

# PolyORB

## a schizophrenic middleware

**Interoperability**  
 Dynamic gateway  
 Dynamic implementation/invocation  
 Simultaneous interacting personalities

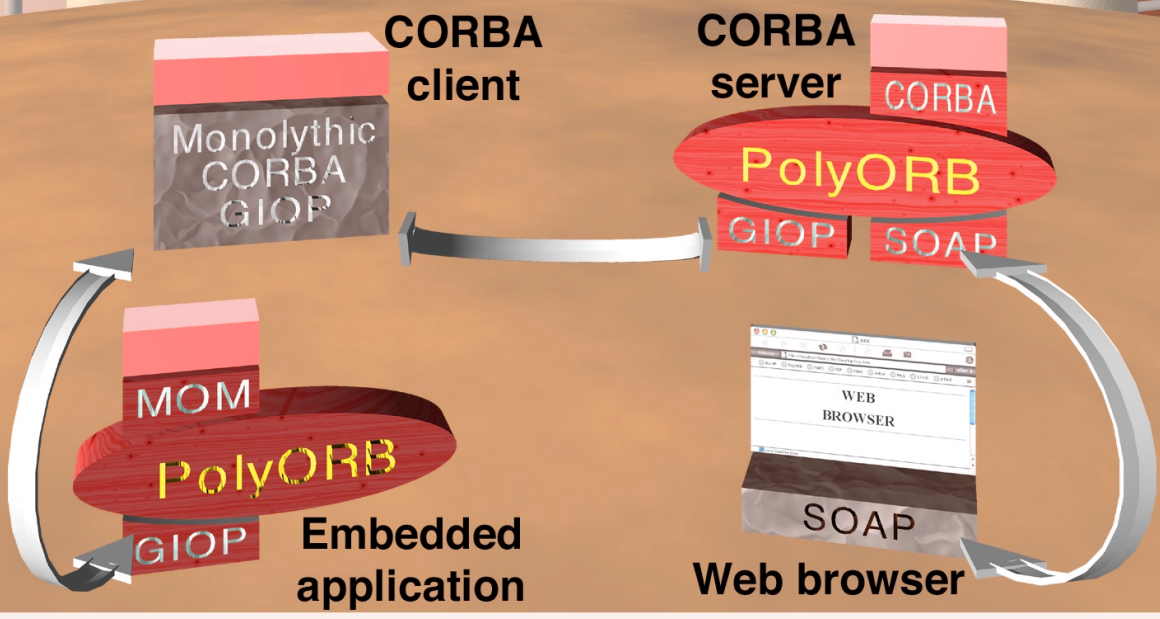
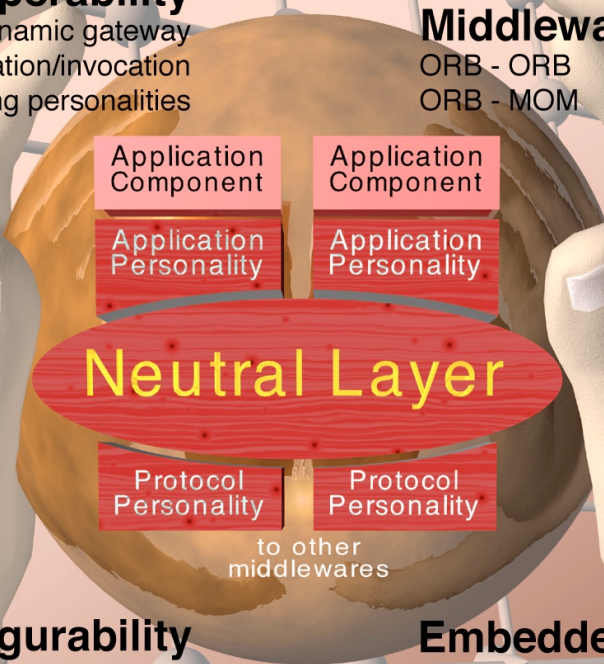
**Middleware to Middleware**  
 ORB - ORB  
 ORB - MOM

**Genericity**  
 Middleware functions  
 instantiated as  
 personalities

**Software Engineering**  
 Rapid prototyping  
 Intensive code reuse

**Configurability**  
 Design patterns  
 Framework of components

**Embedded & Real Time**  
 Smaller footprint, determinism  
 Linked with Real Time kernel



<http://libre.act-europe.fr/polyorb>  
 L. Pautet F. Kordon  
 Telecom Paris LIP6-SRC



# PolyORB Generic Middleware

<http://libre.act-europe.fr/polyorb>

## Current Release

**PolyORB 1.0p** (released 2003-06-16) This is the latest release of PolyORB. This release contains a CORBA compliant instantiation of the generic middleware PolyORB. It includes:

- an IDL to Ada95 compiler,
- Portable Object Adapter (POA), Dynamic Skeleton Interface (DSI), Dynamic Interface Invocation (DII) and Interface Repository (IR) implementations,
- COS Naming, COS Event and COS Time services implementations,
- GIOP 1.0 to 1.2 implementations.

This CORBA implementation can be configured for a full tasking, Ravenscar tasking or no tasking runtime.

This release should be considered as a **stable implementation** of a CORBA middleware over PolyORB.

Other instantiations of PolyORB are available in the CVS public tree: Distributed System Annex of Ada 95, SOAP and MOMA, the Message Oriented Middleware for Ada. They will be added in future releases.

## Introduction to PolyORB

### Distributed applications and middleware

**PolyORB** aims at providing a uniform solution to build distributed applications; relying either on industrial-strength middleware standards such as CORBA, the Distributed System Annex of Ada 95, distribution programming paradigms such as Web Services, Message Oriented Middleware (MOM), or to implement application-specific middleware.

Basically, middleware are framework that hide the complex issues of distribution and provide the programmer with high-level abstractions that enable easy and transparent construction of distributed applications. A number of different standards exist for creating object-oriented distributed applications. These standards offer two subsystems that enable interaction between application partitions:

- the API seen by the developer's applicative objects;
- the protocol used by the middleware environment to interact with other nodes in the distributed application.

Besides, middleware provide implementation guidelines as well as development tools to ease the construction of large heterogeneous distributed systems. Yet, many issues typical to distributed programming may arise: application architectural choice, configuration or deployment. Since there is no "one size fits all" architecture, choosing the adequate distribution middleware in its most appropriate configuration is a key design point that dramatically impact the design and performance of an application.

Consequently, applications need to rapidly tailor middleware to the specific distribution model they require. A distribution model is defined by the combination of distribution mechanisms made available to the application. Common examples of such mechanisms are Remote Procedure Call (RPC), Distributed Objects or Message Passing. A distribution infrastructure or middleware refers to software that supports one (or several) distribution model, e.g.: OMG CORBA, Java Remote Method Invocation (RMI), the Distributed System Annex of Ada 95, Java Message Service (MOM).

### PolyORB : a generic middleware with an instance per distribution model

Typical middleware implementations for one platform supports only one set of such interfaces, pre-defined configuration capabilities and cannot interoperate with other platforms. In addition to traditional middleware implementations, **PolyORB** proposes an original architecture to enable support for multiple interoperating distribution models in a uniform canvas.

**PolyORB** is a polymorphic, reusable infrastructure for building or prototyping new middleware adapted to specific application needs. It provides a set of components on top of which various instances can be elaborated. These instances (or personalities) are views on **PolyORB** facilities that are compliant to existing standards, either at the API level (application personality) or at the protocol level (protocol personality). These personalities are mutually exclusive views of the same architecture.

The decoupling of application and protocol personalities, and the support for multiple simultaneous personalities within the same running middleware are key features required for the construction of interoperable distributed applications. This allows PolyORB to communicate with middleware that implement different distribution standards: PolyORB provides *middleware-to-middleware* interoperability (M2M).

Besides, **PolyORB**'s modularity allows for the easy extension or modification of its core components and personalities to meet specific requirements. Thus standard compliant or specific personalities can be created in a unique process, from early stage prototyping to full featured implementation. The PolyORB architecture also permits the automatic, just-in-time creation of proxies between incompatible environments.

**PolyORB** currently supports the following personalities (in the main development branch, available through CVS access):

- application personalities: CORBA, Distributed System Annex of Ada 95 (DSA), MOMA - MOM for Ada. Interaction between CORBA and DSA partitions has been successfully tested;
- protocol personalities: GIOP (CORBA protocol layer), SOAP.
- under development: a Web Services personality, based on the AWS API.