Integration of Telco Services into Enterprise Mashup Applications

Olexiy Chudnovskyy, Frank Weinhold, Hendrik Gebhardt, and Martin Gaedke

Department of Computer Science, Chemnitz University of Technology
09111 Chemnitz, Germany
{olexiy.chudnovskyy,frank.weinhold,hendrik.gebhardt,
martin.gaedke}@informatik.tu-chemnitz.de

Abstract. In this paper we present our approach to integrate telco services into enterprise mashup applications. We show how cross-network integration and multi-user-oriented mashup concept support execution and orchestration of business processes. We identify the main classes of telco services and provide a reference architecture for telco-enabled mashup applications. Finally, we describe our approach for systematic integration process and give an outlook into our further research.

Keywords: Mashup, Telco Services, Enterprise, Integration.

1 Introduction

The availability and ubiquity of mobile devices is a matter of course nowadays. According to Gartner report of February 2011 more than 1.6 billion mobile devices were sold 2010, which is a 32% increase compared to 2009 [1]. Both operator networks and mobile devices provide sophisticated capabilities regarding voice, video and data transfer (so called telco services), which can be leveraged in business process integration and orchestration scenarios. However, the integration of these functionalities into Web applications is still challenging. We identified the following three problems: First, not all of the operator network services are exposed in ways easy to deal with for Web developers. Though the number of dedicated gateways and APIs grows with every year [2], their heterogeneity and fast evolution complicate the development of consumer applications. Second, without adequate models and tools the integration of telephony services is a time-consuming and error-prone task. And finally, the novelty of the emerging services and device capabilities requires a systematic approach and guidelines to support unskilled Web developers in the integration process.

We claim the adoption of Web mashup techniques will significantly decrease the effort to develop and maintain telco-enabled Web applications. Much work has already been done on the field of Web mashup. Many dedicated models, architectures and development tools exist [3]. All of them are characterized through the end-user oriented development paradigm and continuous reuse of already existing components and functionalities. New goal-oriented mashups can be constructed even without programming skills - leveraging the experience and building blocks produced by other developers [4].

Our goal is to extend traditional mashups towards telco-enabled ones, which would simplify the integration of telephony services and reveal new application possibilities of mashups within enterprise scenarios.

The rest of this paper is structured as follows. First, we illustrate an application possibility of telco-enriched mashups using an example scenario from the property management domain. Then we identify and describe challenges on the way towards integration of telco services into Web mashups. Afterwards, we present a reference architecture for telco mashups and the internals of a corresponding execution platform. Section 5 reviews which aspects have to be considered when developing telco mashups and how this can be done systematically. Finally, we conclude the paper by pointing out the current challenges in our research.

2 Example Scenario

In the following example, we show how telco mashups can support business scenarios dealing with orchestration and integration of business processes. In this example the availability of alternative communication channel, i.e. operator network and mobile devices, enable faster response and data transfer between involved parties. Especially human actors get better integrated into the decision-making processes, as decisions and required information can be provided from anywhere and to any time.

Pete's Apartments (PA) is a medium-sized apartment leasing company. PA takes care about billing and management issues, while flat maintenance is performed by partner firms. PA uses classical Web mashups for business intelligence tasks but also telco-enabled ones to coordinate different business processes and to communicate with its partners.

Lucy works as a customer advisor for PA. She uses a dedicated telco-mashup application to communicate with customers and to initiate various workflows regarding management, flat maintenance etc. One day Lucy gets a call from the renter Joe, who is having problems with his bathroom light (cf. Fig. 1). The mashup identifies Joe by his phone number and displays his customer information on Lucy's screen (1, 2). He reports the problem and Lucy captures it within a dedicated job-management-component (3). Based on the given information the map-component displays craftsmen from partner firms close to Joe's apartment. Lucy selects one or several craftsmen, who should be notified about the job details (4). She uses one of the messaging components (like instant messaging, voice-calls, SMS/MMS) to contact the craftsmen. Lucy sends an MMS to the selected craftsmen with Joe's address and problem description (5, 6).

The electrician Peter is one of the contacted craftsmen. He receives the message from Lucy while on the road and confirms the task via a SMS from his mobile phone. Usually, when in the office Peter replies by calling into the mashup application using his traditional office phone to get further information or coordinate next actions with

Lucy (7). The mashup confirms that Peter accepted the job and displays a notification message on Lucy's screen (8). Lucy accepts his confirmation and assigns him to this job (9).

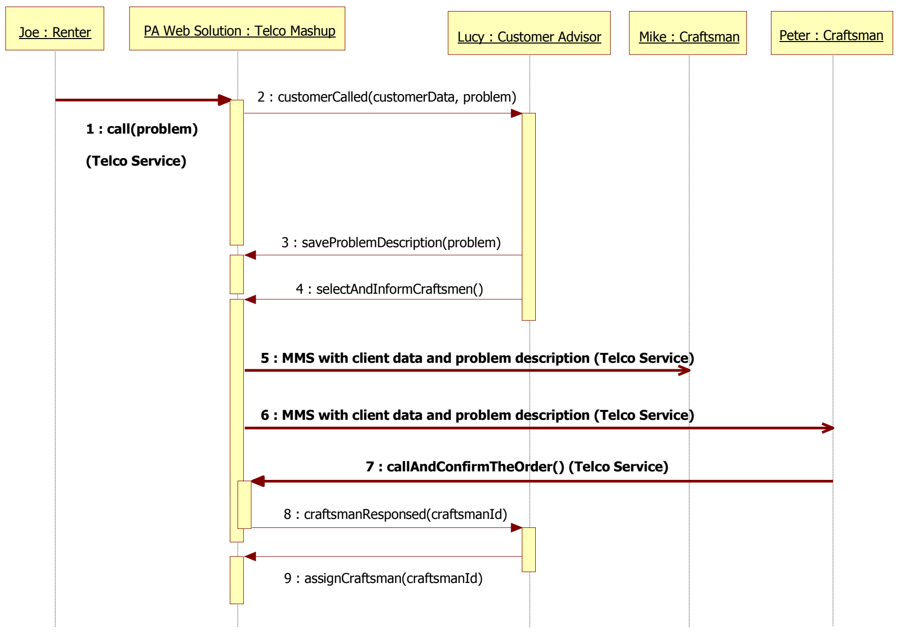
Joe : Renter
PA Web Solution : Telco Mashup
Lucy : Customer Advisor
Mike : Craftsman
Peter : Craftsman
1 : call(problem)
(Telco Service)
2 : customerCalled(customerData, problem)
3 : saveProblemDescription(problem)
4 : selectAndInformCraftsmen()
5 : MMS with client data and problem description (Telco Service)
6 : MMS with client data and problem description (Telco Service)
7 : callAndConfirmTheOrder() (Telco Service)
8 : craftsmanResponsed(craftsmanId)
9 : assignCraftsman(craftsmanId)

Fig. 1. Enterprise mashup application with integrated telco services

The example shows how two different processes (order registration and light repair) can be seamlessly integrated into a single workflow under different communication constraints (Internet, land and mobile phone lines). The integrating medium hereby is the dedicated telco mashup application, transcending technological communication constrains and thus enabling communication and data exchange between involved parties in a new way.

The presented scenario is a typical everyday scenario and limited in complexity – but its implementation is hindered through several problems we have identified during our research.

1. Telecommunication networks and mobile devices are not yet perceived as an efficient communication channel being able to perform process integration and coordination.
2. The missing models and frameworks hinder the integration of telephony services into Web- and software solutions.
3. Development of telco-enabled solutions is a time-consuming and error-prone task. Despite many existing tools and frameworks there is no rigorous and systematic

approach enabling costs-efficient development and evolution of telco-enabled Web applications.

To tackle the stated problems we derive the main research challenges discussed in the next sections:

- What types of telco services do exist and what are their key characteristics?
  It is important to classify and analyze different types of telco services. Services may operate in various networks and provide different data transfer capabilities (e.g. instant messaging, signaling, SMS/MMS etc.) Their characteristics have a crucial impact on scenarios and application possibilities for process orchestration and integration.
- How can telco services be combined with other functionalities and data sources?
  Web mashups has shown that development of new applications based on existing components and functionalities can be easy and even accessible for end users. It is necessary to devise a dedicated mashup model and architecture, which would support data transfer and process integration using telco services.
- How should a systematic approach for telco-enabled mashups development look like?
  It is necessary to devise a dedicated development process and a framework, which would reduce time and costs for the development of telco mashups. The average users should be supported in the process of creating their own mashups in a systematic and efficient way.

In the following sections, we focus on these research questions, analyze different kinds of telco services and introduce telco mashups with dedicated development process and framework.

3 Telco Services and Enterprise Applications

We define telco services as software services that provide communication and collaboration support. Depending on the network these services operate in, we distinguish between internet telco services, converged services and signaling services.

Internet telco services operate exclusively in the Internet, e.g. Voice over IP (VoIP) or instant messaging. The variety of available protocols and technologies enable these services to be used in complex data transfer and workflow execution scenarios between distributed systems. Internet telco services provide an efficient tool for asynchronous data transfer and synchronous voice/video communication. Furthermore, data transferred over services like instant messaging can be processed automatically by software and initiate further execution steps. Skype [5], Sipgate [6] or Google Voice [7] are some examples of the internet telco services providers.

Converged services mediate between different networks and communication protocols. A SMS message or VoIP calls from Internet to a mobile phone are examples of converged services. Converged services enable location-independent data exchange between parties, who have no access to the Internet but can communicate over other channels like operator networks. Especially processes and decision tasks, where people are involved, can benefit from capabilities of operator networks and pervasive availability of mobile devices. The data packets are usually limited in size and the mediation between networks is more expensive. However, small messages are often enough to confirm tasks or to provide required information. Monitoring and management of processes can be performed as well by notification using SMS or MMS. Tropo [8] and Twilio [9] are two wide-spread converged services providers.

Signaling services, which provide access to a network operator's signaling infrastructure. Examples of signaling services are notifications about incoming calls or negotiation of Quality of Service (QoS) parameters. Furthermore, signaling services can be used to establish a connection between two parties in order to initiate data transfer over alternative communication channel. Providers of signaling services are for example Developergarden [10], Comfone Signaling [11] or Orange API [12].

Finally, we define *device APIs* as services, which enable access to device capabilities such as cameras, microphone, location services etc. Device APIs provide additional data, which can be important or wishful for many enterprise scenarios. For example, location data from smartphones with GPS support can be utilized for decision making and task assignment process. As a result a better awareness of communication partners can be achieved. Furthermore device APIs enable mashup applications to be partially executed on the end devices and provide additional functions to the user.

Based on this classification, we derive a reference architecture, which enables Web mashups to integrate the presented telco services.

4 Integrating Telco Services into Mashups

Telco mashups represent an enhancement of classic Web mashups and leverage the capabilities of telco services. Within a mashup telco services are combined with other functionalities, which enable execution of both ad-hoc and complex cross-organizational workflows. We identified several layers of combination and aggregation possibilities regarding data, application logic and pieces of user interface:

- *Service Binding Layer* specifies data sources and services to be integrated into the mashup. Due to the variety of available standards and protocols (SOAP, REST, Atom, RSS etc.) the interface of services exposed to the upper layers should be unified and expressed within one semantically enriched description language. Policies, security considerations as well as quality of service aspects have to be defined at this point to enable cross-organizational data transfer and service invocation. Federation aspects of services should be systematically designed using dedicated modeling languages as presented in [13, 14].
- The *Data Mashup Layer* represents a step, where data coming from a number of heterogeneous sources are transformed, filtered and aggregated. The combination algorithm to be applied might be given either in form of a simple script snippet or using a dedicated mashup language, e.g. EMML [15] or DERI Pipes [16]. The underlying models may vary as well, e.g. the combination of data can be expressed

in form of pipes (the output of service A is connected with input of service B) or in terms of declarative instructions (data federation pattern). The data mashup enables integration of information coming from different organizations and departments in order to visualize workflows, execution states, relationships etc. In enterprise scenarios the aggregated data can be used to make decisions and initiate further execution steps or processes [17].

- The third layer, the *Widget Layer* specifies graphical interfaces and interaction with underlying data mashups or services. The resulting components, called widgets, can be based on various standards, e.g. W3C Widgets [18], Java Portlets [19], Google Gadgets [20] etc. Pre-defined packaging formats and well-defined interfaces to the run-time environment make widgets highly reusable and easily distributable. Widgets can be produced by different vendors and business partners, so that complete processes and workflows are implemented within one single component. To facilitate the reusability of widgets we propose to use a dedicated widget repository. The discovery of components should be enabled through an expressive semantic description language.
- The composition of widgets towards the final Web application is performed within the *Workspace Layer*. A workspace (or UI/UX-mashup) is a set of inter-connected widgets with additional services and configurations regarding inter-widget communication, layout, user interface presentation and user experience. The user of a mashup works with the workspace and consumes functions provided by the widgets. Widgets communicate with each other using a dedicated event bus and access general services implemented by the telco mashup execution platform. Incoming calls or messaging services are propagated by the platform to the workspace, so that each widget is notified about context changes or events. Inter-widget communication is a useful mechanism to transfer data between single business processes and o coordinate execution of single tasks [21].

The Telco Mashup Execution Platform represents the core component of telco-enabled mashups. The platform provides access to built-in telco services and supports the whole lifecycle of a mashup. Based on the presented architecture we derive requirements and identify main functions, which should be implemented by the platform in order to operate telco mashups (Fig. 2).

The platform should provide a bridge between the Internet and one or several operator networks. Telco mashups are executed within the platform, which is actually distributed on the *client side* (embedded into the Web browser) and *server side*. Server side provides access to embedded telco services and mashup management facilities. Upon request, new mashups are instantiated based on their configuration (stored in *mashup repository*) and *user profile settings* (security, billing and service level agreements). The execution of mashups is managed by the *life cycle manager* component, which guarantees, that charging and QoS settings, predefined availability as well as security and federation rules are respected. The *communication manager* hosted on the server side of the execution platform provides endpoints for mobile devices, manages incoming calls and routes them to corresponding mashup instances.

As such, the execution platform provides facilities to manage and operate telco-enabled mashups. Following, we analyze its application and provide guidance to take all presented aspects of the platform and telco mashups into account.

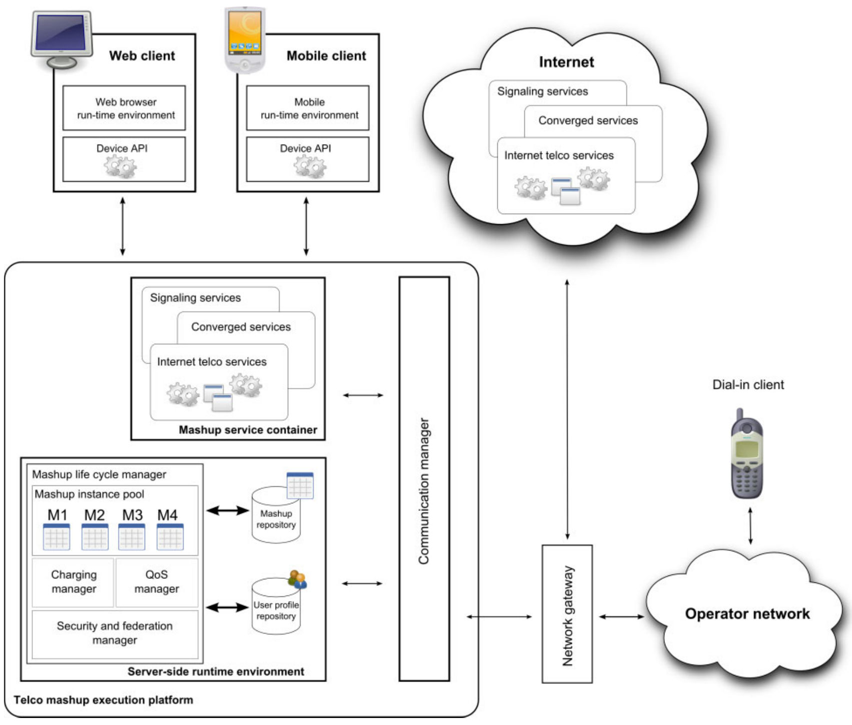

Fig. 2. Telco mashup execution platform

5 Development of Telco Mashups

The development of telco mashups differs from traditional Web applications in many aspects. First, mashups in general are based on the latest, easy-to-use Web technologies like REST, Atom, RSS etc. and serve typically a specific situational need [3]. Second, the development paradigm envisions that even end users are able to build their own mashups. Finally, the heterogeneity of mashup components, data sources and services requires a systematic evolution management and careful mashup design [22]. Following, we analyze these and telco-specific aspects, which should be considered while developing and maintaining telco mashups. We separate concerns and describe tasks to be performed in different phases of mashup lifecycle (Fig. 3).

The lifecycle of a new telco mashup application begins with its Conceptual Design, e.g. with the definition of essential mashup characteristics like title, description, category and purpose. Financial and governance rules, quality of service aspects and usage policies are specified within this stage. The definitions can be made both by end-users as well as skilled developers. The specified policies should be respected in the later design phases as well as during mashup execution. To support end-users in this process, the mashup development platform should provide discovery and recommendation facilities. Mashups built by other users can be re-used as a starting point or as a template for the newly created one.

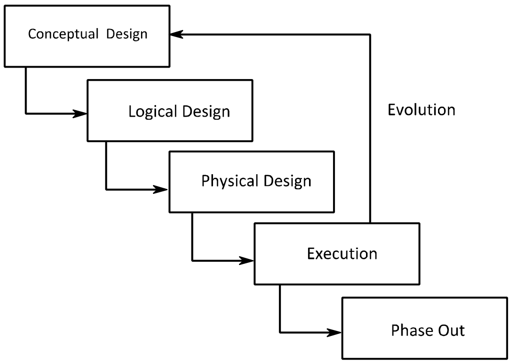

Fig. 3. Telco mashup lifecycle

Within the Logical Design stage one defines the abstract layout and basis components of the mashup. Developer (or end-user) assign a layout to the workspace and select components to be assembled. At this point the mashup is specified on a logical level only, i.e. using abstractions of components instead of concrete implementations. Pre-defined layouts as well as composition suggestions should be provided by the development platform to simplify these steps. The logical description of a mashup instance is an important artifact, which is used in later phases to suggest implementation possibilities or to exchange components at run-time, especially in telco-specific scenarios such as roaming. We suggest using RDF-based description languages and dedicated knowledge models to enable automatic composition and context adaption tasks [23].

The subsequent Physical Design phase can be completed either by skilled developer or automatically derived from the logical description. At this point, the system assigns concrete implementations of widgets, services and data sources to the logical representatives that have been composed as workspace. The mashup development platform should provide a repository with ready-to-use components and templates, which can be completed by mashup developers. If no component satisfies the goals, a dedicated widget editor is used to create new data mashups and wrap them using graphical interface. The look and feel of mashup is customized in compliance to corporate design and specific guidelines. Though the physical design will usually be done manually in the early beginning, the mashup development platform should provide automatic completion facilities as well. They can be used by unskilled developers, for prototyping purposes or to produce simple short-living mashup applications. The decisions made in the logical design phase, such as widget type or component requirements are used now to select concrete implementation and service bindings. For example, a map widget defined in logical design can be represented by either a Google Map or a Bing Map component. After the physical design phase executable description of mashup is available. Parts of the physical design like widget combinations or data mashup definitions are analyzed and stored by recommendation engine, which will suggest them in future if similar mashups are constructed.

The Execution phase is a step when a mashup instance is running and is used by one or several actors to perform their tasks. Telco mashups provide plenty of collaboration functionalities, which don't require the participants to act within one single network. To achieve this mashup platform implements basic telco services and gateways, takes care about network mediation, manages billing and QoS aspects of mashup applications. The application is running according to policies and governance rules defined in the conceptual phase. For example, the platform should guarantee that the maximal number of participants is respected or the operation time is not exceeded. The front-end of the mashup is rendered according to the physical design specification. Hereby the presentation may differ on desktop and mobile clients.

An important phase of each mashup instance is the Evolution stage. While the components and APIs used in the mashup evolve, obsolete widgets might be removed from the workspace or replaced by better ones, and new requirements might be met with the addition of new widgets. The dedicated repositories and recommendation engines simplify modification and extension of existing mashup instances and support their continuous evolution. Service bindings and operation rules can evolve as well, so that dynamic adaption facilities are needed to deal with the changing context. At this point, the logical definition of mashup helps to find alternative implementations of components and to suggest the best fitting ones. Governance rules from the conceptual phase define if and what components can be exchanged. For example, one can disable or restrict messaging functions of mashup while operating abroad in foreign operator networks. Mashup run-time can detect this context change and switch from Internet-based communication to SMS-based one.

Finally, the Phase Out is the last phase of a mashup instance, where the data produced during the execution is collected and archived according to the pre-defined rules and policies. Users cannot access the mashup anymore, but are able to retrieve operation statistics, log files, protocols or collect their own data etc. before the mashup is finally terminated. What information is important and how data should be dealt with after the mashup becomes unavailable is retrieved from the conceptual description of the mashup.

As we have shown, systematic development of telco-enhanced mashup applications and integration of telco services into mashup applications requires many additional considerations (and often dedicated supporting software) during the development process. The quality and effort needed to develop such kind of applications depend among others on the facilities provided by the development platform. We consider reusability as a key success factor for costs- and time-efficient development of mashup applications. Therefore, components like mashup repository and recommendations are integral parts of our proposed mashup platform and will gain more attention in future research and development.

6 Related Work

Much work has already been done in the field of mashups, both on the consumer mashups as well as on enterprise-oriented mashups side. The latter ones are especially related to our work as they enable integration of heterogeneous sources in different dimensions (data, services and UI/UX components) and take governance, management and security aspects into account. Following, we present and analyze some of the recent developments and show their relation to our approach.

IBM Mashup Center [24] is a popular enterprise mashup solution, which targets enterprise users with different needs and skills. The produced mashups enable integration of data, services and widgets from various (also legacy) sources. Similar to our model and architecture, mashups produced by IBM Mashup Center are assembled on both data and UI levels. Similar to our approach, a repository with mashup templates is available, which significantly simplifies the development of new applications. Though IBM Mashup Center provides much support in the mashup design, the telco-related aspects and invocation of local services like device APIs are not covered.

Another representative of mashup development platforms is JackBe Presto [25]. Its goal is to facilitate implementation of management dashboards, enterprise mashups and business intelligence applications. Same as IBM Mashup Center, the JackBe Presto platform provides a graphical editor for data mashups and visualizes them using widget-like objects called Apps. Though JackBe Presto provides a powerful platform to develop enterprise mashup applications, the integration of telco services remains challenging. Incoming voice calls and messages should be handled manually. Collaborative functions and life cycle management is also not considered within this approach.

In academia, the models and architectures of enterprise mashup applications have been thoroughly explored, e.g. in [26], [27], or [28]. Similar to our proposal, the proposed mashup models usually consist of several aggregation layers. The aggregation is performed both on data and UI - this approach covers many of the enterprise use cases and meets different needs of the end-users. Though many approaches exist, none of them addresses the telco aspects of enterprise mashups.

There are some few initiatives in European projects which research on the field of telco service and Web 2.0 integration. For example, OPUCE [29] focuses on building an infrastructure to facilitate the development and orchestration of Web services. The platform supports mashup adaptability and context awareness regarding users, operator networks and devices. Furthermore, it integrates various telco services like in- or outgoing calls, messaging services etc. However, billing and QoS management aspects are not addressed by resulting mashups. OPUCE produces single-user-applications and not multi-user-enabled ones as in our approach.

SPICE [30] is another European project, which targets particularly telco domain. The editor produced in the project enables semantic annotation of services to take non-functional telco-related aspects into account. In- and outgoing calls are supported through a media gateway (Asterisk PBX [31]) and enable also dial-in clients to communicate with mashup application. Also charging and management function are addressed through communication with other platform components over FTP or Ro interface. As with OPUCE, collaboration of several users using different devices is not addressed within SPICE mashups.

The presented approaches deal well with enterprise mashups when it comes to integration of sources available (or made available) over the Web. As we have seen above, integration of telco services is rather challenging and thus requires dedicated models, architectures and composition approaches.

7 Conclusions and Outlook

In this paper we have presented our 'work in progress' on the field of telco mashups. We analyzed how business scenarios benefit from the availability of several communication channels (i.e. Internet and operator network) and demonstrated it using an example scenario from property management domain. We proposed a dedicated telco mashup reference architecture and execution platform. To provide guidance in the development process, we analyzed their lifecycle and gave recommendations to each operation stage. Requirements made on the development platform will serve as basis for our future research. Currently, we are working on the specification of dedicated mashup and workspace description languages, which should cover all the aspects of presented lifecycle. Furthermore, we are going to develop first prototypes of execution and development platforms and apply them to implement the example above.

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

References

1. Market Share Analysis: Mobile Devices, Worldwide, 4Q10 and 2010 (April 22, 2011), `http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1542114`
2. ProgrammableWeb - Mashups, APIs, and the Web as Platform (June 09, 2011), `http://www.programmableweb.com/`
3. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing 12, 44–52 (2008)
4. Roy Chowdhury, S., Rodríguez, C., Daniel, F., Casati, F.: Wisdom-Aware Computing: On the Interactive Recommendation of Composition Knowledge. In: Maximilien, E.M., Rossi, G., Yuan, S.-T., Ludwig, H., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6568, pp. 144–155. Springer, Heidelberg (2011)
5. Skype (April 25, 2011), `http://www.skype.com/intl/en/home`
6. Sipgate (April 25, 2011), `http://www.sipgate.de/basic`
7. Google: Google Voice (April 25, 2011), `https://www.google.com/voice`
8. Tropo - Cloud API for Voice, SMS, and Instant Messaging Services (April 25, 2011), `https://www.tropo.com/home.jsp`
9. Twilio (April 25, 2011), `http://www.twilio.com/`
10. Developergarden (April 25, 2011), `http://www.developergarden.com/startseite`

11. Confome Signaling (April 25, 2011), http://www.comfone.com/index.php/services/signalling
12. Orange API (April 25, 2011), http://www.api.orange.com/
13. Meinecke, J., Gaedke, M.: Modeling Federations of Web Applications with WAM. IEEE (2005)
14. Heil, A., Gaedke, M., Meinecke, J.: Identifying Security Aspects in Web-Based Federations. IEEE (2008)
15. Viswanathan, A.: Mashups and the Enterprise Mashup Markup Language (EMML) (October 18, 2010), http://www.drdobbs.com/article/printableArticle.jhtml?articleId=224300049&dept_url=/java/
16. Phuoc, D.L., Polleres, A., Tummarello, G., Morbidoni, C.: DERI Pipes: visual tool for wiring Web data sources (2008)
17. Truong, H.-l., Dustdar, S.: Integrating Data for Business Process Management. IEEE Data Eng. Bull. 32, 48–53 (2009)
18. Widget Packaging and Configuration (June 09, 2011), http://www.w3.org/TR/widgets/
19. Sun Microsystems: Introduction to JSR 168—The Java Portlet Specification (June 09, 2011), http://developers.sun.com/portalserver/reference/techart/jsr168/
20. Gadgets Specification - Gadgets API - Google Code (June 09, 2011), http://code.google.com/intl/de-DE/apis/gadgets/docs/spec.html
21. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From people to services to UI: distributed orchestration of user interfaces, pp. 310–326 (2010)
22. Cappiello, C., Daniel, F., Matera, M., Pautasso, C.: Information Quality in Mashups. IEEE Internet Computing 14, 14–22 (2010)
23. Fortier, A., Rossi, G., Gordillo, S.E., Challiol, C.: Dealing with variability in context-aware mobile software. Journal of Systems and Software 83, 915–936 (2010)
24. IBM: IBM Mashup Center (2011), http://www-01.ibm.com/software/info/mashup-center/
25. JackBe: Presto (April 24, 2011), http://www.jackbe.com/
26. López, J., Bellas, F., Pan, A., Montoto, P.: A Component-Based Approach for Engineering Enterprise Mashups. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE 2009. LNCS, vol. 5648, pp. 30–44. Springer, Heidelberg (2009)
27. Yu, J., Benatallah, B., Saint-Paul, R., Casati, F., Daniel, F., Matera, M.: A framework for rapid integration of presentation components. In: Proceedings of the 16th International Conference on World Wide Web - WWW 2007, p. 923 (2007)
28. Gurram, R., Mo, B., Gueldemeister, R.: A Web Based Mashup Platform for Enterprise 2.0. In: Hartmann, S., Zhou, X., Kirchberg, M. (eds.) WISE 2008. LNCS, vol. 5176, pp. 144–151. Springer, Heidelberg (2008)
29. Sienel, J., Martín, A.L., Zorita, C.B., Martínez, B.C.: OPUCE: A Telco-Driven Service Mash-Up Approach. Bell Labs Technical Journal 14, 203–218 (2009)
30. Droegehorn, O., Konig, I., Le-Jeune, G., Cupillard, J., Belaunde, M., Kovacs, E.: Professional and end-user-driven service creation in the SPICE platform. In: 2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1–8. IEEE (2008)
31. Asterisk- The Open Source Telephony Projects | Asterisk (April 24, 2011), http://www.asterisk.org/