Data & Knowledge Engineering 70 (2011) 794–819

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

View-based model-driven architecture for enhancing maintainability of data access services

Christine Mayr [a,*], Uwe Zdun [b], Schahram Dustdar [a]

[a] Distributed Systems Group, Information Systems Institute, Vienna University of Technology, Austria
[b] Faculty of Computer Science, Research Group Software Architecture, University of Vienna, Austria

A R T I C L E   I N F O

Article history:
Received 3 August 2009
Received in revised form 18 May 2011
Accepted 18 May 2011
Available online 27 May 2011

Keywords:
Data Access Service
DAS
SOA
Service
DAO
Data Access Object
Repository
Model-driven
MDD
Data model
Data
Database
RDBMS
Tooling
Architecture
Ecore
ORM
View-based
View
EMF
GIS
WFS
Web Feature Services
Geographic
Spatial
Non-spatial
Features
Data access
Data
Model-driven architecture
Views

A B S T R A C T

In modern service-oriented architectures, database access is done by a special type of services, the so-called data access services (DAS). Though, particularly in data-intensive applications, using and developing DAS are very common today, the link between the DAS and their implementation, e.g. a layer of data access objects (DAOs) encapsulating the database queries, still is not sufficiently elaborated, yet. As a result, as the number of DAS grows, finding the desired DAS for reuse and/or associated documentation can become an impossible task. In this paper we focus on bridging this gap between the DAS and their implementation by presenting a view-based, model-driven data access architecture (VMDA) managing models of the DAS, DAOs and database queries in a queryable manner. Our models support tailored views of different stakeholders and are scalable with all types of DAS implementations. In this paper we show that our view-based and model driven architecture approach can enhance software development productivity and maintainability by improving DAS documentation. Moreover, our VMDA opens a wide range of applications such as evaluating DAS usage for DAS performance optimization. Furthermore, we provide tool support and illustrate the applicability of our VMDA in a large-scale case study. Finally, we quantitatively prove that our approach performs with acceptable response times.

* Corresponding author.
E-mail addresses: christine.mayr@inode.at, christine.mayr@brz.gv.at (C. Mayr), uwe.zdun@univie.ac.at (U. Zdun), dustdar@infosys.tuwien.ac.at (S. Dustdar).

0169-023X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2011.05.004

1. Introduction

In modern process-driven service oriented architectures (SOAs), process activities can invoke services in order to fulfill business requirements. A service offers a well-defined interface specified by using a web service description language (WSDL) [60]. Besides invoking services, process activities can perform human tasks, do transformations and/or invoke other process activities. Service repositories [20,12] can be used to manage services and support service discovery at runtime. As shown in Fig. 1, the process activity queries a service repository in order to find a suitable service for dynamic invocation *(1)*. Typically, services need to read or write data from a database. Nowadays, this data access is done by so-called data access services (DAS). DAS are variations of the ordinary service concept: They are more data-intensive and are designed to expose data as a service [54]. They can either be invoked by another service or by a process activity directly. As depicted in Fig. 1, a service repository returns a service, that is running on a DAS provider. Eventually, the process activity dynamically invokes the service on a certain DAS provider *(2)*.

In object-oriented environments, DAS commonly use a layer of data access objects (DAOs) to read and write data from a relational database management system (RDBMS). According to the JEE pattern catalog [43], the DAO pattern abstracts and encapsulates all access to the data source and provides an interface independent of the underlying database technology. The DAO manages the connection with the data source to obtain and store data.

1.1. Status quo

A process-driven SOA is an architectural style for developing large business applications. Accordingly, a huge number of processes, process activities, services, and in particular data access services need to be managed. Nowadays, business process execution languages such as [37] are used as the missing link to assemble and integrate services into a business process [21]. These business process execution languages provide higher level control for services as they describe the services to be invoked and which operations should be called in what sequence. In order to maintain and integrate processes and services, much research work has been done. However, these business process languages do not integrate the semantics of an invoked service such as which DAS reads or writes which data. In contrast, they rather regard the process internal data read and written by process activities.

1.2. Basic problem

Unfortunately, the relationships between the DAS, the underlying DAOs, and the data storage schemes are not sufficiently explored, yet. Fig. 2 overviews these missing links: The DAS in the service repository are neither associated with the DAS source code, nor with the service internal documentation, nor with the data storage schemes. Accordingly, the service internal documentation in the middle of the figure is loosely coupled with the DAS, the DAS source code and the data source schemes. However, from our experiences, in order to efficiently maintain DAS, a further integration of the DAS, the DAS source code, DAS documentation, and the data storage schemes is compulsory. In the following we describe the related problems experienced when maintaining, reusing, documenting, tracing and developing data access in a large enterprise in more detail.

1.3. Difficult maintainability

In organizations, usually data storage schemes are subject to changes. In order to efficiently maintain DAS, it is important to know which DAS are concerned by this change. If a database table schema is redesigned e.g. in case of altering a column, it is essential to find all DAS that read or write data from this table in order to adapt them. Accordingly, if a table is dropped, some DAS may be obsolete and should be not be available anymore. Due to lacking integration of DAS and the data storages, further elaboration to improve maintainability of DAS is required.

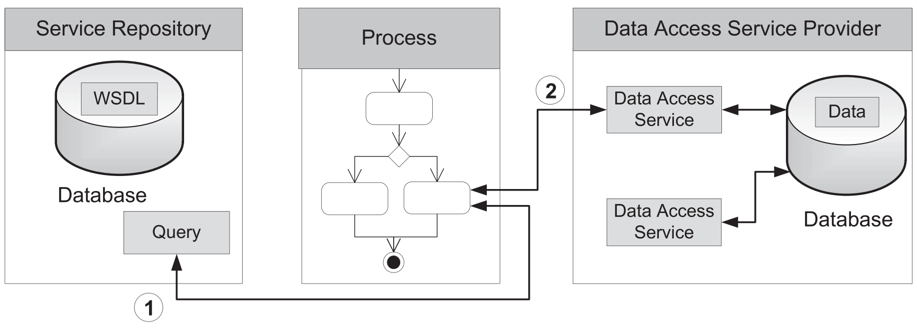

Fig. 1. Data access in a process-driven SOA.