2007

On the Reuse of SOA Components on Business Process Analysis Stages

Martin Juhrisch
Westfälische Wilhelms-Universität, juhrisch@uni-muenster.de

Jens Weller
Westfälische Wilhelms-Universität, jens.weller@tu-dresden.de

Follow this and additional works at: http://aisel.aisnet.org/pacis2007

Recommended Citation
http://aisel.aisnet.org/pacis2007/66
Abstract
In 2005 four German universities created a research program to improve and standardize its administration processes. Therefore, reams of processes have been analyzed and core functions identified. Automatable core functions have been afterwards implemented as web-services using the Service-oriented architecture (SOA) paradigm. For reasons of reuse these web-services have been documented using a service catalogue. However, a real advantage of SOA does not evolve until complex business services become configurable on business level. Thereby, the problem arises to make the meaning of the services understandable to non-IT employees, like business analysts.

In this paper, we introduce a framework for building up a service catalogue addressing business analysts instead of IT engineers. Therefore, we discuss the composition of web-services to more complex business components. These components are described on an IT-level addressing the catalogue manager, but contain also non-formal information for business analysts. Thus, the redesign of the universities processes can be supported now, using web-services encapsulated by business components. Our findings can help to increase the usage and acceptance of web-services on a business process level.

Keywords: Enterprise Architecture, Web-service, enterprise models, component catalogue

Introduction
Today’s German universities are faced with a modernization and performance gap. At the same time the requirements of students and researcher increase, the human resources necessary for the management of the university and its central institutions remain static or are even decreasing (Becker et al. 2006). A consortium of four German universities has started to meet the challenge by analyzing its core processes in order to find reorganization potential of its structure.

A large potential for modernizing the university administrations lies in process reorganization with information and communication technology (ICT). Within the scope of the IT reengineering project “Munster Information System for Research and Organization” (MIRO), we build upon the analysis outcomes and introduce new ICT-components. Additionally, we try to reengineer legacy information systems on the basis of a university wide service-oriented architecture (SOA) (Fremantle et al. 2002). In contrast to previous software architectures, SOA offers the possibility to establish a closer connection of business process flow – represented by process models – and information system infrastructure – represented by web-services (Erl 2005). The integration of SOA components to enterprise models follows this idea trying to close the gap between organizational and IT domain. Thus, we keep the IT infrastructure flexible to be able to adjust its functionality if necessary.

Directly configuring existing web-services within business processes, however, it is not feasible for several reasons. Firstly, most business analysts are not familiar with the technical understanding of a web-service. Thus, there is a need for a more business-like description of
web-services (Turowski 2002). Secondly, web-services cannot be composed in any way. Instead, web-services may require other services to be executed first (White 2005). Thus, a pre-composition of web-services to more business-oriented functions is necessary before using them within business process models.

At the University of Munster the semantic gap between IT and organizational domain is bridged by building up a service catalogue containing all university web-services including a description of their business task using so-called business function types. An instantiated business function type covers a certain part of the business process model that is implemented by a web-service. Assigned to a web-service they represent its outer business perspective. To solve the second problem, mentioned above, there was a need to extend the service catalogue. Thereby, the business function types are composed to more complex ICT components representing aggregated business functions. With the service catalogue e.g. departments or entirely other universities get a look-into the university service landscape and a tool to navigate to the service they enquire.

The aim of this paper is to present necessary enhancements of existing software component catalogues in order to enable a business model-driven configuration of SOA. We illustrate the usefulness of our approach with an extension of a UDDI (Universal Description and Discovery Interface) protocol based service catalogue at the University of Munster. We show consequences for the catalogue utilization and demonstrate how ICT business components can be reused within common enterprise models. Our findings can help to increase the usage of web-services on a business process level. Hence, the positive effects of SOA architectures, like situation dependent adaptation of software functionality to the actual documented utilization and business requirements can be realized (Weller et al. 2006).

The paper is structured as follows. In section 2, we explain the fundamentals of Service Oriented Architecture and enterprise modeling. Section 3 gives a survey of the fundamentals of the reuse of software components and existing catalogue approaches. Afterwards in section 4, we present the results of the SOA implementation at our university and discuss the extension of the service catalogue. The paper ends with conclusions and suggestions for further research.

Background

Web-service technology

The paradigm of SOA provides the basis for the present distributed application framework (W3C 2003). Software components are provided in modular and reusable services. In this context we speak about loose coupling and place independency of participating services since there are no strong logical or physical dependencies between services and involved applications (Knuth 2003).

This paper focuses on a web-service based realization of the SOA paradigm according to the recommendations of the W3C consortium (Erl 2005). The intension of the web-service concept in the present paper derives from the following allocated criteria:

- An abstract interface for embedding the web-service in high level specifications like the Web Service Business Process Execution Language (WS-BPEL) (IBM 2002), to define which messages and data types the web-service understands and in what sequence they will be processed.
- The connection to a concrete transmission protocol as the basis for communication.
- Finally the service itself with an address and the implementation of its functionality.
W3C’s layered architecture for today’s web-service technology focuses on three principle core elements described in detail by Muschamp (Muschamp 2004). The Simple Object Access Protocol (SOAP), the Web-Service Description Language (WSDL) (Gudgin et al. 2004) as well as the Universal Description, Discovery and Integration Language (UDDI) (Walsh 2002) have become de facto standards for XML messaging, web-service description and registration. In addition to these main protocols web-service composition requires higher levels of description. In the course of this demand orchestration and choreography languages base upon several high level standards like business WS-BPEL, WSFL (Web-service Flow Language) (IBM 2001), Web for business process design (XLANG) (Ardissono et al. 2003). Compared to the core protocols this high level languages take a step forward by integrating web-services in business process models. Essence is the integration of business processes across enterprise’s boundaries by modeling web-services in directed graphs in the order of their chronological sequence.

**Enterprise modeling**

In the course of business process reengineering activities modeling approaches are an important technique for an effective intervention into the problem domain (Hammer and Champy 1998). Essential benefit is attained through reduction of complexity by abstraction which facilitates analysis of complex systems (Balzert 1982; Ferstl and Sinz 1994). Hence, within the scope of first stage problem analysis and documentation, visual models are established – so called enterprise models representing business requirements for the actual enterprise situation (Fettke and Loos 2003). The semantic mightiness of enterprise modeling language constructs has to cover non-formal aspects supporting a deep understanding of the business domain and of the potential of an information technology employment as well as formal aspects in order to support the system implementation (Frank 1999). Thus, we need semi-formal languages to model problems, which are not well-structured, highly subjective, and finally not objectively well formed (Harel and Rumpe 2000).

Most grammars do not possess a sufficient number of language constructs to model all phenomena in a domain (Agarwal et al. 1999). Consequently, we might need a multitude of modeling languages to model the dynamic changing business domains (Wand and Weber 2002). For process documentation purpose, e.g., we use modeling techniques like Petri-Nets (Desel 1998) or the Event-Driven Process Chain (EPC) (Scheer 2000). On the other hand in the course of the design of executable business process, languages like the Business Process Modeling Notation (BPMN) (White 2005) have been developed. They enable business domain experts to model understandable graphical representations of business processes with the intention to generate executable Business Process Execution Language (BPEL) artifacts (IBM 2002). A BPEL file contains a XML based graph structure defining a flow of link elements to web-services. Hence, business processes can be performed by a defined flow of web-service invocations. Software that supports language independent modeling is subsumed under the concept of meta-CASE tools like MetaEdit+ (Kelly et al. 2005) or Cubetto toolset (Cubetto 2007).

**Reuse of software components**

In the course of reuse of developed artifacts and aggregated knowledge in software engineering this paper focuses on business software component reuse. In relation to the software engineering discipline Sametinger defines components as encapsulated, well-defined artifacts, describing and implementing a certain portion of the information system’s functionality (Sametinger 1997). From the perspective of a system engineer important criteria for component reusability are a good documentation (Sametinger 1996) and a component
granularity that makes sense for a model driven analysis. The documentation should provide on the one hand information about the reuse status in terms of the component provider, contact and quality level information etc. Additionally we need adequate information to select a component and estimate its suitability in a specific context (Sameting 1997). Of particular importance in this context is the aspect of a standardized way of component development and description.

**Business objects**

Business Objects are an approach to assign the object orientation paradigm not only to implementation level in software engineering but also to the level of conceptual modeling (Weske 1999). Fundamental idea is to bridge the semantic gap between conceptual modeling and technical design and implementation. The gap results from the complexity difference between user and basis machine (Ferstl et al. 1994). The user machine represents the solution in an outside perspective specified by business domain related requirements. On the other hand the developed information system and the basis machine, representing the IT fundament, specify the inner perspective (Körmeier and Esswein 1995). A business objects represents simultaneously a part of the enterprise model (outer perspective) and of the system design (inner perspective). Hence, entities within the enterprise model are directly represented as executable software units within the information system (Casanave 1997). Thus, information systems could be easily communicated, modeled, designed, shared and better prepared to market (Casanave 1997). Business objects can be classified into three different types (Faehnle et al. 1999; Shelton 1997):

- Business Entities representing concepts of the real world.
- Business Process Objects describing flows of activities within the enterprise under participation of business entities.
- Event Business Objects meaning a trigger or a result of business processes.

Within the scope of business objects, representations of real world entities are basically characterized by a distinct name and a description of their semantic and aim in their business domain. Matters of particular interest for the proposed integration framework are the description of a business object’s behavior and its relationships to other objects. The design and the implementation of business objects were originally based upon a CORBA infrastructure (OMG 1999). With the Business Object Facility (BOF) the OMG provides an infrastructure of distributed objects, communicating over standardized interfaces (IDL), using low-level CORBA middleware-services (Weske 1999). However BOF offers also general business objects and enables an information system modeller to combine them to real enterprise objects. The combination to business objects is ruled by the Business Object Component Architecture (BOCA) offering BOCA components which are in turn business or low-level objects (Tech. 1998). The semantic description of business objects behaviour can be specified using the CORBA Component Definition Language (CDL) e.g. by describing pre- and post condition of business object’s operations ruling the business process flow (Weske 1999).

**Software Components**

The design and implementation of business objects is done by one or many software components (Casanave 1997). Regarding the system engineering development process software components are the conduction of the transformation process. They are strongly capped parts of the information system. Hence, the mapping from business objects to software-components can be done without awareness about the prospective information system. This mapping between business objects and software components represents the bridging of the semantic gap.
Prerequisite for a discussion in a component oriented environment is the existence of a component model. Figure 1 illustrates a part of the component model of the CDL (component definition language) in the form of a UML class diagram (Szyperski et al. 1999). CDL provides a uniform abstraction from concrete component technologies like EJB (Enterprise JavaBeans) or COM+ (Component Object Model) or web-services.

Central element is the concept of a component. We can deduce atomic components with a direct interface to a set of operations provided by the component and implemented by associated classes. Hence, they enclose a part of the information systems implementation. Component frameworks are special atomic components for the purpose to combine general atomic components to complex components. Therefore they define a possible combination of two or more atomic components. The result is a complex components consisting of a component framework and a set of atomic components, allocated to certain hot spots. The allocation of atomic components to hot spots is ruled by contracts. A complex component does not possess an own interface, but exports the interfaces of the used component framework.

Furthermore the same concepts can be classified after their contribution to the overall architecture. Consequently we can differ between system components that implement application independent (generic) services e.g. database services, workflow management system services and business components with a clear relation to a specific business domain (Griss 2001; Rautenstrauch and Turowski 2001; Szyperski 1998).

To sum up this paper distinguishes between software components (e.g. web-services or EJB) in terms of an atomic component in CDL implementing a certain part of the information system and complex components like business objects containing a whole set of software components configured after a certain component framework.

**Component catalogue approaches**

Prerequisites for a successful component based software engineering are a technical standardization of component models like EJB, CORBA or web-services, the conceptual
standardization towards easy understanding of components purpose and the support of the development process by appropriate tools (Jaekel and Teschke 2003). Component catalogues are tools trying to foster the reuse of software components by providing functionality for archiving, inquiry and management of components. Within the scope of component catalogues various approaches for a uniform specification of software components emerged.

The universal description, discovery and integration (UDDI) is a standard for an integrated specification framework for the description of a company as well as the specification of the web-services the company provides (Cerami 2002). For the specification of a web-service the standards differs between the white, yellow and green pages. The white pages contain general information in terms of a unique name and an informal textual description. The yellow pages represent classifying information e.g. by indicating the service to a certain business domain (see e.g. UNSPSC). Finally the green pages serve as the registration of the interface and its location (Cauldwell et al. 2001). The UDDI standard focuses on physical reuse of a software component and offers language constructs for a detailed technical description. However, the evaluation of the service semantic stays restricted to the textual description in the white page. Hence, apart from the textual description there is no possibility for a domain expert to decide if the component is suitable for his business domain or not.

The technical aspect in component oriented software reuse is comprehensively explored (Bettag 2001; Szyoperski 2001). A whole range of industry standards like SOAP, WSDL or BPEL emerged; but there is still a lack on a conceptual description of business functions implemented by one or many business software components. The GI-group 5.10.13 are engaged in research of appropriate component catalogue systems that support the management of described component types (see section 3.2). The proposal for a unified specification of business components describes a comprehensive framework for the specification of the outer business perspective of a software component (Turowski 2002). The framework is not limited to the technical interface and combines in addition conceptual information which may be also relevant for our catalogue implementation to estimate the domain related suitability of a business service. Their conceptual information approach is topically structured and enables the analysis of heterogeneity between similar software components (see Figure 2).

---

**Figure 2. Approach for uniform specification of business components (Turowski 2002)**
The interface layer contains all interfaces of the services the component provides and external service interfaces the component requires to its business tasks. The approach proposes on the behavioral level the specification of the semantic of methods through predicate logic confirmation (pre and post constraints), on coordination layer the specification of service flows (Hoare 1969; Liskov and Wing 1994; Meyer 1992). Hence, the advantage of the formal specification of the first three layers is the precise definition (software architectural aspect); however the disadvantage is the hard understanding for domain experts. Apart from the quality layer the specification of the upper layers is done in natural language. Natural language is on the one hand understandable for domain experts, but has the deficit of ambiguity and insufficient precision summarized under language defects. Ortner proposes therefore the uses of norm languages in terms of a controlled language, developed through methodical reconstruction of the naturally spoken domain language (Ortner 1997).

University Implementation
The growing number of web-services and other software components at the participating universities requires an appropriate service registry. In this context we orientate on the ideas of Overhage (Overhage 2004). The component catalogue has to cover the specification of all described component types (see section 3.2) and simultaneously distinguish between them. Additionally we focus on black box reuse meaning that the catalog user relies on the information of an outside perspective of the encapsulated implementation. The configuration of the SOA is performed without knowledge about concrete implementation details (Szyperski 1998). Thus, an extensive and functional description of the outer component perspective is important as we attach great importance to a university wide understanding of catalogued components. Hence, additionally to a standardized technical specification with UDDI we require conceptual information in a notation that allows a comparison of different specified components. An entire specification has to contain all necessary data to rate a component e.g. the semantic or the performance (Gruntz and Pfister 1998; Szyperski 1998).

Most problematic is to adjust the semantic and pragmatic heterogeneity between components. Semantic heterogeneity means that components implement domain specific concepts in different ways. To set an example the concept of a “student account” is comprehended differently between two components. Therefore Overhage proposes an explicit register of concepts together with their definitions and recommends the reuse of components that implements the same conceptual standard (Overhage 2004). Pragmatic heterogeneity references the varying understanding of components about the component internal and external business process flow (Overhage 2004). A solution could be the adoption of a uniform business component framework which implements common business flows. However, we focus on an appropriate documentation of the business process part implemented by the business component.

Project description
The introduction of a university wide information system based on the service oriented architecture (SOA) paradigm is part of the project MIRO. MIRO was launched November 15, 2005 and is funded by the German Research Foundation (DFG). During a period of five years the complex structure of one of Germany’s largest universities is to be adequately provided with an innovative and powerful system for information management. In doing so, the requirements of the Bologna process (European project for creating academic degree standards) and of modern e-government must be taken into consideration. The same holds for the needs and wishes of about 40000 students, 5000 employees in 15 departments offering
more than 130 study programs. MIRO strives to improve the information infrastructure in a systematic and continuous way to enhance the efficiency of key activities of the university (MIRO 2005).

The MIRO project consists of 10 work packages each of which is divided into several subtasks, 69 in total. Main tasks are the development of university’s organization and services to a service-oriented architecture, the implementation of a central identity management, university portal and the enhanced development of innovative information retrieval techniques to improve on the efficiency of research, teaching and study by providing an infrastructure for information search and delivery. The authors of this paper are concerned with the analysis of core business processes and the introduction of supporting information and communication technology (ICT). The process analysis visualizes the different processing steps and information flows within the various administrative units of the university. The introduction of ICT is combined with the development of a service-oriented architecture (SOA) and serves to flexibly support the university’s process organization.

University wide services will emerge performed manually or with IT support. To improve the allocation and reuse of these services inside and beyond university borders we developed a service catalogue to register and manage central university services. Moreover to support the reuse already in the early system engineering phases we propose a model driven integration of services into an enterprise modeling environment. Therefore, the service registration demands a conceptual description of the supported business process formulated with a standardized portion of language constructs.

The results of the project will be communicated to other universities encouraging them to adopt all or some of the solutions and to adjust them according to their own needs. For some results, this might also apply to companies and public administrations.

**Cataloguing of web-services**

Our actual situation requires a university wide register for developed web-services and a documentation of their behavior. This is necessary as we recognize a growing number of participating web-services in the university software architecture caused by an increasing development activity. Hence, the current implementation focuses on the software architectural aspect of component specifications (see Figure 2).

Problems that occur in this context are XML data transformation, a robust and reliable transport from endpoint to endpoint, intelligent message routing as well as business process orchestration based on languages like BPEL (Luo et al. 2005).

The solution could be an enterprise service bus (ESB) in terms of a central architecture control. Regarding the registry problematic some ESB offer an implementation of a business service registry (Luo et al. 2005). In order to perform routing of service interactions, the ESB requires basic routing information provided by a Namespace Directory or more simple by a routing table. The business service registry in contrast provides details of services available to perform business functions. We prototypical use the IBM ESB and its business service registry in terms of an open-standard UDDI directory to publish the availability of our services and encourage their reuse.

As the UDDI standards lacks on a semantic description of web-services we assign standardized business function types to registered web-services. Thus, additionally to the textual description in UDDI we propose a business function assignment (see Figure 3).
A business function type covers the certain part of the business model that is implemented by a web-service (see next section). A university wide set of business functions types solves the problem of semantic heterogeneity and fosters a higher degree of a university wide common understanding of the web-service semantic. By cataloguing the web-service the software architect is specifically restricted in his expressive power as only the predefined set of business function types can be instantiated. As a result the web-service is mapped on his business purpose and we are no longer limited to its textual description.

Future aim is a special university meta-schema as a central architecture element giving a more comprehensive frame for a specification of a component following the layer architecture in section 3.3. To sum up, the first implementation covers the registration of all web-services of the university and assigns to each one a business function as its conceptual outer perspective.

**Using business objects within enterprise models**

The construction of an enterprise system based on prefabricated business components tends to result in new system engineering process steps (Griffel 1998). Basically it is about the identification of the potential part of the business process domain that should be covered by reused business components and the integration of the implementing software components to the new information system. To solve the problem of semantic and pragmatic heterogeneity we propose an integration of high granular business process blocks representing components into enterprise models.

In the first step our catalogue is focused on the reuse of university’s ICT software components that are generic in terms of not supporting the activities of only one special administraitonal unit. Instead these components are related to service-unspecific tasks and supporting activities (Becker et al. 2006). Examples are the university’s Workflow Management System (WfMS), the Document Management System (DMS), the Print-and-Pay
System, the University Portal (myWWU) or the Identity Management System used by a broad area of application within the entire process landscape.

The assignment of software components to university services can be done over the business function concept (Becker et al. 2006). Analogical to the PICTURE approach of Becker et al. a business function constitutes the smallest granularity unit of the conceptual description of a university service component and at the same time the connection between information system and enterprise model. Hence, a business component comprises one or more standardized university business functions that are assigned to a one or more software components. The business component “inquiry”, e.g., is made up on the business functions “edit student library account” and “search book”. The aim of standardized business functions is to normalize the content of the conceptual description of the ICT components in order to increase its comprehensibility to other domain experts. Varying business functions are subsumed in one business function type with possibilities for a parameterization.

In turn business function types are assembled to ICT components illustrated in figure 4. ICT components behavior is carried out by a certain amount of business process blocks. This block would normally be described in detail using a modeling language like EPC in ARIS (Scheer 2000). However this kind of description of the component behaviour would not enable an easy reuse (Becker et al. 2006). Instead of providing a semantically rich business process model, we reduce the semantic and pragmatic heterogeneity. The block represents sequentially modeled business function types. A business function has to be instantiated through the parameterization with certain values. The modeling methods documents the components business process by using predefined, university domain specific business functions types from a repository. The high granular description prevents the refinement of model elements and the modeling of conditional ramifications in the process flow. For a complete picture about coarse-grained modeling languages we reference to Becker et al. (Becker et al. 2006).
4. Coarse-grained modeling following the ideas of PICTURE (Becker et al., 2006)

Figure 4 illustrates the encapsulation of a modeled EPC diagram into several process building blocks representing types of business functions. Thus, the modeling of the components behavior is simplified and the result is easier to understand for domain experts. Hence, we build upon the modeling approach of Becker et al. but within the context of software component description.

Discussion
At present only a few component catalogue implementations manage components that contain a conceptual outside perspective (Gruntz et al. 1998; Turowski 2002). The approach presented in this paper builds upon the ideas of the approach for a uniform specification of business components and the standardization activity with UDDI. The authors investigate further research in a component description that enables the integration of components into enterprise models with the intention to reuse SOA components already in the early analysis stage of the system engineering process.

The question about why to integrate web-services in visual models is equivalent to the question about the motive for introducing OMG’s Model Driven Architecture (MDA) in the earlier stages of software engineering (OMG 2003). Visual modeling and model transformations play the key roles in MDA, which is an approach that focuses on IT system specification, separating system functionality specification from the specification of the implementation of that functionality.

Since we think that the MDA idea for web services can partly be extended also for requirements analysis, this paper introduced a framework for building up a service catalogue to foster a model-driven configuration of SOA components. Combined with business models the service catalogue describes the structure and functions of ICT components while
abstracting away technical details.

Until now, we realized the model-driven assignment of web-services to business function types. Our future work will focus the integration of ICT components into business process models. Therefore, we will extend an existing modeling grammar. Afterwards, we will implement the grammar using a meta-CASE tool, connected to the electronic service catalogue.

References
Cerami, E. Web Service Essentials O'Reilly, Sebastopol (California), 2002.


Harel, D., and Rumpe, B. "Modeling languages: Syntax, semantics and all that stuff, part I: The basic stuff.," Weizmann Institute Of Science.


Luo, M., Goldshlager, B., and Zhang, L.-J. "Designing and Implementing Enterprise Service Bus (ESB) and SOA Solutions," IEEE International Conference on Web Services (ICWS'05), 2005.


OMG "MDA Guide, version 1.0.1."


Shelton, R. "OMG Business Object DTF: Common Business Objects."
Turowski, K. "Vereinheitlichte Spezifikation von Fachkomponenten," Gesellschaft für Informatik (GI), Augsburg.