A Middleware Architecture for Transactional, Object-Oriented Applications

Dissertation am Fachbereich Mathematik und
Informatik der Freien Universität Berlin

eingereicht von Christoph Hartwich
am 14. November 2003, verteidigt am 17. Dezember 2003

Betreuer:

Prof. Dr. Heinz Schweppe
Dr. Ralf-Detlef Kutsche

Gutachter:

Prof. Dr. Heinz Schweppe
Prof. Dr. Herbert Weber
Prof. Johann-Christoph Freytag, Ph.D.

Abstract

Enterprise applications are transactional, distributed multi-user applications that are employed by organizations to control, support, and execute business processes. Traditionally, data-intensive enterprise applications were built on top of centralized transaction processing monitors. Nowadays, these TP monitors are replaced by object-oriented multi-tier architectures. In this thesis, we focus on two aspects of object-oriented, data-intensive enterprise applications: distribution and data management.

We first define an architectural style for multi-tiered enterprise applications. In addition to the tiered structure, the style focuses on an application's *process topology*, which consists of distributed processes as defined by the operating system, data stores, and client/server communication relationships. Usually, the design of an adequate process topology for a concrete application is the task of a software architect. Typically, a design has to address a broad range of requirements, such as scalability, performance, availability, security, the need to integrate existing subsystems, or organizational and legal requirements. As a guide to the design of process topologies, we present a catalogue of patterns (so called *topology patterns*) that form a pattern language for our architectural style.

Traditionally, distribution of enterprise applications is considered a high-level, architectural concern. Decisions on distribution have to be made at an early design stage of a development project and are expensive to change later on. Moreover, existing enterprise application middleware supports only a small set of standard topologies well. However, we argue that demanding enterprise applications require *custom* and *adaptable* process topologies:

- *Custom* topologies are needed to address application-specific requirements, e.g., those concerning replication, distributed data, caching, or integration of existing subsystems. Using simple standard topologies instead of custom topologies can significantly restrict the capabilities of demanding enterprise applications.
- During the life cycle of an enterprise application, requirements are likely to change. For example, an application has to serve an increasing number of concurrent users or has to provide more fault tolerance. A process topology should be *adaptable*, i.e., easy to adapt to changing requirements without having to re-design the application.

Unfortunately, such topologies are difficult to realize with existing enterprise application middleware. We analyze the difficulties and identify six key requirements for middleware to efficiently support custom and adaptable process topologies. Among other things, it is essential to pairwise decouple topology, application code, and data distribution scheme. Furthermore, we present our *Flexible Process Topology* (FPT) architecture, which realizes all these requirements. The architecture defines principles of a middleware framework that enables custom and adaptable distributed process topologies. The architecture relies on a network of object manager components that collectively provide data management services to application code.

As a proof-of-concept, we present an exemplary implementation of a middleware framework for enterprise applications that is based on the concepts of our FPT architecture. The prototype demonstrates that, in typical cases, the distributed structure of an enterprise application can be defined and adapted through *(re)configuration* – without affecting the implementation. This gives developers more flexibility in constructing and customers more flexibility in deploying their enterprise applications. The framework scales from simple two-tier architectures to large-scale distributed structures with features like an arbitrary number of tiers, distributed data stores, and replication of application processes.

Finally, we evaluate adaptability and performance of our proof-of-concept implementation for several typical scenarios.

The main contributions of this thesis are:

- the definition of an architectural style for multi-tiered enterprise applications, along with a pattern language for designing process topologies,
- the identification of requirements for enterprise application middleware to support custom and adaptable process topologies, and
- a middleware architecture (including an implementation and an evaluation) that explicitly supports custom and adaptable process topologies.

Contents

Acknowledgements

First, I would like to thank Prof. Heinz Schweppe, my first advisor, for his guidance, encouragement, and for supporting my work. Many thanks go to my second advisor, Dr. Ralf-Detlef Kutsche, for his support and many fruitful discussions. I am also grateful to Prof. Herbert Weber and Prof. Johann-Christoph Freytag for their valuable comments and for reviewing my dissertation. Special thanks go to all members of the Databases and Information Systems group of the Freie Universität Berlin and to the members of the Graduate School in Distributed Information Systems. I will never forget the great time we had together.

Last but not least, I would like to thank my family, in particular my wife Oxana, for everything.

This research was supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant GRK 316).