Integration of UI Services into SOA Based BPM Applications

Jörg Hohwiller and Diethelm Schlegel

Capgemini, CSD Research, Berliner Str. 76, 63065 Offenbach, Germany
{joerg.hohwiller,diethelm.schlegel}@capgemini.com
http://www.de.capgemini.com

Abstract. A service oriented architecture (SOA) combined with business process management (BPM) puts enterprises in a position to build applications in a very flexible and agile way. The SOA makes services available and BPM suites (BPMS) combine these services with user tasks to complex business processes. Integration of data and application services is supported by common technical standards and a wide range of products that are applicable out of the box. In contrast, the number of standardized user interface (UI) components that can be composed in SOA based BPM applications in a straightforward way is quite low. This is because the demands on such components are much higher. They must be embeddable into graphical front ends and interact with end users as well as the underlying business models. Unfortunately, there is no generally accepted approach and BPM tools provide their own proprietary mechanisms. The main drawback is the lack of re-usability resulting in higher cost and longer development phases. This paper derives basic requirements for UI components in order to make them generally applicable for SOA based BPM applications. It defines UI services meeting these demands in a platform-independent manner and presents one possible implementation based on JSR-286 Portlets and WSRP 2.0. This solution allows the development and embedding of system-independent and re-usable UI services by slightly extending existing BPM suites and enterprise portals.

Keywords: BPM, BPMS, BPMN, SOA, UI-Service, UI-Integration, Portal, Portlet.

1 Introduction

Market conditions are changing increasingly fast. This leads to growing demands on enterprise application landscapes. To be able to react faster on new requirements, function oriented systems are replaced by process related approaches resulting in a move from isolated legacy applications to service oriented architectures (SOA). In doing so, monolithic systems are divided into services ([1]) and grouped into components and domains.

It should be noted that according to [2] the definition of service landscapes is done independent of the underlying technologies. Nevertheless, the following

W. Abramowicz, L. Maciaszek, K. Węcel (Eds.): BIS 2011 Workshops, LNBIP 97, pp. 53–64, 2011.

realization will be based on some available standards and products like web services (WS), enterprise service bus (ESB), or enterprise application integration (EAI) frameworks.

Enterprises introduce business process management (BPM) in order to leverage high flexibility and agility. The use of BPM suites (BPMS) allows the integration of services and user tasks to business applications. Thereby, business processes are modelled and simulated in the BPMS. They can be executed as long as the service orchestration is based on the same technology stacks as used by the SOA realizations.

Data and application services are very well supported by a large number of products following several standards like REST ([3]), SOAP ([4]), WSDL ([5]), UDDI ([6]), enterprise service bus (ESB), enterprise application integration (EAI) and so on. This is reflected in a growing range of business related web services. They are even provided in the cloud, e.g. Amazon Web Services (`http://aws.amazon.com`) and Salesforce (`http://www.salesforce.com`).

In contrast, the integration of user interfaces (UI) in the area of SOA and BPM is more difficult by far. [7] describes the main challenges concerning user interfaces based on SOA and BPM including the heterogeneity of existing application landscapes as well as the absence of approved design patterns. [7] proposes the use of so-called dialogue services but gives no explicit definition. So, there is hardly any standardized solution based on them.

As a result, user interaction is often exclusively realized by the BPM suites themselves. The most common technique is the design of proprietary forms on the basis of business objects. Only a few products support more far-reaching standards like portlets or gadgets. Nevertheless, business process integration as well as the construction of comprehensive user interfaces still highly depend on the selected BPM product.

Graphical interface components can rarely be re-used and must be individually implemented leading to longer development time and higher cost. Based on a survey, [8] expects that the development of user interfaces takes about 50 percent of time. Later on, their maintenance still requires about 40 percent.

This just goes to show the importance of standardized and re-usable UI services that can easily be integrated into BPM suites. They are the prerequisite for product vendors and cloud providers to offer a growing number of user interface components in addition to data and application services. The availability of such UI components, in turn, will accelerate the implementation of user interfaces. To achieve this vision, UI services should satisfy the following conditions:

C1 In the SOA, UI services are accessible by common standards (e.g. SOAP, REST, WSDL and UDDI).
C2 In the BPMS, UI services can easily be integrated in the same way as data or application services.
C3 UI services can be integrated into enterprise portals as well as individual dialogues in a simple and straightforward way.
C4 There is an synchronization mechanism between UI services and business processes (UI⇒Process and Process⇒UI).

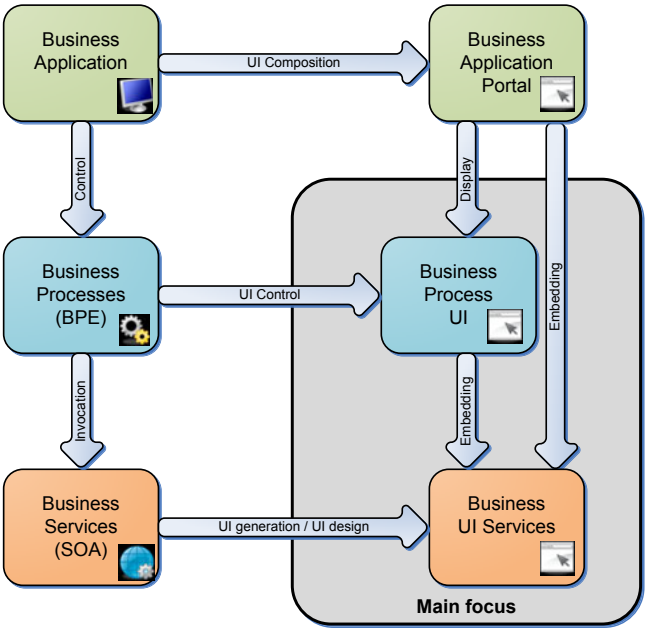

Fig. 1. UI Service Integration

The reason for these conditions with respect to the combination of UI services and SOA based BPM applications is illustrated in figure 1. The left column corresponds to components that are well supported by data and application services.

At the lowest level, the SOA will be supplemented by UI services according to condition C1. Mostly, they relate to existing services and are derived by UI generation or by manual UI design.

Building on this, the business process models are executed by the business process engine (BPE). The BPE invokes the services and controls the UI flow according to the executed process. As demanded by C4, this includes opening and closing modal dialogues as well as event handling and more sophisticated changes in existing user interfaces. In accordance with C2, the business process UI may embed one or more UI services.

At the highest level, there is the business application with its enterprise portal that is implemented by UI composition. It is possible to include UI services and to display business process UI if needed (condition C3). Thereby, the application controls the BPE, e.g. by starting processes or by accessing the currently running tasks.

The grey box shows the main focus of this paper concerning UI service and their integration. The following section 2 presents existing approaches in the area of service orientation and user interfaces. Thereafter, UI services are defined in a

technology-independent way in section 3 followed by a possible implementation based on portlets (section 4). At the end, there is a conclusion and an outlook (section 5).

2 Related Work

A lot of research has been done in the area of graphical user interface development. Thereby, attempts have been made to reduce the complexity by applying component based methods and, at the same time, increase the re-usability of existing UI components. For example, JSF ([9]) represents a step in this direction.

There are some other proposals for **component based UI development**. WebRatio uses a domain specific language called WebML ([10]) to integrate web services whereas VisualWADE is based on an object-oriented notation ([11]). [12] realizes UI composition via annotations and semantic event exchange but has no service oriented background. The CRUISe integration system ([13]) is a SOA approach. It defines UI services using a description language. These UI services are made available by a registry and are dynamically combined to UI mash-ups.

In the area of **web mash-ups**, integration is done in two ways. At first, the connection is established on the data layer. In most cases, this is achieved by distributing events though pipes like RSS feeds ([14], [15]). [16] additionally defines an universal component model that contains all available services (data, application, and UI). Other proposals use the data federation pattern ([17]). Additionally, UI integration is done by combining several web pages to a composite user interface ([18], [15]).

The ServFace Builder uses **UI generation** for the design of web applications ([19]). It automatically produces UI components on the basis of annotated web services. The runtime engine realizes the interaction between individual components. A similar solution is described in [20]. Hereby, the UI component orchestration and behaviour is specified by the business process execution language (BPEL, see [21]).

The so-called **distributed user interface orchestration** extends BPEL for defining user interface specific details ([22] and [23]). The UI components are constructed by compiling the enriched BPEL models.

In 2003, **portlets** were specified by extending servlets ([24]). They produce mark-up code parts that can be inserted into complete HTML pages. Integration is accomplished by so-called portal servers. [25] compares selected frameworks. Unfortunately, there were many restrictions such as proper communication between portlets and integration of JavaScript/AJAX. This prevented a wide-spread application of this standard. In 2008, version 2 of the specification ([26]) was released addressing these shortcomings by adding events, proper AJAX integration and an overall UI component model.

A main disadvantage of portlets is the restriction to Java based platforms. **Web services for remote portlets** (WSRP) ([27] and [28]) make portlets suitable for non-Java environments. As WSRP is based on WSDL it can easily be integrated into service oriented landscapes. While the first version of WSRP

includes only basic functionality, the second one adds important features like event handling.

All the proposals listed above have their focus on building more or less self-contained web applications. They do not offer generic interfaces for embedding UI services into BPMS or enterprise portals in such a way that the UI can be controlled by a business process engine (BPE). All the same, several of them can be used as the technological foundation for the integration of UI services into SOA based BPM applications. This will be shown for portlets (JSR-286 or WSRP) in section 4.

3 UI Services

Section 1 listed four conditions UI services should satisfy when they are integrated into BPMS or enterprise portals. The objective of these requirements is to make sure that the resulting interface provides a seamless user experience.

In this section, UI services are defined in a platform-independent and technology-neutral way. Based on this, compliance with the conditions is deduced. UI services are characterized by their interface that must support a number of functions needed for embedding and controlling of the UI service component.

Definition 1. *A* UI service *denotes a service in the sense of a SOA (see [1]) that supports the following interface functionality:*

- ***Life cycle (LC)*** *it is possible to handle the complete life cycle of UI services with respect to its presentation (e.g. created, closed, normal, modal, minimized, maximized, hidden).*
- ***Rendering (RE)*** *the UI service is able to render itself depending on its life cycle status (for example by producing mark-up code to be integrated in portal web pages). This method highly depends on the technology stack.*
- ***Event handling (EV)*** *events can be sent as well as received by the UI-service in order to communicate with other components (for example, when the user clicks on buttons or changes the selection). Events usually contain additional information.*

It is noted that the precise design of the interface functionality depends on the selected technical basis. The actual realization is assumed to be compliant with the corresponding requirements.

It must be shown that UI services fulfil the four conditions (see section 1). Because they belong to the services of a SOA, the first two conditions C1 and C2 apply as soon as they are implemented using suitable technical foundations. The life cycle and rendering functionalities allow the integration into enterprise portals (condition C3). The synchronization required by condition C4 can be implemented based on the event handling mechanism.

Some portal frameworks and few BPM suites use technical standards that already provide the relevant features with regard to life cycle, rendering, and

event handling. Others support additional features like access to shared variables (states) that will simplify UI service implementation in certain cases. However, it is not crucial because it can be realized by events. For example, reading shared variables can be accomplished by sending READ-events that are answered by a VALUE-events. Writing variables works in a similar way.

3.1 Architecture Overview

Figure 2 shows an overview of an architecture that embeds UI services into a SOA based BPM application landscape. At top level, the user interacts with the enterprise portal that integrates BPMS control and one or more individual UI services. The BPMS provides services for process management and task handling as well as manually designed forms. The business process engine (BPE) in turn calls services that are available in the SOA. This includes UI services because they meet the four conditions so that they can be accessed by common standards and integrated into BPM suites and enterprise portals similar to application services.

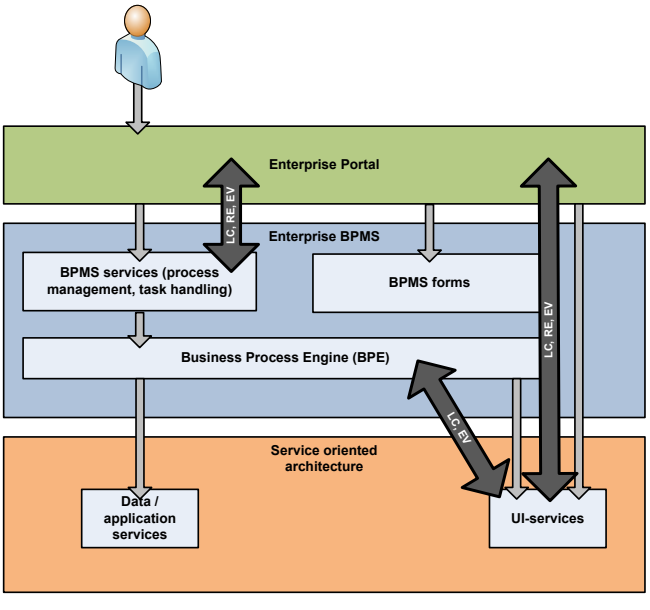

Fig. 2. Architecture Overview

Embedding of UI services into the enterprise portal needs the additional functionality described in the definition above. It relies on life cycle management (LC) to create UI components in the corresponding status as well as on rendering (RE) to display the component inside portal pages. Furthermore, the event handling (EV) must be supported by both UI services and enterprise portal. The portal framework is responsible for event distribution.

UI services synchronization with the BPE is realized by two mechanisms. First, it is possible to influence the UI behaviour by using life cycle (LC) functionality. For example, the business process will be able to open a modal dialogue or to bring certain UI components to the foreground. On the other hand, the BPE is able to synchronize with UI services using event handling (EV). A possible application is a process step that waits on user interaction. In this case, the UI service generates an event that is received by the BPE.

Many enterprise scope BPM suites already have integrated portals. As a consequence, there is no separated enterprise portal layer. The user interacts with the BPMS that controls the business processes and coordinates UI integration. With respect to the UI services, this makes no difference and the statements remain valid. The functionalities described above allow the seamless integration in this situation, too.

3.2 Synchronization

This section defines approaches for synchronizing business processes and UI services by using an appropriate process modelling notation. While many BPMS bring their own, an increasing number of tools support standardized notations. This paper uses BPMN 2.0 ([29]) because it is a very popular, well-known standard and, on the other hand, already contains elements needed for the integration of user tasks and services. It is even possible to express more technical aspects like escalations, timers, conditional branching, parallelism, and synchronization.

Figure 3 illustrates how business process models initiate actions in the user interface. The complete process model is represented by a pool that is divided into

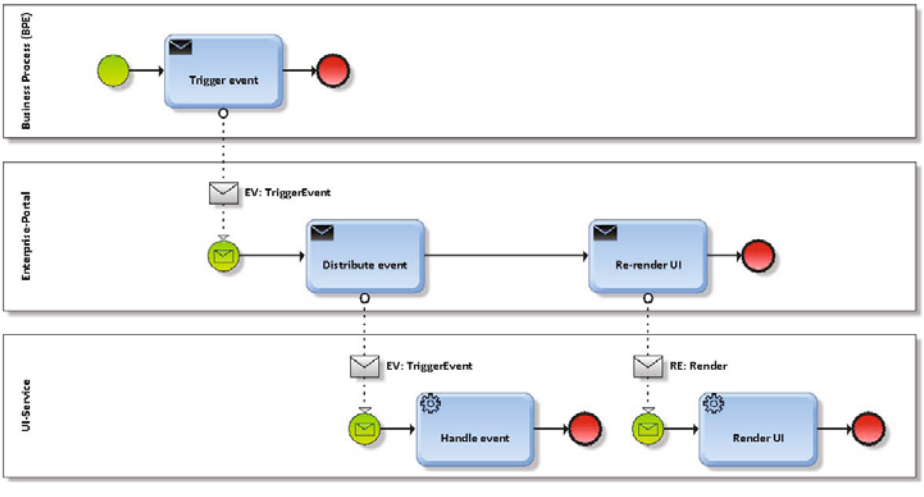

Fig. 3. Business Process Event

three lanes. The upper one corresponds to the actual business process whereas the two other have a technical background showing the operations within the enterprise portal and the UI service. Usually, the notation for sending the event message is sufficient from a business perspective and the technical lanes can be simplified or even completely omitted. Here, they are added for explaining the detailed sequence of interactions.

The business process triggers one or more UI services by sending an event to the enterprise portal. The enterprise portal distributes this event to the appropriate receivers and finally re-renders the user interface to let possible changes take effect.

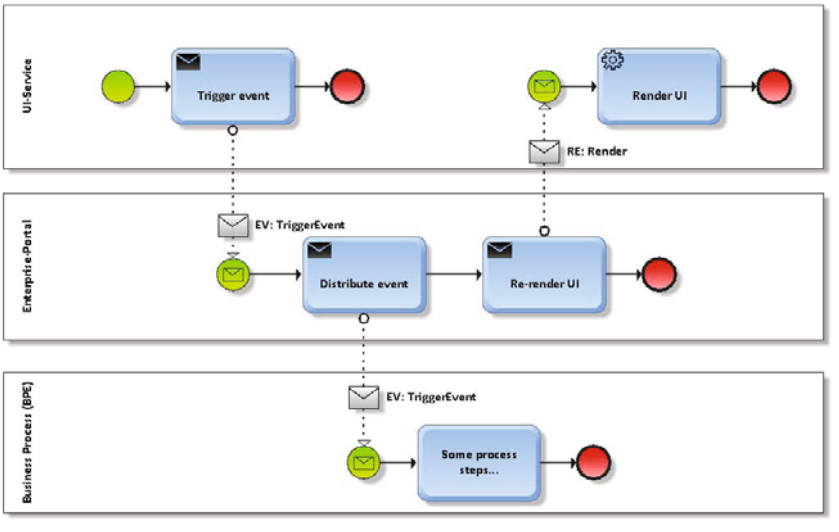

Fig. 4. UI-Service Triggers BPMS

Synchronization works in reverse as well (see figure 4). The UI service sends the event to the portal server that distributes it to other components including the BPE. So, the business process is able to receive it and proceed with any further steps. In this case too, the portal re-renders the user interface because the event could have influence on other UI services. Normally, the technical lanes may be omitted as proposed above.

Figure 5 illustrates the possibility to open UI services as modal dialogues. The business process sends a life cycle message to the appropriate UI service and waits until the close event is received. Finally, the UI service is cleaned up by using another life cycle message. Please note that the model is simplified for two reasons. First, the enterprise portal lane is omitted as suggested, and secondly, there is no error handling in the case of failed operations.

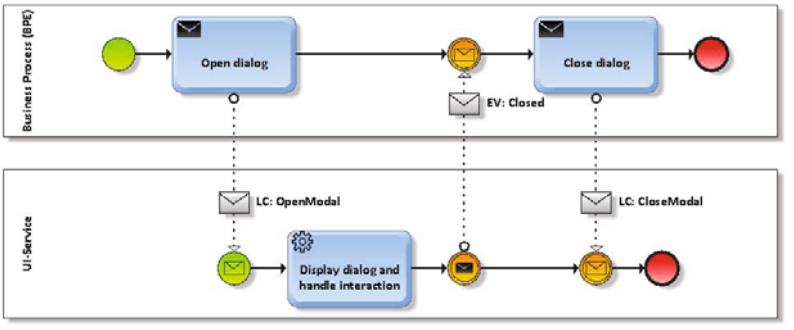

Fig. 5. BPMN Model for Modal Dialogues

4 UI Services with JSR-286 and WSRP 2.0

In section 3, the concept of UI services is introduced in a platform-independent manner. Thereby, some approaches that were pointed out in section 2 could serve as basis for a technical realization. This paper focusses on Java portlets in accordance with JSR-286 or WSRP 2.0. The reason for doing so is that portlets and WSRP are standardized and well-known mechanisms. They are supported by many products in the area of SOA and BPM. First of all, it must be examined whether Java portlets and WSRP correspond to the definition of UI services. This is achieved by inspecting their specifications:

- **Life cycle:** WSRP 2.0 provides life cycle functionality by defining operations of the following categories: service description, registration, and portlet management. They include initialization, cloning, configuration, and destroying of portlets. JSR-286 specifies several life cycle methods, especially *init()*, *processAction()*, and *destroy()*.
- **Rendering:** WSRP 2.0 implements rendering support by the operation *getMarkup* while JSR-286 contains the method *render()*. The return values are code fragments that can be included into portal web pages.
- **Event handling:** WSRP 2.0 defines event handling using two arrays (*publishedEvents* and *handledEvents*) together with the operation *handleEvents*. JSR-286 declares the relevant events in the *portlet.xml* deployment descriptor and supplies the *processEvent()* method. In both cases, new events are generated within life cycle and rendering functions.

Altogether, it is obvious that WSRP 2.0 and JSR-286 implement the interface functionality of UI services. This means that the four conditions for a seamless UI service integration are fulfilled. Moreover, they even offer further concepts like shared states, resource handling, and AJAX support.

Figure 6 gives an overview of the resulting SOA based BPM application. The basis consists of a landscape of web service and WSRP 2.0 or JSR-286

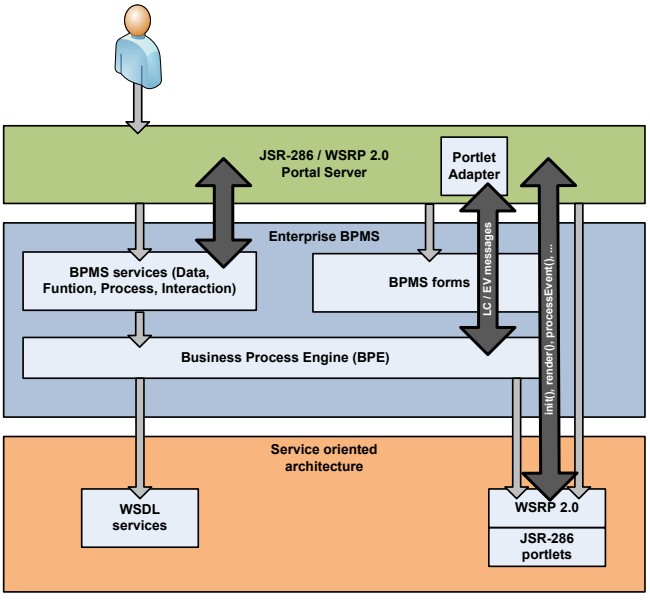

Fig. 6. Implementation Overview

portlets (SOA). The enterprise scope BPMS allows the business processes to invoke the relevant web services.

The portal server is based on WSRP 2.0 or JSR-286 and integrates BPMS control and SOA portlets. This assumes that the BPMS provides appropriate functionalities based on web service and portlets. The interaction between BPE and UI services is realized by a so-called *portlet adapter* that translates the portlet functionality for use in the BPE.

Most components are available as standard solutions. There are two important elements that may be absent: (1) the BPMS service interface and (2) the portlet adapter. In this case, they have to be developed as an extension.

5 Conclusion and Further Work

This paper has presented a solution for making UI components available as services that can be integrated into SOA based BPM applications. At first, some requirements concerning the integration of user interfaces have been explained. After providing an overview of existing approaches, UI services have been defined in a platform-independent way so that the demands are met. An architecture overview and synchronization methods between business processes and UI services have been derived. So, the suitability of implementations can easily be evaluated by verifying the compliance to the UI service definition.

Finally, the paper has proposed a possible realization on the basis of JSR-286 or WSRP 2.0 portlets. The benefit of this solution is that most of the components are already available in the form of standard products. Thereby, it has been shown that portlets are compliant to UI services.

As a next step, the theoretical results will be applied in practice. It is planned to implement sample systems using several BPM suites and portal servers. Besides that, more research is needed in the area of alternative technical platforms. Investigations must be carried out into how the concept of UI services can be realized by mobile platforms like smart phones or tablet computer.

References

1. Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, J.P., Voß, M., Willkomm, J.: A method for engineering a true service-oriented architecture. In: Cordeiro, J., Filipe, J. (eds.) ICEIS, vol. (3-2), pp. 272–281 (2008)
2. Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M., Richter, j.P., Voß, M., Willkomm, J.: Quasar Enterprise. dpunkt.verlag (2008)
3. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, University of California, Irvine, Irvine, California (2000)
4. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J.J., Nielsen, H.F., Karmarkar, A., Lafon, Y.: Soap version 1.2 (2007), `http://www.w3.org/TR/soap12`
5. Chinnici, R., Gudgin, M., Moreau, J.J., Schlimmer, J., Weerawarana, S.: Web services description language (wsdl) version 2.0 (2007), `http://www.w3.org/TR/wsdl20/`
6. Clement, L., Hately, A., von Riegen, C., Rogers, T., et al.: Oasis uddi specification (2005), `http://uddi.org/pubs/uddi-v3.0.2-20041019.pdf`
7. Nandico, O.F.: Integration für den benutzer: Einheitliche benutzungsoberfläche mit interaktionsservices. OBJEKTspektrum Ausgabe 02/2011 (Febuary 2011)
8. Myers, B.A., Rosson, M.B.: Survey on user interface programming. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 1992, pp. 195–202. ACM, New York (1992)
9. Burns, E., Kitain, R.: Javaserver faces technology, `http://java.sun.com/j2ee/javaserverfaces/`
10. Gómez, J., Bia, A., Parraga, A.: Tool Support for Model-Driven Development of Web Applications. In: Ngu, A.H.H., Kitsuregawa, M., Neuhold, E.J., Chung, J.-Y., Sheng, Q.Z. (eds.) WISE 2005. LNCS, vol. 3806, pp. 721–730. Springer, Heidelberg (2005)
11. Acerbis, R., Bongio, A., Brambilla, M., Butti, S., Ceri, S., Fraternali, P.: Web Applications Design and Development with Webml and Webratio 5.0. In: Paige, R.F., Meyer, B. (eds.) TOOLS (46). LNBIP, vol. 11, pp. 392–411. Springer, Heidelberg (2008)
12. Paulheim, H.: Seamlessly integrated, but loosely coupled: building user interfaces from heterogeneous components. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, ASE 2010, pp. 123–126. ACM, New York (2010)
13. Pietschmann, S., Voigt, M., Meiner, K.: Dynamic composition of service-oriented web user interfaces. In: Perry, M., Sasaki, H., Ehmann, M., Bellot, G.O., Dini, O. (eds.) ICIW, pp. 217–222. IEEE Computer Society (2009)

14. Yahoo! Inc., et al.: Yahoo pipes, http://pipes.yahoo.com/pipes/
15. JackBe Corporation, et al.: Jackbe presto, http://www.jackbe.com/products/
16. Daniel, F., Casati, F., Benatallah, B., Shan, M.C.: Hosted Universal Composition: Models, Languages and Infrastructure in Mashart. In: Laender, A.H.F., Castano, S., Dayal, U., Casati, F., de Oliveira, J.P.M. (eds.) ER 2009. LNCS, vol. 5829, pp. 428–443. Springer, Heidelberg (2009)
17. 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)
18. IBM Corporation, et al.: Ibm mashup center, http://www.ibm.com/software/products/de/de/mashupcenter
19. Dannecker, L., Feldmann, M., Nestler, T., Hbsch, G., Jugel, U., Muthmann, K.: Rapid Development of Composite Applications Using Annotated Web Services. In: Daniel, F., Facca, F.M. (eds.) ICWE 2010. LNCS, vol. 6385, pp. 1–12. Springer, Heidelberg (2010)
20. Menge, F.: Generation of user interfaces for service compositions. Master's thesis, Hasso Plattner Institut (2009)
21. Jordan, D., Evdemon, J., et al.: http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
22. Lee, J., Lin, Y.Y., Ma, S.P., Lee, S.J.: Bpel extensions to user-interactive service delivery. J. Inf. Sci. Eng. 25(5), 1427–1445 (2009)
23. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: From People to Services to UI: Distributed Orchestration of User Interfaces. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 310–326. Springer, Heidelberg (2010)
24. Abdelnur, A., Hepper, S.: Jsr 168: Portlet specification (2003), http://www.jcp.org/en/jsr/detail?id=168
25. Akram, A., Chohan, D., Dong Wang, X., Yang, X., Allan, R.: A service oriented architecture for portals using portlets. In: UK e-Science All Hands Meeting, Nottingham (2005)
26. Hepper, S.: Jsr 286: Portlet specification 2.0 (2008), http://www.jcp.org/en/jsr/detail?id=286
27. Kropp, A., Leue, C., Thompson, R., Braun, C., Broberg, J., Cassidy, M., Freedman, M., Jones, T.N., Schaeck, T., Tayar, G.: Web services for remote portlets specification (2003), http://www.oasis-open.org/committees/wsrp
28. Thompson, R., Allamaraju, S., Freedman, M., Jacob, R., Kramer, A.: Web services for remote portlets specification v2.0 (2008), http://docs.oasis-open.org/wsrp/v2/wsrp-2.0-spec.html
29. OMG, et al.: Business process model and notation (bpmn) version 2.0 (2011), http://www.omg.org/spec/BPMN/2.0