement in telco.

A primary reason for this weak support for telco features in mashups is a general lack of understanding about what they are and how they can be developed. To foster research in this area and advance current mashups toward the telco domain, we review the state of the art in telco services, derive a reference architecture for mashup platforms, and compare it with existing platforms. We also identify challenges and open research questions that are specific to the telco domain.

Scenario and Challenges

To better understand what a telco mashup might look like, consider the following application scenario. Several consultants from a multinational firm are discussing the technical architecture for a project proposal. They use a corporate collaborative environment consisting of a multichannel Web application and a shared whiteboard. All participants connect to the application via different clients: Marco via a smartphone using

 Published by the IEEE Computer Society 1089-7801/12/$31.00 © 2012 IEEE

a mobile Web browser, Steve using a desktop Web browser, Jürgen with a tablet using a mobile Web browser, and Maria through a traditional mobile phone using the phone's built-in capabilities.

The first three consultants use Web-based IM, while the fourth uses SMS messaging. The collaborative environment provides a telco mashup that combines these two communication channels with the whiteboard. After a while, the consultants decide to switch to voice, using a facility of the collaborative environment based on a dedicated voice-over-IP (VoIP) service. Maria can either dial in to the ongoing session, or the application calls her on a known number. Because Maria can't draw with her phone, she sketches her ideas on paper and sends a photo taken with her phone's built-in camera via MMS to the telco mashup, which renders it to the other consultants.

This scenario is rather complex, and supporting it requires a telco-ready platform. Devising such a platform is nontrivial and requires a thorough understanding of both telco services and APIs, and telco mashups.

Telco Services and Device APIs

In our example scenario, some features of the collaborative environment must interact with remote software services (functionalities accessed via the Internet using protocol-based message exchanges) that provide telco support (such as the VoIP service). Others require local device capabilities (for example, the phone's camera). We call the former *telco services* (software services that provide communication and collaboration support) and the latter *device APIs*.

Telco Capabilities

To enable user-generated, value-adding services, telco companies such as Orange (www.orangepartner.com/site/enuk/access_orange_apis/p_access.jsp), Telefonica (https://bluevia.com/en/), or Deutsche Telekom (www.developergarden.com/apis/) invest in *service delivery platforms* (SDPs) that expose network capabilities to third parties. At these platforms' core is the *telecommunication application server*, which is based on technologies such as Session Initiation Protocol (SIP) servlets, JAIN SLEE (the Java APIs for Integrated Networks Service Logic Execution Environment), Parlay-X, or the IP Multimedia Subsystem (IMS). Although these telco services are evolving slowly, nontelco companies such as Google, Yahoo, Twilio, and Tropo already provide similar services for managing calls, messaging, or presence.

We distinguish three types of telco services, depending on the networks used and their purpose:

- *Internet* telco services operate exclusively on the Internet, using it as communication infrastructure. Examples include VoIP and IM.
- *Converged* services operate across the Internet and operator networks, mediating between different networks and communication protocols. A VoIP call to a mobile phone or fixed-line phone would be a converged service.
- *Signaling* services provide access to a network operator's signaling infrastructure. Notifying a mobile phone about an incoming call or negotiating QoS parameters are examples of signaling operations.

We also determined three dimensions that we can use to analyze telco services. Consider a developer who wants to integrate telco services or APIs into his own mashup. The developer must first understand what a given service or API actually provides — that is, what *communication paradigm* it supports. After identifying a candidate service, the developer will typically want to know how to use — that is, interact with — it in the mashup; we call this the service's *interaction paradigm*. Finally, to further discriminate services based on nonfunctional properties, the developer must know at what cost or service levels the candidate service is delivered, determined via *service-level agreements* (SLAs).

The communication paradigm describes the direction of the communication channel and the number of parties involved. Unlike with commonly used Representational State Transfer (REST) APIs, cardinality plays a decisive role in communication with a telco service. Whether we can use one-to-one or one-to-many communication depends on the service itself. In both cases, one service is the sender, but there are a different number of receivers. Another important property of telco services is their synchronicity. Whereas voice and video communications need synchronous communication (participants' copresence is required), messaging is asynchronous (participants can write and read messages at different times).

The interaction paradigm looks at how telco mashups handle the interaction with a service or API. A subdimension, *binding*, describes how a telco mashup transfers content — that is, voice and video services are based on streaming data because delays in synchronous communications are undesirable or even prohibited (as with real-time streaming). Another subdimension is *internal state management*, which instantiates and manages resources and communication channels for services. For instance, establishing a Global System for Mobile Communications (GSM) phone call implies acting first on a control channel to obtain a separate stream for the actual communication. In telco mashups, where

we might have multiple parallel communication connections open simultaneously, this demands suitable stream state management. All these aspects differ from common Web mashups that call, for instance, REST APIs, which are stateless.

Finally, SLAs look at QoS, cost, security, and related aspects. Common Web mashups benefit from the wide variety of free services available on the Web. In the telco domain, however, services are potentially subject to charges from network operators, based on different options (pay-per-use, subscription model, prepaid/postpaid billing models, discount plans, and so on). Thus, telco services are usually executed in a controlled environment where QoS, security, and billing are guaranteed.

Device Capabilities

Modern mobile phones have evolved into full-fledged computing devices that can both run mashups inside mobile browsers and enable them to leverage advanced device capabilities, such as a built-in camera or SMS texting. Telco mashups should thus be able to process incoming telco events (for example, phone calls or SMS messages) and allow telco mashups to access phone facilities (initiate phone calls or consult the agenda, for instance). Telco mashups can utilize these features via device APIs, which enable access to embedded cameras or webcams, location services, SMS and MMS interfaces, and the like. Cross-device standards, such as the W3C's Device APIs (www.w3.org/2009/dap/) or Widget Handset APIs from the Wholesale Applications Community (WAC; www.wacapps.net), provide APIs accessible from within regular Web applications and offer capabilities including position, accelerometer, messaging, system information, camera, and microphone. For example, an application can capture an image with the following JavaScript code using WAC standards:

```
camera.captureImage(onCapture
   ImageSuccess,onCaptureImage
   Error,{destinationFilename:
   "images/a.jpg", highRes:true});
```

Or, the application could capture the same image in HTML 5 using the W3C DAP Media Capture specification:

```
<input type="file" accept=
   "image/*" id="capture">
```

By themselves, device APIs offer nothing especially new. The challenge for telco mashups is to seamlessly merge device APIs and telco services with Web mashups in a way that isn't tied to any specific phone model, operating system, or service operator.

A Telco-Specific Mashup Platform

We define a telco mashup as a Web mashup that, in addition to optional data, application logic, and UIs, also integrates telco services or device APIs to support communication and collaboration among multiple users (as in our reference scenario) or provide them with individual telco features (such as an advanced GPS navigation mashup).

Our example scenario poses some novel requirements that existing mashup platforms don't yet support. For this scenario, a telco mashup must

- manage streaming media involving multiple users;
- integrate device APIs running inside client devices;
- manage QoS and billing;
- provide multichannel access to support different device types;
- provide multimodal access to support different interaction paradigms; and
- provide multiuser access to enable communication and collaboration.

Streaming A/V conferencing is different from just streaming a video or audio file from a Web server. In the latter case, if the stream breaks, a user can simply start it again; no special support is required from the Web server. However, during our example phone conversation, if any participant's stream breaks, the platform must be able to reconnect that user to the ongoing live conference by tracking who is involved in which conversation. So, if a telco mashup uses multiple collaborative streams, it must be able to manage each individual stream's state at the client side. This might require suitable browser extensions, client-side state management logic, or server-side logic, depending on each specific telco mashup's nature. Using device APIs doesn't directly impact the platform logic; however, if the telco mashup uses device APIs for communication among participants, the platform must provide for the necessary client–server data communication channel (for instance, to broadcast Maria's photo). Both telco services (such as streaming services) and device APIs might require monitoring and tracking QoS. More importantly, using converged and signaling services inside a mashup requires that the platform as runtime environment manages billing information, taking into account different contract options.

Multichannel access requires the platform to deliver its mashups via different communication networks and protocols, such as the Internet or conventional telco networks. Multimodal access requires support for different interaction paradigms, such as voice for Maria and traditional hypermedia for Jürgen, Marco, and Steve. Multiuser access requires not only proper user identity management and authentication but also the ability for multiple users to navigate (co-browse) the same mashup — that is, to work on the same mashup

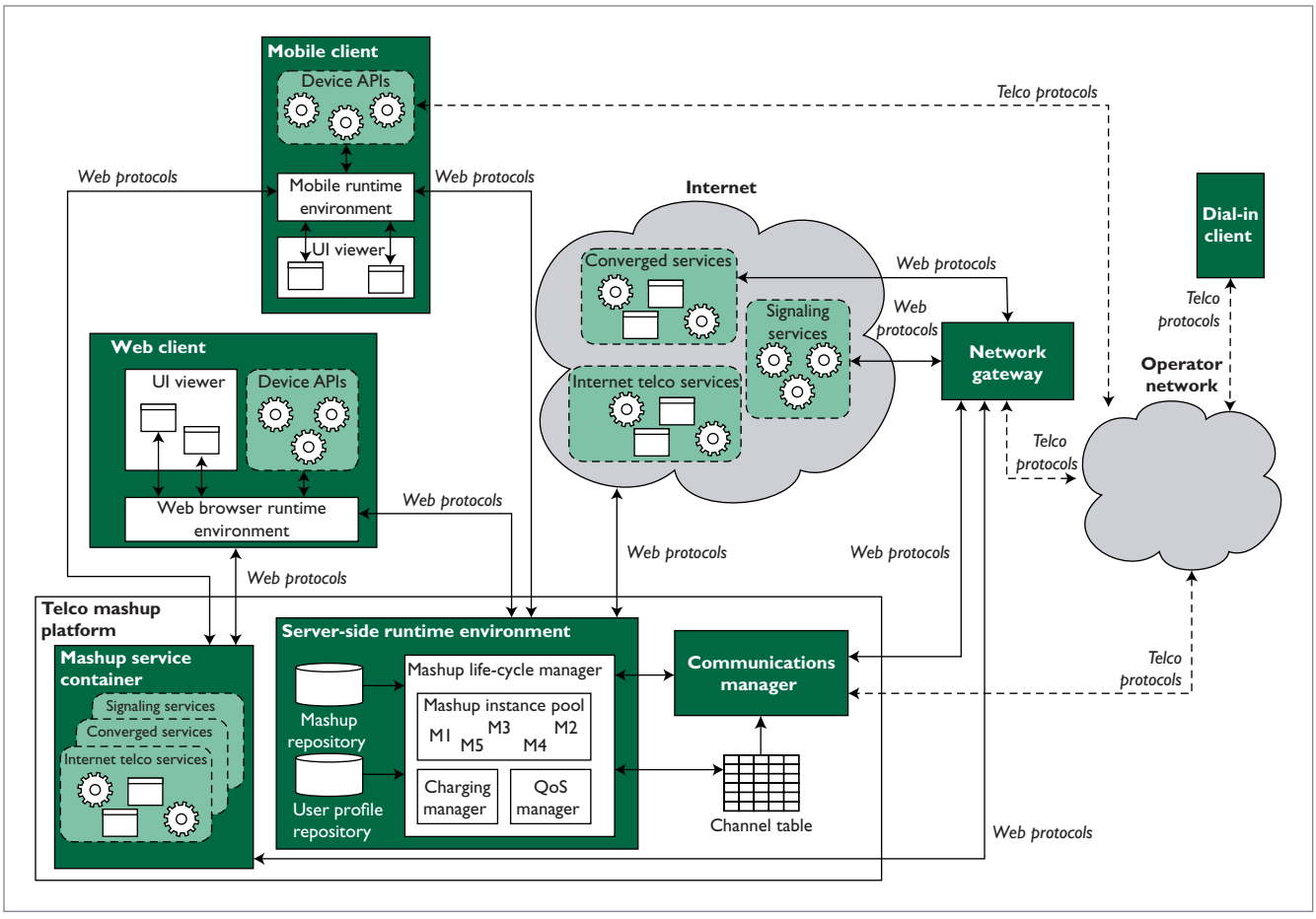

Figure 1. Telco mashup reference runtime architecture. UI components are represented as rectangles; services without UIs appear as cogwheels.

instance (to collaboratively draw the architecture picture, for instance).[2] This is different from providing each user with an independent mashup instance, which is customary in today's mashup platforms.

Understanding these subtleties is paramount to developing a telco mashup platform that can adequately support real-life telco mashup scenarios. Figure 1 shows our reference architecture for telco mashups.

Our architecture shows how to deliver telco mashups via multiple channels. Maria's phone uses a traditional operator network (such as GSM), while Steve's PC, Marco's smartphone, and Jürgen's tablet use the Internet. To allow mashups to execute the necessary converged and signaling services and to mediate between the Internet and the operator networks, the platform needs either a network gateway (typically provided by an operator or telco service provider, as described in the "Gateways in Telco Services" sidebar) or a dedicated telco application server (such as that available from www.opencloud.com).

The communications manager can provide multimodal access, allowing Maria to instantiate the mashup from her phone, even if Marco, Steve, or Jürgen aren't present. As in phone conferences, multiuser access requires a shared resource that everybody can connect to, and a respective identifier. In the architecture, we represent this resource via the mashup instances managed by the mashup instance pool, which maintains the necessary correlation and life-cycle information for the mashup UIs running inside the client devices.

To assist client devices in managing streams (both incoming and outgoing calls and Web-based streams), a channel table correlates users with their streams and channels and the respective mashup instances. The channel table also lets users book a telephone channel for A/V conferences (scheduling) and route asynchronous messages. With the channel table's help, the communications manager knows that Maria's photo is to be routed to Steve, Marco, and Jürgen and not to other platform users. Using device APIs affects the client-side runtime environments' capabilities more than those of the

Gateways in Telco Services

Telco services such as converged and signaling services are possible thanks to communication networks that actually predate the Internet; these services constitute the Internet's backbone and evolved independently. Fixed access, for instance, is provided via the Public Switched Telephone Network (PSTN, or POTS, for Plain Old Telephony System), and mobile access via the Universal Mobile Telecommunications System (UMTS), General Packet Radio Service (GPRS), and Global System for Mobile Communications (GSM) networks.

Each network uses its own protocols (such as the GSM Mobile Application Protocol) and signaling conventions (for example, Signaling System No. 7 [SS7]), which are different from the Internet's TCP/IP stack. Consequently, we can't implement a Web-based telco service that directly interoperates, for instance, with a GSM voice call. Implementing such a service requires bridging between the two network types and mediating between their respective protocols. Telco operators provide this functionality through *network gateways*, which are reachable from the Internet via standard Web protocols such as REST/HTTP or SOAP, and which expose some of the operator network's capabilities (such as the GSM voice call).

Specifically, network gateways allow access to telephony infrastructure, such as the infrastructure in Figure A. A telephony network essentially handles two pieces of information[1]: the content that's transmitted (such as voice or data) and control signals that instruct the network on how to transmit content and allocate the needed resources. In the past, control signals used in-band signaling techniques — that is, signals were transmitted together with voice or data over the same channel. Due to its intrinsic bandwidth-efficiency problems, this technique was soon replaced by out-of-band control channels and dedicated signaling protocols. SS7 is the most popular out-of-band signaling protocol. As the figure illustrates, an infrastructure can have circuit-switched technologies (such as PSTN or GSM) and packet-switched technologies (such as UMTS Packet Switched Data [PSD] or voice over IP). Circuit-switched technologies establish a dedicated circuit path (via suitable SS7 control signals) before content is transmitted. In package-switched technologies, content is fragmented into packages that can be transmitted over different paths and reassembled at the destination. Package-switched technologies, therefore, require

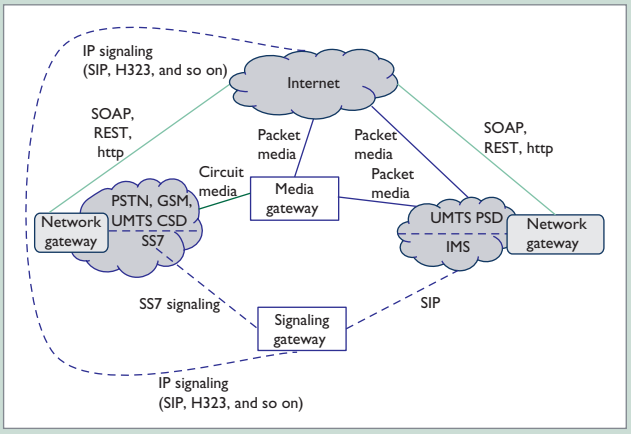

Figure A. The typical topology of today's telecommunications infrastructure. Solid lines represent content flow; dashed lines represent control signals.

establishing a session between the caller and the receiver, usually via the Session Instantiation Protocol (SIP). Media gateways provide for converting between circuit-switched and package-switched technologies, while signaling gateways do the same for control signals.

Given their crucial role in bridging the Web and telco worlds, network gateways have recently been the subject of several standardization activities, such as Parlay-X,[2] OneAPI (www.gsmworld.com/oneapi), and Wholesale Applications Community standards (www.wacapps.net). To further reduce the complexity and costs of operating heterogeneous networks, next-generation networks, such as the IP Multimedia Subsystem (IMS),[3] propose using just one set of protocols for all kinds of networks — that is, Internet protocols.

References

1. R. Bates and D. Gregory, *Voice & Data Communications Handbook*, 5th ed., McGraw-Hill Communication Series, 2007.
2. *Parlay X Web Services, Part 1: Common*, tech. specification 3GPP TS 29.199-1, 3rd Generation Partnership Project, Sept. 2004; www.3gpp.org/ftp/tsg_cn/tsg_cn/TSGN_25/Docs/PDF/NP-040360.pdf.
3. M. Poikselka, G. Mayer, and H. Khartabil, *The IMS Multimedia Concepts and Services*, 3rd ed., Wiley, 2008.

server side. These client-side components must be able to provide access to device capabilities in a way that's compatible with standard Web technologies – for example, by using Web browsers that implement the respective W3C or WAC APIs or via suitable browser plug-ins. Communication among these APIs and the server-side platform then occurs via standard Web protocols. A dedicated QoS manager and a charging manager handle QoS and billing, respectively.

In addition to these telco-specific features, a telco mashup platform will typically be able to host services (third-party and its own components) in its mashup service container and ready mashups in a dedicated mashup repository (in either executable or interpretable format). For instance, the repository might cater to the voice call service used

in our scenario, whereas the shared whiteboard could be sourced from the Internet. The mashup life-cycle manager (part of the server-side runtime environment) must instantiate a mashup from the mashup repository on request, causing the instantiation of one or more client-side runtime environments that host the actual mashup UI. These runtime environments might be native mobile applications, regular Web applications, or JavaScript libraries running inside the Web browser.

Current Mashup Platform Analysis

To determine which of the aforementioned requirements are already supported by state-of-the-art mashup platforms, we analyzed Yahoo Pipes (http://pipes.yahoo.com), Intel Mash Maker (http://software.intel.com/en-us/articles/intel-mash-maker-mashups-for-the-masses/), JackBe Presto (www.jackbe.com), IBM Mashup Center (http://www-01.ibm.com/software/info/mashup-center/), WSO2 Mashup Server (http://wso2.com/products/mashup-server/), MyCocktail,[3] ServFace,[4] Karma,[5] Cruise,[6] MashArt,[7] Mashlight,[8] Opuce,[9] Spice,[10] and SOA4All.[11]

None of these platforms provides support for multiuser mashups. This shortcoming also impacts other important telco mashup requirements — that is, the ability to manage streaming media involving multiple users. Streaming media management in single-user mashup tools is relatively simple and supported by most available tools (such as embedded YouTube videos). However, the lack of support for multiuser mashups makes multiuser streaming management impossible. Regarding access to mashup instances via different channels — that is, the Internet and operator networks — only Opuce and Spice support bidirectional network integration (see the communications manager in Figure 1). Opuce uses a JAIN SLEE server to integrate telco protocols; however, the preferred solution so far in the majority of platforms we analyzed is to delegate all interactions with operator networks to dedicated converged services. The ability to interact with a system via different channels also enables new multimodal interaction. Instantiating a mashup only from a voice device running in an operator network isn't yet possible; non-Web clients can be included only in a running instance of a mashup by, for example, calling them from the mashup. Additionally, dealing with multiuser mashups requires that the platform manage bidirectional channel integration (for example, to broadcast an incoming MMS or voice call stream to multiple mashup participants), which no current platform supports. Only Opuce and Spice support integration with operator networks, but they offer very limited support for interaction paradigms other than classic hypermedia and thus aren't suitable for regular phones.

Another requirement we discussed is integrating device APIs. Most of the platforms we analyzed can generate Web-based mashup applications; some also let users create native device apps (such as Mashlight). So, although they could exploit standard interfaces (such as W3C or WAC interfaces) to access device APIs, most don't yet support such APIs.

Finally, another important requirement for telco mashups is managing QoS and billing. Only telco-specific tools partly address this aspect. Opuce adds annotations with pricing and QoS parameters to service descriptors, yet so far these annotations aren't used at runtime. Spice comes with a dedicated component for SLA management and billing (based on IMS and the 3rd Generation Partnership Project).

Challenges and Outlook

Our aim here was to approach the telco domain from an Internet perspective. We specifically looked at how Web mashups can integrate with telco network and device capabilities. Our analysis shows that a minimum level of telco support already exists in some mashup platforms, yet developers must still implement advanced telco features manually. We've identified several research challenges that seem crucial for telco mashups to be successful.

First, telco service providers must develop Web-ready streaming and signaling services that are easy to use and manage. For instance, setting up a video conference using the public Skype API still requires a programmer to master the Skype telephony protocol, which is complex and vendor-specific. Exposing such a complex API to a mashup environment is like not exposing it at all. Although some authors have proposed a framework based on state machines[12] or communication hyperlinks,[13] a shared telco service model doesn't yet exist.

Second, browser vendors must implement full support for device APIs. The W3C and WAC proposals for interfacing device capabilities are reasonable and easy to use. However, support for them even inside the latest browser versions is still weak and partly browser-specific, which hinders adoption.

Third, network operators and the Web community must agree on standard, cross-operator APIs for negotiating QoS and for billing, as well as respective monitoring and charging infrastructures. As of today, the market is fragmented, each operator adopts its own policies and technologies, QoS isn't adequately tracked, and each telco service requires its own payment logic.

Related to the previous point, the two communities must also develop cross-network user identification and authentication protocols to enable

seamless network integration. Suitable single sign-on and federation protocols seem of paramount importance.

Mashups are also required that can manage intermittent connectivity. Especially in the mobile Web, network disconnection is the rule, not the exception. Yet, we typically must still deal with services that can't work without the Internet, and we don't have robust solutions to handle connectivity problems at the application level.

Finally, we need to be able to design mashups that are adaptive — that is, that can autonomously fall back to lower-quality services if higher-quality ones aren't available, or that can switch to a different service if we cross a border and operate while roaming internationally. Telco services are typically country-specific, and using them while roaming could result in huge costs.

Luckily, some of these open challenges are already on the research agenda of academia and industry, and most network operators open APIs to the public. For instance, the GSM Association's OneAPI initiative (www.gsmworld.com/oneapi) aims to devise cross-operator, lightweight Web APIs with typical telco network capabilities. The Web-telco convergence so far moves mostly from traditional telco networks toward the Web, which means that the number and variety of telco services available on the Web is destined to grow significantly. This, on the other hand, requires the Web community to better understand, master, and suitably interface with the telco world.

Acknowledgments

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

References

1. J. Yu et al., "Understanding Mashup Development," *IEEE Internet Computing*, vol. 12, no. 5, 2008, pp. 44–52.
2. F. Daniel et al., "Toward Process Mashups: Key Ingredients and Open Research Challenges," *Proc. 3rd and 4th Int'l Workshop Web APIs and Services Mashups* (Mashups 10), ACM Press, 2010; http://doi.acm.org/10.1145/1944999.1945008.
3. C.A. Iglesias et al., "Combining Domain-Driven Design and Mashups for Service Development," *Service Eng.*, Springer, 2010, p. 71.
4. M. Feldmann et al., "Overview of an End-User-Enabled Model-Driven Development Approach for Interactive Applications Based on Annotated Services," *Proc. 4th Workshop Emerging Web Services Technology* (WEWST 09), ACM Press, 2009; http://doi.acm.org/10.1145/1645406.1645410.
5. R. Tuchinda, P. Szekely, and C.A. Knoblock, "Building Mashups by Example," *Proc. 13th Int'l Conf. Intelligent User Interfaces* (IUI 08), ACM Press, 2008; http://doi.acm.org/10.1145/1378773.1378792.
6. S. Pietschmann et al., "CRUISe: Composition of Rich User Interface Services," *Proc. 9th Int'l Conf. Web Eng.* (ICWE 09), LNCS 5648, Springer, 2009, pp. 473–476.
7. F. Daniel et al., "Hosted Universal Composition: Models, Languages, and Infrastructure in mashArt," *Proc. 28th Int'l Conf. Conceptual Modeling* (ER 09), Springer, 2009; http://dx.doi.org/10.1007/978-3-642-04840-1_32.
8. L. Baresi and S. Guinea, "Consumer Mashups with Mashlight," *Proc. ServiceWave* (ServiceWave 10), Springer, 2010, pp. 112–123.
9. J. Sienel et al., "OPUCE: A Telco-Driven Service Mashup Approach," *Bell Labs Tech. J.*, vol. 14, no. 1, 2009, pp. 203–218.
10. O. Droegehorn et al., "Professional and End-User-Driven Service Creation in the SPICE Platform," *Proc. 2008 Int'l Symp. World of Wireless, Mobile, and Multimedia Networks* (WoWMoM 08), IEEE Press, 2008, pp. 1–8; http://dx.doi.org/10.1109/WOWMOM.2008.4594818.
11. M. Zuccalà, "SOA4All in Action: Enabling a Web of Billions of Services," *Proc. ServiceWave* (ServiceWave 10), Springer, pp. 227–228.
12. R. Arlein et al., "Telco Meets the Web: Programming Shared-Experience Services," *Bell Labs Tech. J.*, vol. 14, no. 3, 2009, pp. 167–185.
13. V. Verdot, G. Burnside, and N. Bouché, "An Adaptable and Personalized Web Telecommunication Model," *Bell Labs Tech. J.*, vol. 16, no. 1, 2011, pp. 3–17.

Hendrik Gebhardt is a PhD student in the Distributed and Self-Organizing Systems Group at the Chemnitz University of Technology. Contact him at hendrik.gebhardt@informatik.tu-chemnitz.de.

Martin Gaedke is a full professor in the Department of Computer Science at Chemnitz University of Technology. Contact him at gaedke@informatik.tu-chemnitz.de; http://vsr.informatik.tu-chemnitz.de/people/gaedke.

Florian Daniel is a postdoctoral researcher at the University of Trento, Italy. Contact him at daniel@disi.unitn.it; www.floriandaniel.it.

Stefano Soi is a PhD candidate in information and communication technology at the University of Trento, Italy. Contact him at soi@disi.unitn.it.

Fabio Casati is technical lead for the research program on business process intelligence at Hewlett-Packard USA. Contact him at casati@disi.unitn.it.

Carlos A. Iglesias is an associate professor at the Universidad Politécnica de Madrid, Spain. Contact him at cif@gsi.dit.upm.es; www.gsi.dit.upm.es.

Scott Wilson is a senior researcher at the University of Bolton, UK. Contact him at scott.bradley.wilson@gmail.com.