Servicebase Management Systems: A Three-Schema-Architecture for Service-Management

Matthias Fischer, Marco Link, Erich Ortner, Nicole Zeise

Fachgebiet Entwicklung von Anwendungssystemen
Technische Universität Darmstadt
Hochschulstr. 1
D-64289 Darmstadt
(fischer, link, ortner, zeise)@winf.tu-darmstadt.de

Abstract: The development of service-oriented applications has reached a point where more specifications are needed to enable holistic transparent service-management. A three-schema-architecture for design of servicebase-management-systems according to the ANSI/SPARC-model for database design is proposed in this paper. The conceptual schema of the proposed architecture decribes the servicebase, what provides a consistent overview and simplifies the localization of services. The external schema is used to provide a consistent access to the servicebase by customers. The internal schema provides the concept of independence of implementation and execution of services. The architecture aims at offering a platform where service-vendors and service-customers have the chance to trade services with access and usage independence as well as implementation and execution independence based on a comprehensive and clear conception and architecture.

1 Introduction

Market and enterprise dynamics have become one of the most popular topics discussed in context of business management. In this scope process-orientation was a first step for companies to smooth their way to flexibility. But for a dynamic adaption in changing environments more than adapting processes is needed. Therefore it is necessary to adapt existing IT systems and to implement new requirements as far as possible. In this area, which is not just focused on technology, a new direction of science has been developed: a service science. So the first question we have to answer is: what is a service? In the present context a service is understood as a piece of code which transforms input data into output data by specified methods. To allocate, manage and organize the mass of atomic and composite services of companies in a transparent way, servicebase-management-systems are needed. One of the most lasting achievements in the area of information technology standards is the three-level-architecture for database design by ANSI/SPARC. It also is a useful framework to define a standard schema for the description of services. First, we will show the three-schema-architecture for servicebase-management in general. In the next section we present how the layers of the architecture work and give a big picture.

2 A Three-Schema-Architecture for Service-Management

For the reasons given in the introduction of this paper, we propose an organizational structure for services which is inspired by the three-schema-architecture for databases. The architecture proposed is foundational for the design of servicebase-management-systems. Besides the construction and administration of a conceptual service-schema, it is necessary to deal with the implementation and execution of the services (by using an internal schema) as well as with the composition and usage of the services by customers and users (by using an external schema) of the services. These three schemata are elaborated as layers of the three-schema-architecture of services in the next subsections.

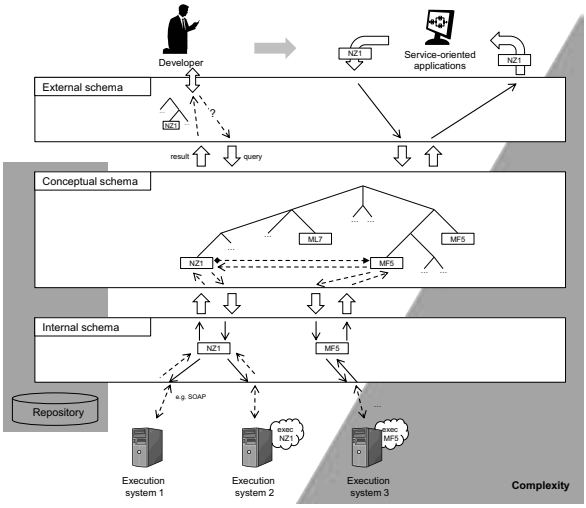

Figure 1: Interaction between the three layers of a servicebase-management-system

The aim of the layered structure is to reduce complexity, i.e. starting at the lowest layer with highest complexity and reducing it up to the top layer which is closest to the service-customer. To achieve this effect, the layers are encapsulated and can only communicate using well defined interfaces. There are three goals pursued by three layers: first consistent access (external schema) to ease the use of services. Secondly, a consistent description and model of each of the services (conceptual schema) to be able to depict a consistent schema of all services and to ease search for suitable services for a certain application is provided. And third creation independence, to be able to improve single services without adapting the applications using that service (internal schema). Due to the brevity of the paper, the functions of each layer (and the corresponding interfaces) are described in a general and implementation-independent way. An implementation-specific description depends – in addition to goals and functions – far too much on technology used to give a general description. There has to be an effort to develop a language for each of the several layers respectively tiers in the architecture. That language can be used to model concrete objects on the next lower layer of the architecture for service-management.

Before introducing and motivating the three layers of the proposed archictecture, figure 1 depicts the interactions of the three layers to be presented.
Two different kinds of agents can use the servicebase-management-system: on one hand the developer of process-centric applications and on the other hand the applications used in processes themselves. Developers use a servicebase-management-system as a support-system in locating appropriate services for defined scenarios and to get a consistent interface (the external schema) to access the services selected. Applications use this interface to access the required services. In general this means that the suggested architecture of the external schema provides a possibility to relay received requests of applications to the conceptual-service-schema.
The conceptual-service-schema is used for organization of the logical service schema itself. It provides a categorical overview of the set of accessible services and simplifies the search for appropriate services by that. If applications using a servicebase-management-system are running, the conceptual-service-schema is used to map the administrated logical service schemata to the service objects in the internal schema. The internal schema is used to manage the physical service instances. It instantiates and calls the distinct services. It is responsible for keeping SLAs. Furthermore the internal schema manages all resources available for the servicebase-management-system.

3 Conceptual Service-Schema: Consistent Administration of Services

The purpose of the conceptual schema of servicebase-management-systems is to give an organized representation of all services available. That includes a centralized management to ease the search for suiting services for a given task. Therefore, consistent descriptions of the services are made available. That kind of description includes inter alia semantics of the service (which task can be performed by the service?) and further pragmatic information, like possible fees for the use of the service, expected response-time etc. The interface defines what kind of information or data the services needs to be executed. If there are side-effects of a service (like changes in a database or of some business rules), these have to be documented, too. All documentation and description of services aims at easing up the search for suitable services, performed by programmers or even by artificial-intelligence-agents.
In opposite to the conceptual schema of databases, redundancy is no harm in context of the logical representation within the conceptual schemata of servicebase-management-systems. More than that, redundancy can make the search for services by humans even easier, since not all services can be categorizedin a unique manner. The connection of logical service-objects to physical instances of the services is made using the interface of the internal schema (using metadata from the repository). The conceptual-service-schema just has to care about the connections of the logical service-objects to the corresponding instance of the internal schema. This way, an integrated consistent schema of all services can be made available while guaranteeing the physical independence of the services from each other.
While concrete access is done using the external schema, the conceptual-service-schema

connects to the internal schema using a repository. This way, the schema – and applications making use of it to connect to services – does not have to be changed if the physical place of a service changed. Access to concrete physical instances of a service is made by the internal schema to achieve independence of the services.
From the goals stated for the conceptual-service-schema of a servicebase-management-system, several tasks and requirements can be derived. The available services are represented as logical objects in the conceptual-service-schema. Part of the logical representation is metadata that describes possible connections from one service to another (which services are used by a certain service and which services make use of the service itself). With respect to services and their representation in the conceptual-service-schema, one has to pay particular attention to composite services. Since a servicebase-management-system that handles only atomic services is not very useful, composite services have to be made available. Therefore, properties of composite services have to be taken into account. These are inter alia dependencies of the composite services from the services used (which may be atomic or composite themselves), the prerequisites resulting from the use of other services, the parameters needed and the postconditions resulting from the services used.
To attain the goals stated and to deal with the deduced tasks, there are already some solutions available. To describe the properties of services, the Web Service Description Language (WSDL) might be used [W3C07]. The overall system including dependencies of the services may be modeled using UDDI (Universal Description, Discovery and Integration) [Com02].
But there are parts of the system that cannot be covered with existing solutions. These parts include the presentation of the overall conceptual-service-schema (description of all services and their connections). This is a vital point, since not only with respect to service discovery problems – the automated search for suitable services – the semantics of services have to be described unambiguously. Developers of applications have to be enabled to recognize and choose the correct services for their purpose. That is why the overall service-schema has to be presented in an organized and well documented form. Semantic integration has been solved with respect to databases by making use of an object-oriented rational language (cf. [WOI04]).

4 Internal Schemata: Implementation and Execution of Services

In context of architectures for database-management-systems, an internal schema specifies how and where data has to be stored physically with regard to effective and efficient access during its use [CB05]. Analogous, an internal schema for services describes how specific information about service-execution can be used and where it is stored. Service implementations are accessible via interfaces and include data and execution logic [KBS08].
Using standardized protocols to exchange messages (e.g. SOAP) [Fro07], a computer-internal software- and hardware-independent representation of services can be an interface. Coming along with this, the independence of implementation and execution makes the portability of this architecture possible. In line with concepts of databases, common used or critical elements can be operated on specific (e.g. specific requirements on securi-

ty or performance) physical execution systems. In case of performance problems or even failures, internal adjustments are possible, regardless of the conceptual and external schema. The servicebase-management-system can generate physical copies of affected services in its internal schema – if necessary – to guarantee the response times of certain services declared in existing SLAs during possible performance bottlenecks. In case of a composite service within the internal schema, which may act as a service-customer, the service access has to take place by using the conceptual-service-schema. If a service would not comply with this rule, the goals of independencies could not be achieved. Within the internal schema of a servicebase-management-system, the principle of encapsulation must be preserved. The way of using services by services via the conceptual schema is necessary to preserve the logical independence of individual services, as they would have to be adjusted with a change of the services used. This holds in analogy to the service-customers.

5 External Schemata: the Usage of Services

The external schemata which are closest to the customer, allow a user optimized dynamic construction of applications, based on provided services of the conceptual-service-schema. Therefore it is necessary to develop a specific language, which allows the individual composition of services provided by the conceptual-service-schema to create a holistic application system. This implies that the language has to target the creation of synergies between the services provided.
In addition to construction of applications, the external schema offers a consistent interface for the use of embedded services in applications. The interface is consistent because it is the central instance for addressing services, which is available by the servicebase-management-system. Particular addressing of concrete services always stays service-dependend, it e.g. depends on parameters required by the service. The advantage of the proposed servicebase-management-architecture is that physical and logical relocation of a single service (which has to be called by a new address) requires no adaption of applications which use this service, if the logical address of the service management system does not change. For all practical purposes, the proposed architecture could be implemented according to the implementation of different database-management-systems as an application which uses JPA (Java Persistency API) [BK07]. This way, drivers for each application using the servicebase-management-system have to be installed. In case of using another servicebase-management-system, or in case of the change of an address of the used servicebased-management-system, there is no effort required to change the application. Only the driver, which connects the application with the service-management-system, has to be replaced.

6 Conclusion

A servicebase-management-system would excellently complement the range of systems for service-oriented and process-centric application systems. Key challenges for computer scientists in this area are, for instance, the overall control of the process execution, an appropriate interaction between systems and humans and an effective and efficient management and execution of all necessary software artifacts. There is currently no integrated tool for service-discovery and their dynamic execution available. These tasks are often covered by an Enterprise Service Bus system [KBS08] in conjunction with standards such as UDDI or WSDL. A consistent and integrated approach cannot be identified at this stage. For this reason, the current work could be used as basis for an ideal complementary system in the context of SOA tools and concepts. Based on the requirements of changes in organizational concepts of entire enterprises [Mar10], for the future it has to be proofed, that a portability of this concept is possible to services, which are not based on IT elements.
As future work, the detailed technical preparation of the three schemata as well as an implementation of a prototype system can be identified. Related to that, a language, comparable with SQL, for querying and modifying the conceptual service schema will be necessary. For a long-term view, we aim to transfer this architecture also to non-technological services. For this case the system serves as a schema-repository for dialog- and interaction-schemas (for language and physical actions). Instead of computer systems, humans are the executing entities.

Bibliography

[BK07] Christian Bauer und Gavin King. *Java persistence with Hibernate: Revised edition of Hibernate in action*. Manning, Greenwich, Conn., 2007.

[CB05] Thomas M. Connolly und Carolyn E. Begg. *Database systems: A practical approach to design, implementation, and management*. Addison-Wesley, Harlow, 4. ed.. Auflage, 2005.

[Com02] UDDI Committee. UDDI Version 2.04 API Specification: UDDI Committee Specification, 19 July 2002, 2002.

[Fro07] Thilo Frotscher. Der Webservices-Technologiestack. In Gernot Starke und Stefan Tilkov, Hrsg., *SOA-Expertenwissen*, Seiten 489–506. dpunkt.Verl., 2007.

[KBS08] Dirk Krafzig, Karl Banke und Dirk Slama. *Enterprise SOA: Service-oriented architecture best practices*. Prentice-Hall, Upper Saddle River, NJ, 7. print.. Auflage, 2008.

[Mar10] Wolfgang Martin. Service-orientiertes Business: Change von Business/IT-Alignment. *it management*, (01):16–19, 2010.

[W3C07] W3C. Web Services Description Language Version 2.0 Part 1: Core Language, 2007.

[WOI04] Hartmut Wedekind, Erich Ortner und Rüdiger Inhetveen. Informatik als Grundbildung: Teil IV: Objektsprache/Metasprache. *Informatik Spektrum*, 27(5):459–466, 2004.