Architecture of and Migration to SOA's Presentation Layer

Stefan Link[1], Fabian Jakobs[1, 2], Ludwig Neer[2], Sebastian Abeck[1]

[1] Cooperation & Management, Universität Karlsruhe (TH), 76128 Karlsruhe
[2] CAS GmbH, 76131 Karlsruhe
{ link | jakobs | abeck }@cm-tm.uka.de, { fabian.jakobs | ludwig.neer }@cas.de

Abstract. Service-oriented architectures (SOA) enable enterprises to quickly respond to changing business requirements. Since there are human users interacting with SOA by, for example, entering data or making decisions, user interfaces are an integral part of the architecture. In this article the state-of-the-art concepts and relevant standards related to the presentation layer of service-oriented architectures are described. In a case study it is shown how to migrate an existing presentation component of a customer relationship management (CRM) system to SOA.

Keywords: Service-oriented Architectures, Service-oriented Approach, Web Services for Remote Portlets, Presentation Layer, Portal, Portlets, Migration

1 Introduction

The service-oriented paradigm allows the providing of abstract software functionality through services that can be flexibly composed to support business processes. A service-oriented architecture (SOA) enables the integration of existing applications by providing their functionality as services. Thereby the value of existing software assets is improved, redundancy in IT infrastructure can be avoided and a quick reaction to changes in business processes becomes possible [14]. To yield these benefits and to support the service-oriented paradigm, SOA can be divided into several logical layers (see Fig. 1.): The bottom layer of the SOA reference model is the application and data layer which comprises the existing (legacy) systems. Those systems can abstractly be seen as functional components. From the SOA perspective it does not matter if these legacy systems are internally monolithic or multi-tier architectures. SOA integrates and thereby leverages these systems by exposing their functionality as reusable services. At the process and integration layer of SOA the services provided by the application layer are mapped to the IT-supported parts of business processes. Assembling services in order to accomplish business logic and business processes is called orchestration. [7] defines orchestration as "the implementation of a business related workflow owned by a single entity through the combination of business relevant services". Processes themselves realized by orchestration exhibit a service interface [10].

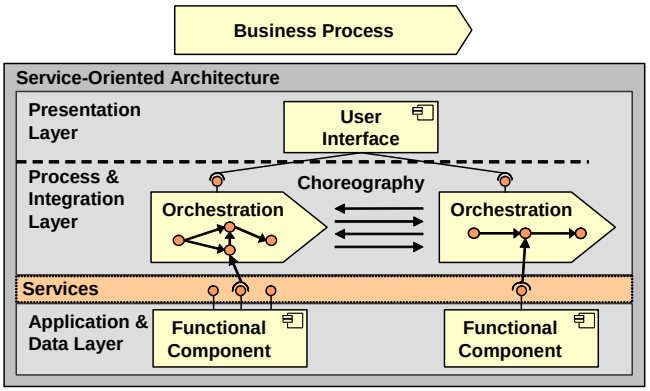

Fig. 1. Reference Model of SOA

These orchestration service interfaces can be invoked by the presentation layer in different ways. On the one hand they can be invoked directly by a user interface residing on the presentation layer, on the other hand it is possible to integrate a task management system between the process and integration layer and the presentation layer [13]. A task management system assigns different tasks to the responsible users or roles and hence allows for long running processes in which multiple users or organizational units are involved.

In this article we focus on the internal architecture of SOA's presentation layer and the interface to the end user and do not further investigate the connection to the process layer. The central questions dealt with by this article are:

- How to apply the service-oriented approach to the presentation layer?
- What are possible technologies and standards used on SOA's presentation layer?
- How to migrate existing user interfaces to SOA's presentation layer?

Based on these questions this article is organized as follows: First the service-oriented approach is introduced and applied to the presentation layer of SOA. Next a classification of presentation services is given. Portlets and portals as state-of-the-art standards for use on the presentation layer of SOA are focused upon in chapter 4 and chapter 5 introduces the Web Services for Remote Portlets (WSRP) standard as one of the most relevant standards found on the presentation layer of SOA. Chapter 6 presents our approach to migrate presentation components to SOA and chapter 7 outlines how we implemented the migration process by presenting a case study. As this article provides an overview of the state-of-the-art on the presentation layer of SOA we will point out the related work we consulted in chapter 8 and finally in chapter 9 postulate interesting research tasks we found.

2 Service-Oriented Approach for the Presentation Layer

The benefits of a service-oriented approach like loose coupling, interoperability and reusability are based on a basic interaction model involving three primary parties [3]: the service provider, the service consumer and the service registry. The interaction between these three parties is often referred to as the "find-bind-execute" paradigm. The service-oriented approach has been successfully applied to the application layer of SOA. Functionality which is implemented by e.g. a legacy application is provided through a service interface and thereby made accessible to service consumers as shown in Fig. 1. The mentioned benefits of the service-oriented approach are also appealing for the presentation layer because quick changing business processes require adjustable, interoperable and flexible user interfaces. Hence there is the need for a well-defined service interface on the presentation layer [6]. Furthermore the find-bind-execute paradigm has to be applied to the presentation layer, meaning the service provider, the service consumer and the service registry have to be identified for the presentation layer.
We distinguish presentation services [5] from business services as described further in the next section. On the presentation layer the find-bind-execute paradigm is implemented by the following three parties:

- **Presentation Service Provider:** Provides presentation components as services. As on the application layer there usually are several service providers offering presentation services specified to different purposes.
- **Presentation Service Consumer:** Invokes one or more presentation services in order to integrate them into a specific context like e.g. a control panel. There can be several presentation consumers on the presentation layer, each of which serves a certain business purpose.
- **Presentation Service Registry:** All presentation services made available by service providers are listed and can be found by service consumers. Service registries can be thought of as company-wide or even as world-wide.

To be able to identify these parties on the presentation layer, the reference model of SOA shown above has to be refined. Fig. 2 concentrates on the presentation layer showing a refined reference model where the mentioned service interface is presented providing the foundation for achieving interoperability and reusability on the presentation layer. The presentation container acts as a presentation service consumer and invokes several presentation services which are based on presentation components. So these presentation components are provided by one or more presentation service providers and are integrated to SOA through a service interface.
In order to find the adequate service, the presentation service consumer can search the presentation service registry. It offers a service interface as well through which services are discovered by the presentation service consumer. For this reason the service interface provided by the service registry is not presentation- but business-oriented. We denoted this service interface on the business service layer.

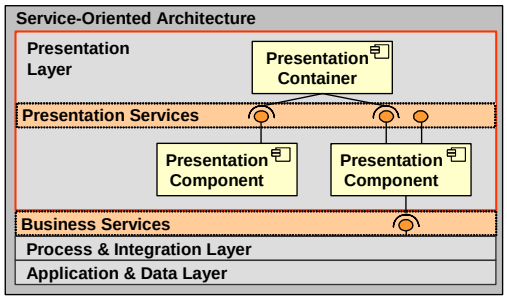

Fig. 2. Refined Model of SOA's Presentation Layer

3 Presentation Services

Figure 2 shows that there are different types of services integrated to SOA to serve different purposes. One has to distinguish between presentation and business services. As [5] points out presentation services provide a user interface and thereby allow for direct interaction between the human user and the service. This interaction is the key difference to traditional business services. Some sources therefore refer to these services as interaction services [4]. A business service focuses on processing data without any human interaction. This service follows the request-response model by receiving a request, processing it and generating a response on a programmatic level. It thereby generates a business value that is used in direct machine-to-machine communication.

A presentation service does not process any business-related data and thus generates no business value. It enables human users to interact with business services by providing e.g. markup fragments which can be aggregated by presentation containers like, for example, a portal. The presentation services act as a gateway between the business services and the user. Usually a business process is supported by more than one single business service and thereby may need more then one presentation service. A portal is one typical possibility to involve human users with business processes [10]. It contains portlets representing the user's interface to the business processes.

4 Portals and Portlets

A presentation service may provide for example markup fragments. The markup fragments have to be aggregated in a broader context like a portal [11]. From a technical perspective, a portal provides a container for aggregating content from various applications for presentation to the end user. The user does not recognize the different applications and s/he does not care how the content or functionality is provided. S/he wants to use one single interface which should be adjustable to her/his

needs. A portal is one possible solution as it usually is a Web-based application acting as gateway between human users and a range of different business services. The user can customize it in look and feel while it further supports the single sign-on approach for security [2]. From the user's perspective a portal is a customizable working space granting access to and integrating all applications needed with a single login.

Within a portal, several portlets are aggregated. They can each be seen as a user interface to an application and they are running inside a portal page along with any number of similar portlets. Portlets are defined as self-contained pluggable user interface components which are managed by a container (the portal) [1]. They process requests and generate dynamic content. For example if a user pushes a button or some other sort of event-triggering element of the user interface, the portlet processes this request and generates the adequate content to be displayed to the user.

A portlet can be implemented in very different ways. Some are standard-based (e.g. JSR 168) while others are proprietary to the portal which hosts them. Each of these portlets generates fragments of mark-ups which the portal aggregates to create a complete page that is presented to the user. Finally a portlet provides the functionality a presentation service is based on. If a portlet is developed following a portlet standard it can certainly be applied to any portal implementing this standard but yet this portlet would still be limited to one portal framework meaning that it does not support interoperability or loose coupling and hence does not comply with the service-oriented approach. Thus, there is a need for a platform- and framework-spanning portlet standard.

5 Web Services for Remote Portlets

Web services support the service-oriented approach in SOA enabling interoperability between different systems based on the use of open standards [15]. Therefore the Organization for the Advancement of Structured Information Standards (OASIS) adopted another Web service protocol for aggregating content and interactive Web applications from remote sources named Web Services for Remote Portlets (WSRP). With WSRP it is possible to integrate portlets from different (remote) service providers without concern to the implementation and provision of the portlet itself. There is one well-defined Web service interface where the portlet can be invoked. So if e.g. a company needs to implement a new business process with human interaction, it can first look for an adequate presentation service in a service registry.

If successful, there is no need to implement the user interface on one's own. In this way the service-oriented approach is applied to the presentation layer of the SOA. Before presenting the architecture suggested by WSRP, the main actors defined by the WSRP standard have to be introduced (compare with the find-bind-execute paradigm):

- **WSRP Producer:** Provides at least one portlet which can be customized by a consumer who integrates the portlet to her/his portal. The producer itself is implemented as a Web service. If the producer provides more than one portlet, it

often provides a portlet runtime (a so called portlet container) in order to manage the portlets.

- **WSRP Consumer:** Implemented as a Web service client like, for example, a portal that is able to invoke the Web services via WSRP. WSRP supports n:m relations between providers and consumers. Thereby it is for example possible for a consumer to support business process flows integrating several portlets from different producers. As the portlets are provided as services, the consumer provides proxies to access the needed portlets.
- **WSRP Portlet:** Is very similar to a standard portlet. The main difference is found in the access to the portlet. The WSRP portlet is provided through a WSRP producer and is accessed remotely through the interface defined by that producer. Unlike a portlet, a WSRP portlet cannot be accessed directly but instead must be accessed through its parent WSRP producer.

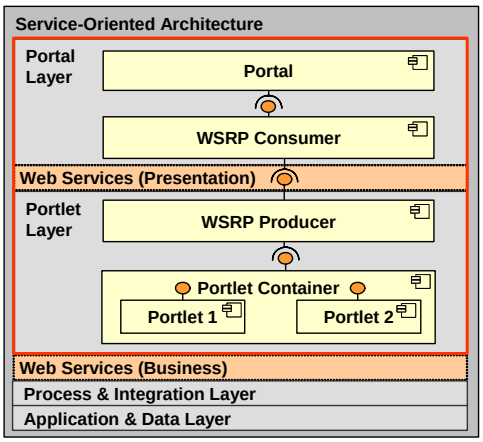

Fig. 3. WSRP Architecture on Presentation Layer of SOA

Fig. 3 extends the reference model of a SOA with the architectural specifications of WSRP. The WSRP standard defines a set of interfaces that all WSRP producers have to implement. The standardization of these interfaces allows WSRP-compliant portals to interact with remotely running portlets. There are two required and two optional interfaces forced by WSRP specification in order to enable a standardized communication between WSRP consumer and WSRP producer:

- **Service Description Interface (required):** Through the service description interface, a WSRP producer offers its portlets to the WSRP consumers. The WSRP consumer uses this interface to discover which portlets the producer offers. Additionally this interface is used if the consumer needs additional metadata about the producer itself. This metadata e.g. contains information if the producer requires registration or if a cookie has to be initialized before interacting with a consumer.
- **Markup Interface (required):** This interface enables a consumer to interact with a remotely running portlet. As the portlets are used as services the consumer uses

this interface to perform all actions necessary concerning the input of the human user. If for example a user submits a form (markup) from the portal page, the resulting response is created by delivering the appropriate markup. It might also be necessary for the portal to receive the latest markup of the portlet (user clicks e.g. a refresh button). The portlet then delivers the markup according to its current state.

- **Registration Interface (optional):** With this interface WSRP allows for an in-band mechanism for registration. This registration enables the provider to come up with a customized behavior and appearance for the portlet which differs from the standard appearance of the portlet. This mechanism further allows for a role-based interaction, e.g. each role is allowed to see different portlets. behavior and appearance.
- **Portlet Management Interface (optional):** A portlet hosted by a portlet container has a life cycle. It is instantiated, executed and finally destroyed. With the portlet management interface the WSRP consumer obtains access to the life cycle of the remotely-running portlet. The consumer then has the ability to customize a portlet's behavior or even destroy an instance of a remotely-running portlet.

Following the WSRP standard the service-oriented approach can be implemented on SOA's presentation layer. Further, WSRP provides the means to address another task concerning service-orientation. As SOA is widely accepted as a future software architecture, one has to consider possibilities of migrating existing legacy applications to all layers of SOA. The migration of presentation components is one task which has to be dealt with.

6 Migration to SOA's Presentation Layer

With the advancements of integration technologies it is possible to integrate existing applications in SOA, e.g. via Web services. As emphasized in the introduction, it is necessary for a company to be able to adjust to quickly changing demands of the customers. The establishment of SOA as a flexible and adjustable architecture to support such rapidly changing business processes is a strategic goal. Whereas SOA is the future architecture, existing systems have to be migrated towards SOA. The migration process from an existing IT application infrastructure to SOA is very work intensive so one has to think about how a migration process can be performed with minimal influence on the day-to-day business [6]. It is reasonable to strive for a smooth migration which is defined as a migration process where an application component remains applicable in both the old and the new software architecture at each point of time in the migration process. There are several advantages of such a smooth migration [9, 12]:

- Often substantial investments have been made to develop legacy applications and thereby it is economically not an option to just abandon them.
- During the migration process business has to go on. It is impossible for an enterprise to stop selling its products and services for several months just for the purpose of re-organizing its internal software systems.

- As the documentation of legacy software usually is poor, the software itself is the only place where business logic is "documented".

Another benefit of a smooth migration is that providers offer third party modules for these legacy applications. Each of them has to be taken into account if the migration is economically advantageous or not. Until the legacy application is shut down, their modules remain usable.

A smooth migration on the presentation layer has, according to the above definition, to provide presentation components applicable in both the old and the new software architecture. SOA, with the use of WSRP on the presentation layer, is our chosen architecture to migrate to because of its many advantages. As preconditions to the suggested migration concept, we assume that the architecture which is to be migrated is not a Web application but a traditional client-server application. So the main tasks concerning migration on the presentation layer are to first extend the legacy software architecture to be able to integrate and invoke portlets and then to provide the existing presentation components as portlets. This provision is done by an iterative process consisting of three basic steps:

1. Identify a reusable presentation component of the legacy client.
2. Duplicate the presentation component using portlet technology.
3. Integrate this portlet back to the legacy client.

The first step is a thorough analysis of the legacy client. Reusable presentation components must be identified and examined if these components can be used to build useful presentation-oriented Web services for SOA. Another important point is to check if the component can be technically implemented using portlet technology. Since portlets are based on standard Web technologies such as HTML, CSS or JavaScript, not everything that could be done in the legacy client running on a local system can be done in a portlet. Portlets, for example, cannot directly use services of the user's local operating system. This results in a sorted list of components suitable for the migration process. The order of this list is influenced by the estimated cost and the expected benefit of the migration of this component.

Next, the user interface component must be implemented as a portlet. Traditional presentation components of Web applications tend to have a very ineffective usability in terms of user feedback and response times [8]. To achieve acceptance by users of the legacy client, the new portlet should at least reach the same level of usability as the legacy client provided. Current trends in Web application development (e.g. AJAX [16]) try to solve these issues by transferring some of the user interface code as JavaScript into the client and communicate asynchronously between Web browser and server.

The portlet can now be used in any WSRP capable portal but reusing it in the legacy client needs some initial work. The client must be extended to display portlets along with the present user interface. This extension can be split into three components, which have to be added to the client:

- **Portal Controller:** Acts as a WSRP consumer and is responsible for establishing the communication with the WSRP provider. It further establishes the communication to the legacy presentation components via the client controller.

- **Portal Plugin:** This component is needed to display the aggregated markup fragments of the invoked portlets to the user.
- **Client Controller:** Permits the legacy client to communicate with the embedded portlets. With this extension in place the legacy client can benefit from all published portlets of the SOA.

Fig. 4 shows the migration scenario. The central integration point in this architecture is the portlet container. Each user interface component that can be implemented as a portlet can immediately be integrated into the portal and the modified legacy client. The advantage of this approach is based on the fact that after each step in the migration process a fully functional system is available. The legacy client's presentation components are migrated step by step.

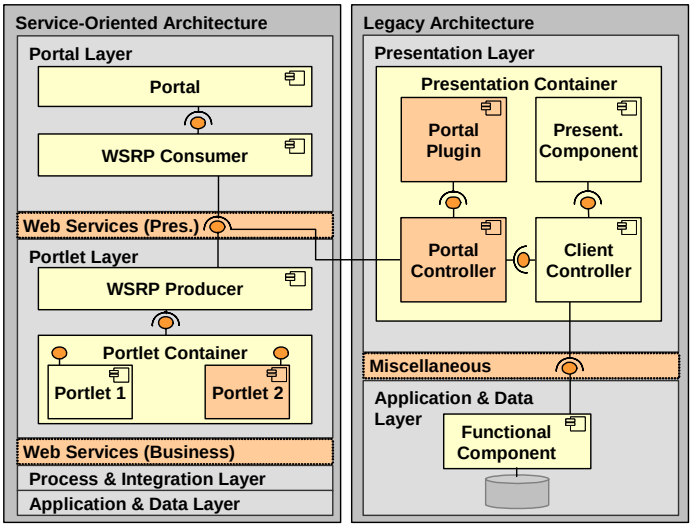

Fig. 4. Migration Scenario

7 Case Study: Migration of a CRM client

This exemplified migration process is now demonstrated on a client of a customer relationship management (CRM) system. This CRM system has been in use for several years. It is based on Delphi and has to be migrated according to the approach described in the last chapter.

The first step to take is to analyze the existing client in order to identify a reusable presentation component. The CRM system provides among other features a calendar component. This component is needed in the future client and therefore has to be migrated. As the calendar component will be implemented as a portlet, one has to

make sure that the usability of the calendar portlet is at least equal to the old one. We managed to achieve an excellent usability with the AJAX framework qooxdoo [17]. Qooxdoo provides most of the basic functions needed to implement a calendar as Web application like to drag and drop dates. While implementing the calendar portlet we encountered several problems we had to solve. Most of them were based on the fact that a portlet may run amongst several others within a portal and is not aware of its neighbor portlets:

- **Persistence of state:** If the user decides to reload the whole page the state of the portlets using AJAX is lost because this state is stored in JavaScript variables which are reset if the portlet is reloaded. In this case we used the portal's session context to keep the state of the AJAX portlets consistent.
- **Common namespace:** Portlets do not only share the space on a portal page, they also share a common namespace in a common JavaScript context. All global apparent classes, methods or variables can unintentional be modified by different portlets. This problem gets worse if one thinks about a portlet running in several instances. The solution comes with a unique portlet identifier with which every hyperlink, form etc. is extended. This way one can for example determine in which instance a hyperlink was clicked.
- **Asynchronous communication:** Within a portal every request sent from a portlet is relayed through the portal itself. Further a portal always answers a request from a portlet by sending the whole page anew so the benefit of asynchronous communication is lost. We solved this problem by implementing a servlet running parallel to a portlet and sharing its session data. Servlets are able to send requests unchanged to their client and therefore qualify as a solution for this problem.

Afterwards the existing client had to be extended to be able to use the new portlets. We extended the Delphi client by integrating a Web browser component and used it as runtime environment for the portlets. Further we used the API of the Web browser to manipulate the pages document object model (DOM) or to deploy our own JavaScript functions. Finally we followed the approach presented in [18] and were able to asynchronously call Web services with AJAX.

8 Related Work

The reference model of SOA (see Fig. 1) that was the starting point of this article can be found in different publications, which cover the presentation layer as follows:

- Erl [7] agrees with that view on SOA but does not address the presentation layer.
- Arsanjani [3] addresses the presentation layer as a "future layer" which has to be taken into account in order to implement solutions based on the increasing convergence of standards like WSRP.
- Caste [5] focuses attention on the presentation layer demonstrating the architectural design given by the WSRP standard.

The two most important publications related to the migration aspects of an SOA are:

- Lewis et al. [12] introduce the benefits of a SOA and point out, that migration to a SOA is neither easy nor automatic. They agree with [9] that for many organizations it is impossible to walk away from investments made with existing legacy applications and to redevelop the needed functionality as services from scratch. A migration process called "The Service-Oriented Migration and Reuse Technique" (SMART) is provided. SMART collects input information about the legacy system and after performing several activities it outputs a service migration strategy. The strategy is thereby residing on a high business level and finally recommends e.g. which components of the legacy application can be derived from services.
- Hasselbring and Reussner [9] start from the same initial situation: there is a legacy system which cannot be abolished and, therefore, has to be migrated to SOA. They present an architecture pattern called Dublo (Duplication of business logic). We based our solution on the Dublo approach whose basic idea is that business logic is formulated in a new business logic tier and a legacy adapter for access by the new business logic to the existing legacy business logic is written. This adapter is used to access the database so it is only accessed via the existing legacy code. A new presentation layer is added which manages new business logic implemented to the new business logic tier.

9 Conclusion and Outlook

The application of the service-oriented approach to SOA's presentation layer enables enterprises to easily adjust or redesign user interfaces to quickly changing business processes. Instead of redeveloping the user interface each time when new requirements have to be met, the user interface is rearranged through a new composition of presentation services. As shown the WSRP standard supports this service-oriented approach by giving the architectural means to provide presentation components as services. Yet, one still has to compose these presentation services manually. As on the application layer, business process modeling languages help to almost automatically derive an executable code from a business process model, making it preferable for developing user interfaces. Several constraints like the usability or the convenience of the user interfaces have to be taken into account. We understand that one possible approach could be to use an existing modeling language like BPMN and to add information concerning the user interface already at design time. In order to be able to add this information it must be investigated if the given elements of a modeling language are sufficient or if it becomes inevitable to extend the meta-model of the according modeling language.

The approach we presented enabled us to migrate existing presentation components to portlets and to use them within both the legacy and the new service-oriented architecture. Migration is and will become more and more important within the next years as the introduction of a service-oriented architecture within enterprises becomes a strategic goal. Considering the heterogeneous IT systems which are in use today it is unlikely that a common, general and still applicable migration process can be derived. Therefore it is all the more reasonable to develop individual migration strategies as a source of the basics for new migration projects.

References

1. Asif Akram, Rob Allan, Rob Crouchley: WSRP Reincarnation of Service Oriented Architecture, CCLRC Daresbury Laboratory, e-Science Centre of Excellence, University of Lancaster, UK http://www.grids.ac.uk/eSC/AllHands2005/AHMCD/papers/436.pdf
2. Asif Akram, Dharmesh Chohan, Xiao Dong Wang, Xiaobo Yang and Rob Allan: A Service Oriented Architecture for Portals Using Portlets CCLRC e-Science Centre, CCLRC Daresbury Laboratory Warrington WA4 4AD, UK http://www.grids.ac.uk/eSC/AllHands2005/AHMCD/papers/406.pdf
3. Ali Arsanjani: Service-Oriented Modeling and Architecture, IBM developer works, 2004. http://www-128.ibm.com/developerworks/webservices/library/ws-soa-design1/
4. Wolfgang Beinhauer, Thomas Schlegel: User Interfaces for Service Oriented Architectures, July 2005, http://www.webservice-kompass.de/fileadmin/publikationen/User_Interfaces_for_Service_Oriented_Architectures.pdf
5. Bryan Caste: Introduction to Web Services for Remote Portlets, IBM Developerworks 2005 http://www-128.ibm.com/developerworks/webservices/library/ws-wsrp/
6. Kishore Channabasavaiah and Kerrie Holley: Migrating to a service-oriented architecture http://www.cytetech.com/documents/SOA-IBM.pdf, White Paper IBM, April 2004
7. Thomas Erl: Service-Oriented Architecture: Concepts, Technology and Design, Prentice Hall PTR, ISBN 0-13-185858-0 August 04, 2005.
8. Jesse James Garret: A New Approach to Web Applications, Technical Essay, 2005 http://adaptivepath.com/publications/essays/archives/000385.php
9. Wilhelm Hasselbring, Ralf Reussner: The Dublo Architecture Pattern for Smooth Migration of Business Information Systems: An experience report, Proceedings of 26th International Conference on Software Engineering (ICSE 2004), IEEE Computer Society Press, Mai 2004 http://ieeexplore.ieee.org/iel5/9201/29176/01317434.pdf
10. Frank Leymann: Web Services - Distributed Applications without Limits, Business, Technology and Web, Leipzig, 2003.
11. Jeff Linwood, Dave Winter: Building Portals with the Java Portlet API, ISBN: 1-59059-284-0, 2004 http://chronos.org/alphawiki/attach?page=Portlets%2Fportlet.pdf
12. Grace Lewis, Ed Morris, Liam O'Brien, Dennis Smith, Lutz Wrage: SMART: The Service-Oriented Migration and Reuse Technique, September 2005, Technical Note http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn029.pdf
13. Frank Leymann, Dieter Roller, M.-T. Schmidt: Web Services and business process management. In: IBM Systems Journal (2002) 41, S. 198-211.
14. James McGovern, Sameer Tyagi, Michael Stevens, Sunil Mathew: Java Web Services Architecture, IBM Developer Book, Juli 2003 http://java.sun.com/developer/Books/j2ee/jwsa/
15. Arthur Ryman: Understanding Web Services, July 2003, IBM Technical Article http://www-128.ibm.com/developerworks/websphere/library/techarticles/0307_ryman/ryman.html
16. Jesse James Garret: A new approach to Web applications, essay from adaptive path, February 2005 http://adaptivepath.com/publications/essays/archives/000385.php
17. The qooxdoo AJAX framework: http://www.qooxdoo.org
18. Ahmet Sayar, Galip Aydin, Marlon Pierce, Geoffrey Fox: Integrating AJAX Approach into GIS Visualization Web Services, AICT-ICIW'06 http://ieeexplore.ieee.org/iel5/10670/33674/01602302.pdf?isnumber=&arnumber=1602302