Support for Mobility and Replication in the AspectIX Architecture

M. Geier, M. Steckermeier,
U. Becker, F.J. Hauck, M. Meier, U. Rastofer

September 1998 TR-I4-98-05

Technical Report

Computer
Science Department

Operating Systems — IMMD IV

Friedrich-Alexander-University
Erlangen-Nürnberg, Germany

This paper was submitted as position paper to the

III Workshop on Mobility and Replication,
European Conference on Object-Oriented Programming,
Brussels - Belgium, July 20, 1998

Support for Mobility and Replication in the *AspectIX* Architecture

Martin Geier, Martin Steckermeier, Ulrich Becker, Franz J. Hauck, Erich Meier, Uwe Rastofer

{geier, mstecker, ubecker, hauck, meier, rastofer}@informatik.uni-erlangen.de
IMMD IV, University of Erlangen-Nürnberg
Martensstr. 1, D-91058 Erlangen, Germany
Tel.: +49.9131.85.8029, Fax: +49.9131.85.8732

Abstract: CORBA as a standardized object-based middleware for distributed computing still lacks sufficient support for mobility and replication, although there are several proposals to integrate these mechanisms. *AspectIX* is a more flexible and more open architecture than CORBA, but *AspectIX* is still fully CORBA compliant.

Unlike CORBA with its static client-server relationship, *AspectIX* uses the concept of distributed objects. Each distributed object is represented by at least one local part, called fragment, that communicates with other fragments to synthesize the desired behaviour. The fragment implementation that is actually used depends on nonfunctional aspects which are specified on the distributed object via a typed interface.

Based on this model *AspectIX* provides a single mechanism that is especially suited to realize both: mobility and replication.

1 Introduction

CORBA as a standardized object-based middleware for distributed computing has gained high interest over the past few years. Nevertheless, the standard and especially current implementations still lack sufficient support for replication and mobility which are of crucial importance in large distributed systems. *AspectIX* is a more flexible and more open architecture than CORBA, but still remains fully CORBA compliant. Therefore *AspectIX* applications can freely interact with other CORBA applications using standard protocols like GIOP.

Unlike CORBA with its static client-server relationship, *AspectIX* uses the concept of distributed objects. Each distributed object is represented by at least one local part, called fragment, that communicates with other fragments to synthesize the desired behaviour. This local fragment can carry more semantics than a simple CORBA stub and can be replaced at runtime by another fragment to fulfill the application's requirements, possibly based on information about the current environment. The fragment implementation that is actually used depends on nonfunctional aspects which are specified at the distributed object and configured via a typed interface.

This position paper will show how *AspectIX* can support mobility and replication using distributed objects and a configuration interface for nonfunctional aspects. The paper is organized as follows: Section 2 describes the *AspectIX* architecture. Section 3 illustrates how the architecture can be used to realize replication and mobility. Section 4 will give our conclusion.

2 The *AspectIX* Architecture

From the outside, an *AspectIX* implementation looks like a standard CORBA implementation [OMG98]. There are location transparent names for objects (in CORBA: Interoperable Object References, IORs), which are converted to a local object referring to the distributed object. In CORBA the local object is a stub that delegates invocations to a so called server object.

Unlike CORBA, the *AspectIX* architecture adopts a fragmented object model similar to *Fragmented Objects* from INRIA [MGN+94] and *Globe* from the Vrije Universiteit Amsterdam [SHT97]. A distributed object consists of several so called fragments, which can interact (see Fig. 2.1). A client of the object always has at least one of these fragments in its local address space.

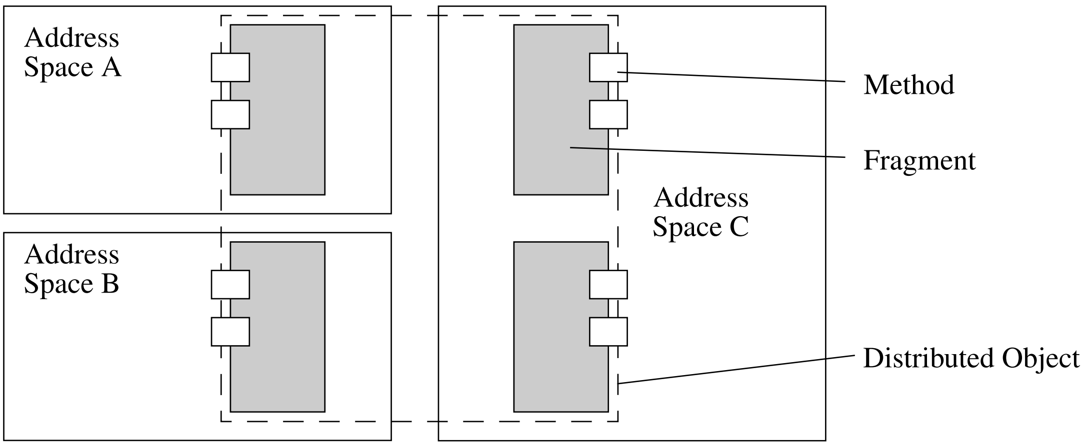

Fig. 2.1 A Distributed Object in Three Adress Spaces

A fragment could be a simple stub (as in CORBA), which is created on the client side. The stub may connect to another fragment (in CORBA: the server object) that holds the object's functionality. On the other hand, fragments at the client side can be more intelligent. An intelligent fragment may hide the replication of the distributed-object's state, it may realize real-time constraints on the communication channel to another fragment, it may cache some of the object's data, and it may locally implement some of the object's functionality, just to name a few possibilities.

AspectIX extends CORBA in the sense that we adopt CORBA's IDL for multiple language support and CORBA's ORB and object interfaces for CORBA compliance. Thus, the local fragment of a distributed object provides an interface described in CORBA IDL. When a fragment is created, e.g. as a result parameter of a method invocation, the ORB creates two local objects in the desired target language: a fragment interface and a fragment implementation (see Fig. 2.2).

The fragment interface is a generic object that is automatically generated during the development process. It only depends on the IDL description of the distributed object's interface and its main purpose is to delegate method calls to the fragment implementation. In the simplest case,

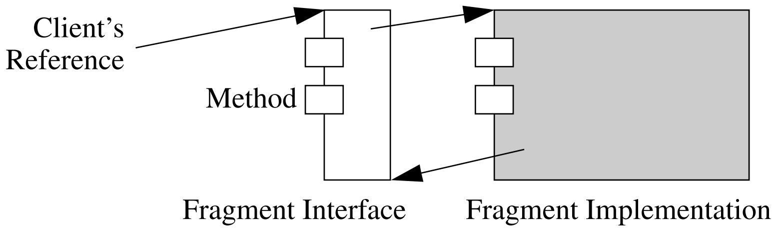

Fig. 2.2 Components of a Fragment

the implementation does nothing more than a remote method invocation, thus the combination of interface and implementation object realizes the same semantics as a traditional CORBA stub.

The separation into fragment implementation and fragment interface allows to exchange the fragment implementation without loosing the validity of the client's reference. Moreover, there can be different interface objects sharing the same implementation object similar to [DM92], which allows to distinguish between internal interfaces needed for communication between fragments and external interfaces offered to the client application.

A distributed object consists of at least one fragment, but can be extended by additional fragments at runtime, which not necessarily have to be in distinct address spaces (see Fig. 2.1). As far as needed, communication between these fragments is done via standard *AspectIX* communication mechanisms.

Nonfunctional aspects (similar to [KLM+97]) can be specified via a typed configuration interface. This specification determines which local fragment implementation represents the distributed object. When the aspect specification changes at runtime and cannot be satisfied by the currently active implementation, it is transparently exchanged by another one. As there can be more than one fragment of the same distribute object in one address space, it is also possible to have different aspect specifications coexisting at the same time.

3 Support for Mobility and Replication

Realizing mobility and replication with this architecture is straight forward by extending and shrinking the distributed object.

In case of replication the distributed object (Fig. 3.1 ①) is simply extended by an additional fragment which just acts as a replica (Fig. 3.1 ②). Nevertheless this replica represents still the same distributed object, i.e. it is transparent to the client application (e.g. Address Space B in Fig. 3.1) whether it accesses a remote object (Address Space A in Fig. 3.1) or a local replica. It is the task of the replica to implement the specified consistency model communicating with the other fragments using standard *AspectIX* communication mechanisms.

For mobility, we first use the same mechanism as for replication, i.e., we extend the distributed object with a new fragment at the destination site. After transferring the whole state from the original fragment to the new one, the old fragment can be replaced by a simple stub acting as a forwarding entity (Fig. 3.1 ③). If no further communication to the distributed object is required from the original site, the fragment on this site can be deleted; this results in a migration of the distributed object (Fig. 3.1 ④). To the client application, this migration is atomic as it only sees

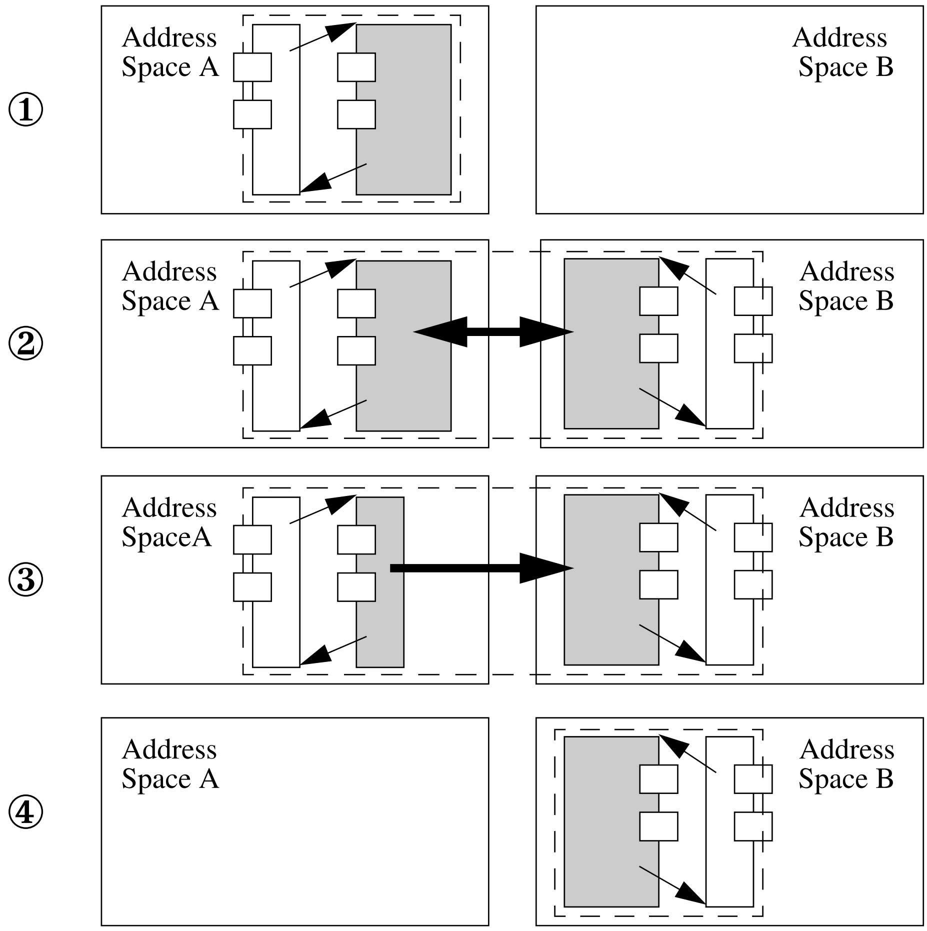

Fig. 3.1 Realizing Replication and Mobility in *AspectIX*

the distributed object not its interior fragments that might be in an intermediate state.

Neither the decision whether an object has to be replicated or migrated, nor the choice of the implementation necessary to realize the desired behaviour have to be done by the client application. Instead the client specifies properties e.g. the required consistency level in form of so called *Aspects* [KLM+97]. It is the task of the distributed object to decide if the current implementation can satisfy these properties or whether it has to be replaced, e.g., to replace a simple stub by a local replica. It also chooses the implementation that can fulfill the requirements best, based on information of the distribution of objects, on the load on different sites, or on costs and performance of available network connections.

4 Conclusion

This paper describes the basic ideas of *AspectIX*, a new middleware architecture that extends CORBA, although remaining fully compliant. By using a model of distributed objects consisting of several fragments instead of a client-server model, *AspectIX* is capable of transparently integrating mechanisms like replication and mobility. Nonfunctional properties like replication strategies, consistency models, etc. can be specified in the application via aspects and can result in dynamic and transparent exchanges of implementations.

5 References

KLM+97 G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, J. Irwin: *Aspect-oriented programming*. Techn. Report SPL97-008 P9710042, Xerox Palo Alto Reserach Center, Feb. 1997.

DM92 P. Dickman, Mesaac Makpangou: *A Refinement of the Fragmented Object Model.* 3rd Int. Workshop on Object-Orientation in Operating Systems, Dourdan, France, Sept. 1992

MGN+94 M. Makpangou, Y. Gourhant, J.-P. Le Narzul, M. Shapiro: “Fragmented Objects for distributed abstractions”. In: T. L. Casavant and M. Singhal (eds.), *Readings in Distributed Computing Systems*, IEEE Computer Society Press, 1994, pp. 170–186.

OMG98 Object Management Group: *The Common Object Request Broker Architecture*, Version 2.2. February, 1998.

SHT97 M. van Steen, P. Homburg, and A.S. Tanenbaum. “The architectural design of Globe: a wide-area distributed system.” *Technical Report IR-422*, Vrije Universiteit Amsterdam, March 1997.