The Adaptation Model of a Runtime Adaptable DBMS

Florian Irmert, Thomas Fischer, Frank Lauterwald, Klaus Meyer-Wegener

Friedrich-Alexander University of Erlangen and Nuremberg
Department of Computer Science
Chair for Computer Science 6 (Data Management)
Martensstrasse 3
91058 Erlangen, Germany
{florian.irmert, thomas.fischer, frank.lauterwald, kmw}@cs.fau.de

Abstract. Nowadays maintenance of database management systems (DBMSs) often requires offline operations for enhancement of functionality or security updates. This hampers the availability of the provided services and can cause undesirable implications. Therefore it is essential to minimize the downtime of DBMSs. We present the CoBRA DB (**Co**mponent **B**ased **R**untime **A**daptable **D**ata**B**ase) project that allows the adaptation and extension of a modular DBMS at runtime. In this paper we focus on the definition of an adaptation model describing the semantics of adaptation processes.

1 Introduction

In recent years the database community has realized that common database systems do not fit into every environment [6]. The obvious solution is the development of specialized DBMSs for each environment. However this approach is not suitable with respect to development cost, time to market and maintenance.

Tailor-made DBMSs [5] try to answer this challenge by adapting a DBMS towards a specific environment by providing a common code base from which customized DBMSs may be derived. Changing the functional range of such a DBMS however requires a shutdown and redeployment of a new version. Taking an application offline is often not feasible in some environments.

In the CoBRA DB (**Co**mponent **B**ased **R**untime **A**daptable **D**ata**B**ase) project [2] we propose an approach to tailor a DBMS at runtime. It uses a kind of DBMS "construction kit" and basic modules that are necessary in every DBMS. A DBMS can be assembled by choosing the appropriate modules for the intended functionality of the system. An important challenge is the modification of modular DBMSs at runtime. In our prototype it is possible to add, exchange, and remove modules while the database system is running. As a foundation we have developed an adaptation framework [3] that provides the exchange of components at runtime in a transparent and atomic operation. In this paper we present the adaptation model of CoBRA DB and its adaptation types.

The original publication is available at www.springerlink.com (DOI: 10.1007/978-3-642-02843-4_19)

2 CoBRA DB Runtime Environment

Runtime adaptation as a prerequisite to the proposed adaptable DBMS enables addition, removal and exchange of DBMS components at runtime without any downtime of the whole system.

To meet these requirements we have designed a runtime environment [3] based on a service-oriented component model [1,4]. The runtime environment is based on the OSGi Service Platform [4]. A component in the CoBRA runtime environment implements at least one service. The architecture of the CoBRA DB itself with its functional properties must therefore be sufficiently described by its services. To enable the adaptation to different environments, components implementing the same service can be exchanged at runtime. The resulting requirements for runtime adaptation are met by a transparent dynamic proxy concept which ensures the atomicity of the adaptation and the consistency of the state transfer from one component to the replacing one. Details can be found in [3].

3 Adaptation Model

Based on the prerequisites given by the runtime environment we developed an adaptation model in order to formalize the constraints for possible adaptations. This includes the structural model of the CoBRA DB architecture as well as the modeling of the different adaptation possibilities.

For the concrete specification of the adaptation model of the CoBRA DB we have chosen a service-oriented point of view. Figure 1 shows the structure of our model which consists of three design levels. The functionality of the DBMS is modeled by the specification of services (L3) like a *PageService* providing page-oriented access. The service descriptions of L3 must remain valid, even over adaptations. One or more services are embodied as components on L2, which are themselves implemented in one or more classes (L1). The components defined on L2 are the object of adaptation.

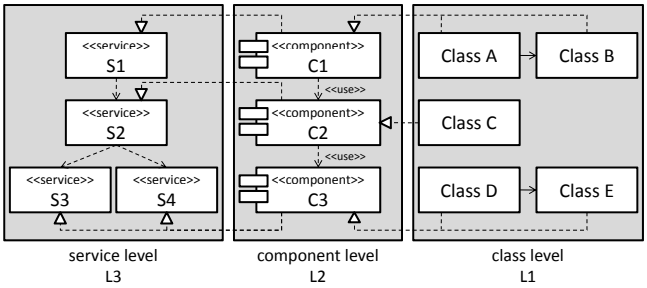

Fig. 1. Model hierarchy

To reflect the different changes possible in this model hierarchy we have identified different adaptation types.

3.1 Adaptation by component exchange

Components that implement the same services are interchangeable through *adaptation by component exchange* and may therefore only differ in their non functional properties. In figure 2 service P is realized by component C1. During the adaptation C1 is replaced by component C2 (which also implements service P).

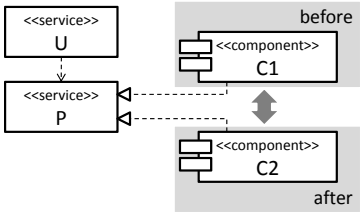

Fig. 2. Component exchange

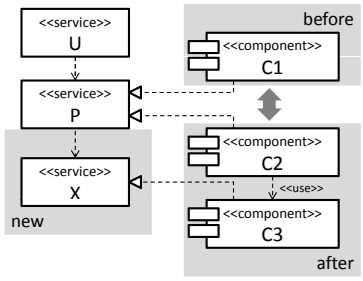

Fig. 3. Addition of a service

An example for this adaptation type is the exchange of the buffer replacement strategy, e.g. a FIFO strategy can be replaced by LRU (both implement the same service, but with a different algorithm).

Changes on L3 are also possible by component exchange on L2. Figure 3 shows the addition of a new service X implemented by component C3. Service P is realized by both component C1 and component C2 with the difference that C2 needs C3. Therefore C2 may only be deployed if a component that implements X is available. Obviously it is crucial to serialize the install operations in a suitable manner (C3 has to be installed before C1 can be replaced by C2). Removal of services on L3 is performed analogously.

3.2 Addition/removal of decorator components

Another adaptation type is the addition or removal of decorator components. The scenario depicted in figure 4 shows the addition of a component C2 which acts as a decorator for C1. Both implement the same service P but the addition of C2 improves the non functional properties for the realization of P. An example is the addition of a buffer component to an I/O component in order to speed up the response time of P.

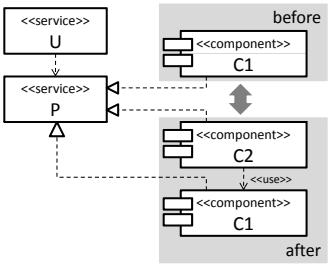

Fig. 4. Addition of a component

4 Conclusion and future work

This paper introduces an approach to model adaptation in a DBMS. We specify a model to define a common set of functionality which can be tailored to different scenarios by composition of their implementing components and the adaptation of these components can be performed at runtime.

Our current research focuses on the finalization of a full-fledged runtime adaptable DBMS for embedded systems to evaluate the adaptability in more real world scenarios, especially in the field of pervasive computing. Another challenge is the definition of transaction semantics in adaptive systems and the enhancement of the framework to cover crosscutting concerns in order to build an adaptable transaction management.

References

1. H. Cervantes and R. S. Hall. Autonomous adaptation to dynamic availability using a service-oriented component model. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 614–623, Washington, DC, USA, 2004. IEEE Computer Society.
2. F. Irmert, M. Daum, and K. Meyer-Wegener. A new approach to modular database systems. In *Software Engineering for Tailor-made Data Management*, pages 41–45, 2008.
3. F. Irmert, T. Fischer, and K. Meyer-Wegener. Runtime adaptation in a service-oriented component model. In *SEAMS '08: Proceedings of the 2008 International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, May 2008.
4. OSGi Alliance. OSGi Service Platform core specification, release 4, August 2005.
5. M. Rosenmüller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, T. Leich, O. Spinczyk, and G. Saake. FAME-DBMS: Tailor-made data management solutions for embedded systems. In *Proceedings of the EDBT'08 Workshop on Software Engineering for Tailor-made Data Management*, pages 1–6. University of Magdeburg, 2008.
6. M. Stonebraker and U. Cetintemel. "One Size Fits All": An idea whose time has come and gone. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 2–11, Washington, DC, USA, 2005. IEEE Computer Society.