Interoperability of distributed systems - a gateway solution *

Andrzej Uszok
Institute of Computer Science
University of Mining and Metallurgy
Al. Mickiewicza 30, Kraków, Poland
e-mail: {uszok}@ics.agh.edu.pl

November 1994

Abstract

Cooperation of various distributed systems is today very desirable. Many solutions to this problem have been proposed. One of them is a gateway. It is an interoperability mechanism very well suited to object-oriented systems, but it has already proved its usefulness for other purposes. This paper presents concept of this interoperability mechanism, its diverse possible realizations and ways of usage. Drawing on the experience with implementing and using gateway in our Interoperability Architecture, we try to address several important questions concerning the features of this solution.

1 Introduction to Interoperability

With growing number of distributed computer systems and information resources, there is an increasing need for technology to integrate them creating an integrated distributed computing system. Global interoperability may be achieved by the universal adoption of a particular standard, however it is not feasible because of various technological assumptions and functional requirements. Hence, the best achievable solution is to create mappings between the features of the various models [11, 6] and provide interoperability mechanisms for different computing paradigms (imperative, functional or logic) and environments.

The interoperability of systems has to be achieved on various levels. It is related to:

- physical representation of exchanged data,
- protocol of network communication,
- protocol of invocation dispatching,
- specification of interfaces including offered services,
- naming of cooperating components,
- preservation of services transparencies.

An introduction of some special mechanisms or services layering above distributed environments and below user-specific applications is required to cope with these problems. Many variants of these mechanisms have been proposed and still are proposed [5, 6, 8], which in different way try to accomplish requirements of cooperating systems.

*This work was sponsored by Polish Research Council (KBN), under grant no.: 8 S503 015 06.

2 Idea of Gateway

A gateway is an intermediate software module introduced between cooperating components to mediate between them and to achieve their interoperability. It enables interaction between them by translation and dispatching their invocation which cross domains boundaries. However, its activity should be seen as not limited to invocations dispatching, but may be extended in preservation of services' transparency. For example, a gateway can match, maintain and negotiate the interfaces that application's components expect from others. It also insulates components from specific features and changes being made to others e.g. legacy systems may be combining in modern software.

Gateways can work as *smart agents* which track components in the domains. Cooperating with each other, they can support trading of components' services, localizing them, binding them together and reconfiguring a whole system. They could pass authentication information between components and ensure security of exchanging messages, as they work in open, multiusers environments exposed to many threats. They could actively participate in management process, enable implementation of various management policies. A fault-tolerance of their work can be guaranteed by introduction of a replication and a group communication mechanism (like in ISIS).

A cooperation of various distributed system via wide area networks is gaining wide importance and interest. However, the usage of WAN introduces a lot of problems which do not exist or which are not so distinct in local networks. Gateway can cope with:

- a wide variety of communication protocols – translating these different protocols one to another or to one universal inter-gateway protocol,
- network fail or partition – insulting, looking for alternative connection or rising appropriate exception,
- significant messages delay, limited bandwidth – introducing cashing, filtering and piggybacking.

It seems that a *gateway* interoperability solution is very well suited to construction of a complex distributed applications in a heterogeneous environment.

3 Gateway application

Gateway is especially appropriate for object-oriented systems. For instance it has been proposed as an interoperability mechanism for CORBA systems [8] because of its flexibility and possibility of independent diverse implementations. It is already widely used as a solution for retrieving data from a set of heterogenouse databases [6]. Variants of gateway – *mediator* has been used [9] in software engineering to compose and integrate application from components, increasing their independence. Also gateway –*interceptor*, between HTTP and Orbix is currently implemented by APM Ltd. Cambridge.

4 Design analysis of gateway construction

The main activity of gateway is very well defined; it is translation and dispatching of invocations between components. It justifies an idea of designing gateway as an object-oriented framework [3] i.e. a module containing the basic and necessary gateway functionality. Such a framework may be refined to meet the particular features and needs of given interworking environments. For instance, for particular systems, the gateway's communication module can be specialized for their protocols, or the gateway can be enhanced with authentication mechanism, etc. However, the main structure of the gateway will remain unchanged, as most of the software components will be reused. Such an universal architecture of this module may generally be used in heterogeneous distributed environments. The versatile architecture and requirements of cooperating systems force the open construction of the gateway framework; users should be able to add new services and/or new activities when desired.

There are a lot of technical problems in gateway's basic activity and internal architecture, which should be addressed during development of its framework:

- bootstrapping and maintaining persistence of its state,
- cashing of components references,
- garbage collection of gateway state and proxy objects,
- cooperation of many gateways,
- interfaces matching and their negotiation,
- consistency management of its local data bases (Interface Repositories in CORBA-complaint systems),
- translation of components identities and name service,
- preserving security of forwarded information.

These problems have to be solved first in order to provide efficient implementation of gateway, as performance is a critical factor in the usefulness of systems.

In next sections we will present our current experience with gateway's design and construction. They serve us as a testbed for determination and investigation of problems enumerated above.

5 Implementation of a gateway in object-oriented distributed systems

In several recent papers [12, 10, 4] we have proposed and described implementation of *Interoperability Architecture* (IA), built around a family of distributed languages and software tools integrated by the specialized component called *Interoperability Server* (IS). The ultimate goal of this architecture is a preservation of original computational models of linked environment. For this purpose stream-like communication mechanism with other IA's environments was developed for declarative systems (Strand, PCN) and Linda [4], and an idea of gateway was applied for object-oriented systems (ANSA [1], Orbix [7]).

A gateway for object-oriented distributed systems like ANSA or CORBA is implemented as a special object which represents IA in the environment. This object looks like an ordinary one for all objects and mechanisms of the environment. It registers services provided by other IA system in the environment's repository and exports from it services needed by other environments. On the other hand it provides mechanisms for communication with IS. Its main functionality is to translate messages from the internal environment representation to the IS protocol.

This module should be built separately for each new application because it has to contain appropriate *stub* parts generated according to specifications of methods exported and imported by it (interface determined gateway). However in some systems (CORBA-compliant) it is possible to built this module

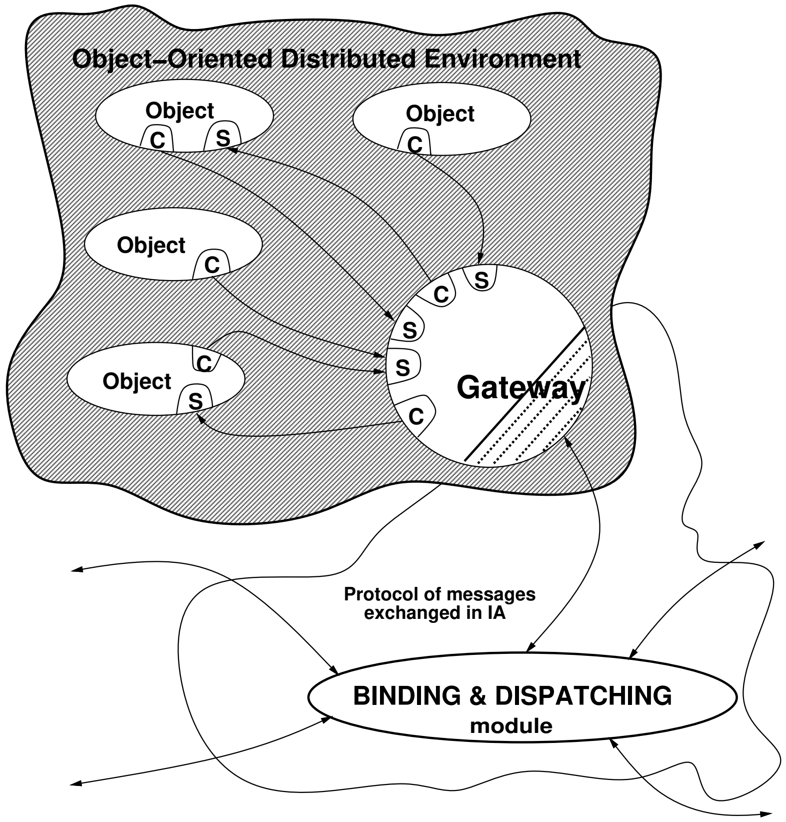

Figure 1: *Gateway* module in Object-Oriented Distributed Environment

in the universal way using DII (Dynamic Invocation Interface) and DSI (Dynamic Skeleton Interface – now discussed extension to CORBA [8]). These mechanisms enable gateway to be a server for any method and call any method from other objects. The generic gateway built in such a way possesses a reflection features, its methods and functionalities can change or be configured in a free way.

From the other hand gateway is an ordinary object of the given environment. Thus, no program executing in this environment (except for gateway) is aware of the fact that services offered by it are defined in other languages and systems. The original computational model of the environment has been left unchanged.

6 Summary

In our opinion, the value of the sum of several different distributed environments can be dramatically increased by providing them with constructs enabling their cooperation without the loss of their original computational model and services' transparency. The idea of a gateway addresses this challenge. In this paper we survey current state of this idea and discuss its possible development in near future. We present our experience with this mechanism, which is extensively used during construction of our Interoperability Architecture. Concentrating on aspects of gateway design related to object-oriented distributed environments we can also see possibilities of applying it to other systems. We claim that the process of creating gateway should be more automatic and its configuration and autonomous behavior has to be further investigated.

References

[1] *ANSAware 4.0 — Application Programmer's Manual*, APM Ltd. Cambridge (1992)

[2] *Draft Common Object Request Broker Architecture Revision 1.1*, OMG Report 91-12-1, OMG Inc. (1991)

[3] R. H. Campbell, N. Islam, R. Johnson, P. Kougiouris, P. Madany *Choices, Frameworks and Refinement*, Proceedings of IWOOS '91, Xerox PARC (USA) (1991)

[4] G. Czajkowski, A. Uszok, K. Zielinski *Distributed Declarative Systems as Parts of Cooperating Software Environments*, Proceedings of ICLP'94 Post-Conference Workshop 'Integration of Declarative Paradigms', Santa Margerita Italy, TR of Max-Planck Institute für Informatik (1994)

[5] D. Konstantas *Object Oriented Interoperability*, Proceedings of ECOOP'93, Kaiserslautern Germany (1993)

[6] F. Nayeri, B. Hurwitz, *Dispatching Experiments in a Distributed System*, TR of GTE Laboratories Incorporated (1993)

[7] *The Orbix Architecture*, IONA Technologies Ltd. (1993)

[8] *ORB Interoperability*, ORB 2 RFP Submission, IONA Technologies, SunSoft Inc. (1994)

[9] K. Sullivan, D. Notkin *Reconciling Environment Integration and Software Evolution*, TR of the University of Washington 91-08-08 (1991)

[10] A. Uszok, G. Czajkowski, K. Zielinski *Interoperability Gateway Construction for Object-Oriented Distributed Systems*, Proceedings of Nordic Workshop of Programming Environment Research NWPER'94, Lund Sweden, TR of the Lund University (1994)

[11] J. C. Wileden, A. L. Wolf, W. R. Rosenblatt, P. Tarr, *Specification Level Interoperability*, Com. of ACM, May vol. 34 (1991)

[12] K. Zieliński, G. Czajkowski, A. Uszok *Coordination through Interoperability in Distributed Processing*, Presented at ECOOP'94 Workshop 'Models and Languages for Coordination of Parallelism and Distribution', Bologna Italy (1994)