Improving Data Independence, Efficiency and Functional Flexibility of Integration Platforms

Matthias Böhm[1], Jürgen Bittner[2], Dirk Habich[3], Wolfgang Lehner[3], and Uwe Wloka[1]

[1] Dresden University of Applied Sciences, Database Group
mboehm@informatik.htw-dresden.de
wloka@informatik.htw-dresden.de
[2] SQL Gesellschaft für Datenverarbeitung mbH Dresden
juergen.bittner@sql-gmbh.de
[3] Dresden University of Technology, Database Technology Group
dirk.habich@inf.tu-dresden.de
wolfgang.lehner@inf.tu-dresden.de

Abstract. The concept of Enterprise Application Integration (EAI) is widely used for integrating heterogeneous applications and systems via message-based communication. Typically, EAI servers provide a huge set of specific inbound and outbound adapters used for interacting with the external systems and for converting proprietary message formats. However, the main problems in currently available products are the monolithic design of these adapters and performance deficits caused by the need for data independence. First, we classify and discuss these open problems. Second, we introduce our model-driven DIEFOS (data independence, efficiency and functional flexibility using feature-oriented software engineering) approach and show how the feature-based generation of dynamic adapters can improve data independence, efficiency and functional flexibility. Finally, we analyze open research challenges we see in this context.

Keywords: Enterprise Integration Platform, Application Integration, Adapter Architecture, Dynamic Adapters, DIEFOS Approach

1 Introduction

The trend towards heterogeneous environments comes with an increase in importance of Enterprise Application Integration (EAI). Such an integration platform consists of a set of inbound adapters, a core message broker and a set of outbound adapters. The large number of supported external system types results in the need for data independence (independent-system-type data representations for internal processing) and, simultaneously, for efficient integration task processing (minimum overhead for data independence). These requirements—but particularly the first one—typically result in very generic inbound and outbound adapter architectures. There, the architecture of such adapters is quite monolithic, which results in low functional flexibility of such software components. This means that for each external system type, a single adapter is needed, though specific functional modules could be reused. An example for this is a TCP connection handler which sends the specific messages to the physical target systems—it might be reused by several adapters like HL7 and B2MML adapters.

In order to solve this problem of monolithic adapters (which affects the functionality as well as the performance), we describe the problem characteristics in Section 2 from a pragmatic perspective, influenced by the commercial enterprise integration platform TransConnect®. Further, we propose our DIEFOS approach and explain its core phases in Section 3. In general, one of the main questions in this context is whether or not model-driven approaches can be applied in the field of application integration. Finally, in Section 4, we conclude our paper and highlight open research challenges we see.

Although there is a lot of related work concerning MDA techniques [1] and MDA tools (e.g., AndoMDA, MOFLON [2] and Fujuba), only a very low support for model-driven development can be recognized in application integration platforms (e.g., SQL GmbH TransConnect, SAP XI, BEA Integration, MS Biztalk and IBM Message Broker). In this context, the so-called RADES approach [3] tries to give an abstract view on EAI solutions using technology-independent and multi-vendor-capable model-driven engineering methodologies. Unfortunately, this approach does not focus the problems considered here (data independence, efficiency and functional flexibility). Further, also approaches for automatic generation of Web service adapters [4–6] and BPEL adapters [7]. These techniques are too specific to the integration technology used. In addition, the semi-automated generation of adapters for legacy applications is addressed in [8]. However, such a semi-automated approach is not suitable. The dynamic adapter generation approach [9] addresses the dynamic adding of new data sources and their invocation rather than the functional flexibility of adapter generation.

2 Problem Description

Here, we introduce a generalized EAI server architecture and describe the addressed problems. As illustrated in Figure 1, an EAI server consists of typical components. There is a set of `Inbound Adapters`, which listen passively to incoming messages and convert these into internal representations. Further, the internal messages are processed by the runtime environment. This environment uses a set of `Outbound Adapters` to actively interact with external systems. According to the layers of transformations [10], the adapters realize the layers *transport* and *data representation*. The main problem is the monolithic adapter architecture with very generic message interfaces, which cause the use of uniform message representations (e.g., XML messages). This also causes the problem of *P1: Poor Performance*. Further problems include *P2: Functional Restrictions* (chosen technology), *P3: Development Effort* (redundant functionality) and *P4: Data Independence* (dependencies between adapter interactions). To overcome these problems, message representations (alternative representations, schemas) as well as adapter architectures (generic adapters, adapter generation) have to be reconsidered. We follow an adapter generation approach that allows different alternative message representations.

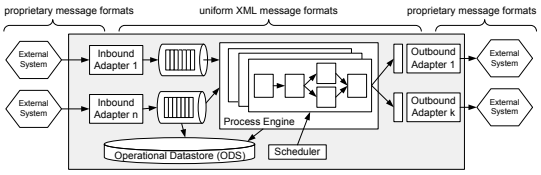

Fig. 1. Generalized EAI Server Architecture

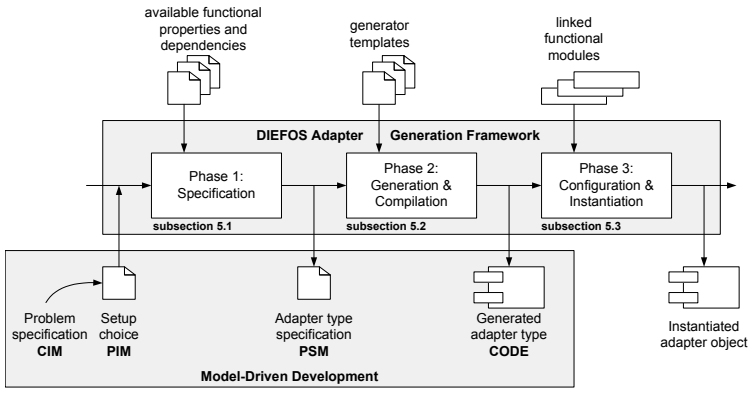

Fig. 2. DIEFOS Generator Framework - Macro-Architecture

3 DIEFOS Approach

The DIEFOS approach (**D**ata **I**ndependence, **E**fficiency and functional flexibility using a **F**eature-**O**riented **S**oftware-development) solves the problems described in Section 2. Basically, this framework—whose macro-architecture is illustrated in Figure 2—comprises the three phases *1: Specification*, *2: Generation & Compilation* and *3: Configuration & Instantiation*.

First, an informal problem specification (CIM) is provided. It is manually transformed into a setup choice (the applicable alternatives are given by feature diagrams similar to Figure 3), which represents the platform-independent model (PIM). This choice—in conjunction with available functional properties and dependencies—is used in order to create the formal adapter type specification (PSM) using an XML model representation. Second, within the generation step, a java class (CODE) is generated from the adapter type specification input, using specific code templates. Finally, this class is compiled and loaded into the JVM. Third, the created instance of the generated adapter as well as the linked functional modules have to be configured. We use an approach where specific functionality can be reused in function modules almost without any overhead. So, during runtime, these hard-coded modules are used as a library.

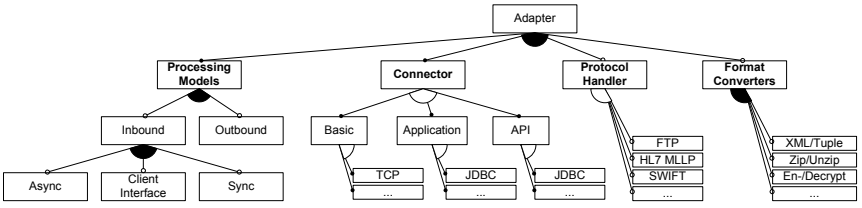

Fig. 3. Adapter Type Specification Feature Diagram

4 Summary and Open Challenges

The overall motivation for this work was the existence of the four pragmatical problems: (1) poor performance, (2) functional restrictions, (3) development effort and (4) need for data independence. The goal was to realize an adapter architecture which ensures data independence with minimal overhead concerning the processing efficiency. Further, the functional flexibility should also be maximized while minimizing the development effort at the same time using model-driven development.

In order to solve the given problems, we first observed the adapter problem characteristic of real-world integration platforms. Second, we proposed the DIEFOS approach, which overcomes the given problems using a model-driven generation approach. This allows for dynamic composition of adapters and comprises the three phases: *1: Specification*, *2: Generation & Compilation* and *3: Configuration & Instantiation*. The goal is to generate adapter types in a feature-oriented manner. The dynamic combinations of format converters, protocol handlers and physical connectors make it possible to ensure the data independence, functional flexibility and even the efficiency can be ensured. However, there are open problems and challenges. Those include but are not limited to: (1) a conceptual adapter specification model, (2) the debugging and testing of generated dynamic adapters, (3) the use of a configuration history for consistent recovery processing, (4) the self-configuration for the generation of adapter specifications based on workflow descriptions and (5) the separation of data and meta data for functional correctness and avoidance of runtime errors. Due to the practical relevance, we want to invite interested research groups and industry vendors to participate in the discussion on this approach and open challenges.

References

1. Kleppe, A., Warmer, J., Bast, W.: MDA Explained. The Model Driven Architecture: Practice and Promise. Addison-Wesley (2003)
2. Amelunxen, C., Königs, A., Rötschke, T., Schürr, A.: Moflon: A standard-compliant metamodeling framework with graph transformations. In Rensink, A., Warmer, J., eds.: Model Driven Architecture - Foundations and Applications. (2006)
3. Dorda, C., Heinkel, U., Mitschang, B.: Improving application integration with model-driven engineering. In: ICITM. (2007)
4. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing adapters for web services integration. In: CAiSE. (2005) 415–429
5. Lee, K., Kim, J., Lee, W., Chong, K.: A tool to generate an adapter for the integration of web services interface. In: CBSE. (2006) 328–335
6. van den Heuvel, W.J., Weigand, H., Hiel, M.: Configurable adapters: the substrate of self-adaptive web services. In: ICEC. (2007) 127–134
7. Brogi, A., Popescu, R.: Automated generation of bpel adapters. In: ICSOC. (2006) 27–39
8. Pieczykolan, J., Kryza, B., Kitowski, J.: Semi-automatic creation of adapters for legacy application migration to integration platform using knowledge. In: International Conference on Computational Science (4). (2006) 252–259
9. Gong, P., Gorton, I., Feng, D.D.: Dynamic adapter generation for data integration middleware. In: SEM. (2005) 9–16
10. Hohpe, G., Woolf, B.: Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions. Addison-Wesley (2004)