Editor: Christof Ebert ■ Alcatel ■ christof.ebert@alcatel.com

MICO: An Open Source CORBA Implementation

Arno Puder

Welcome to a new column on open source software. As more people try to incorporate OSS into their products, having a list of potential products and some meaningful comparison among them seems useful. This column promotes sharing of specific OSS experiences from a user perspective. Our aim isn't to drag people away from proprietary (closed) software but rather to gather and disseminate users' feedback on the many available OSS components and tools.

This inaugural column is on MICO, an OSS implementation of CORBA. Despite the growing availability of other middleware, CORBA is still broadly used and evolving, especially for heterogeneous environments.

I look forward to hearing from you, both readers and prospective column authors, about this column and the products and tools you want to know more about. If you'd like to write for this column, see the sidebar of author guidelines. —*Christof Ebert*

The Common Object Request Broker Architecture is a specification for creating, distributing, and managing distributed program objects across a network. Both the International Organization for Standardization and X/Open have sanctioned CORBA as the standard middleware architecture for distributed objects. CORBA was specifically designed to support heterogeneous environments, different vendors' products, and several popular programming languages. Numerous implementations of the CORBA specification exist today, both in the commercial and the open source domain.

MICO (www.mico.org) is one such implementation; I and several other developers are its cofounders. In this column, I'll discuss MICO's internal architecture and then offer a few guidelines to help you choose the right CORBA implementation for your purposes.

MICO overview

Inspired by the GNU project, the name "MICO" stands for "*MICO Is* CORBA." Seven years after the first public release, MICO has evolved into a mature open source project, with close to a half million lines of source code contributed by more than 150 programmers. In 1999, an international, technology-neutral vendor consortium called the Open Group offered MICO a free license of its CORBA test suite. MICO passed several thousand test cases, so the Open Group awarded it its CORBA compliance brand. Today MICO is used in both academia and industry. The largest known commercial deployment is with the Weather Channel, which broadcasts weather forecasts in the US and uses MICO to collect and distribute weather data throughout North America.

Implementing CORBA, a voluminous specification occupying thousands of pages, in MICO required us to make several important design decisions. We based MICO's internal design on a microkernel approach. Its Object Request Broker is simple: the ORB can only connect objects in the same address space. We moved all extra functionality, such as the Internet Inter-ORB Protocol (IIOP) and the Portable Object Adapter (POA), into special-purpose adapters outside the ORB. A microkernel approach yields a modular, extensible architecture, an especially important feature for programmers who want to contribute. MICO's ORB is particularly extensible: new functionality can be plugged into it without affecting the rest of the system, even at runtime. This approach has been successfully used for pluggable transport mechanisms, in which different transport layers can be added to MICO. As proof of concept, MICO supports UDP (User Datagram Protocol).

The IDL (Interface Definition Language) compiler's implementation follows similar design principles in terms of modular and extensible architecture. In MICO, the IDL compiler and the Interface Repository (IR) are closely related: the IR stores the abstract syntax tree that the IDL compiler's front end creates. By doing so, MICO achieves maximum reuse of its own components. The fact that an IDL specification described the IR's interface posed an interesting engineering problem: the IDL compiler depends on the IR, and vice versa. The latter dependency occurs because the IR is a regular CORBA object and thus requires stubs and skeletons that the IDL compiler normally creates. We can solve this chicken-and-egg problem via a bootstrap process. As a result, MICO's source code contains code that was automatically generated during the bootstrap process. Another consequence of this architecture is that the stubs and skeletons generated by the IDL compiler are the same for all platforms. So, this generated code must be generic enough to be compiled by all C++ compilers.

MICO is completely written in C++. Initially, MICO only supported the GNU C++ compiler. As MICO's user base grew, demand for different compilers began to surface. Over time, we ported MICO's source code to various C++ compilers—a tedious task. We were surprised at how different various C++ compilers can be. Internally, MICO makes extensive use of the Standard Template Library, whose many useful data structures facilitate the implementation of C++ programs. The STL's downside is that it's built using C++ templates, and only a few C++ compilers offer good support for templates. Some C++ compilers and linkers can't remove duplicate template instantiations, resulting in huge binaries. Although this isn't MICO's fault, it sometimes has been called bloated because of this.

Users often ask why MICO doesn't support Java. Most people don't realize that supporting another language basically means reimplementing the complete ORB. Just adding a new back end to the IDL compiler isn't sufficient; the ORB library also must be ported to the other language. Our response to such requests is that MICO supports C++ well and that Java users can find other open source implementations such as JacORB. Because interoperability is no longer a vision but a reality (perhaps not for Web Services but definitely for the CORBA domain), IIOP seamlessly connects components written in different programming languages.

Choosing a CORBA implementation

Managers must often choose a specific CORBA implementation for their project, but guidelines for doing so aren't clear-cut. They might have to choose among dozens of CORBA implementations, with different strengths and weaknesses. Yet, asking a few basic questions can help narrow the field.

- Which programming language are we

Table 1

Product feature matrix of CORBA implementations

Product	Open source	C++	Java	SOAP	Interface repository	Minimum CORBA	Asynchronous messaging interface	CORBA Component Model	Security	Notification service
Visibroker (www.borland.com/visibroker)	No	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes
JacORB (www.jacorb.org)	Yes	No	Yes	No	Yes	No	Yes	No	Planned	Yes
JDK 1.4 (http://java.sun.com/j2se)	No	No	Yes	Planned	Planned	No	No	No	No	No
MICO (www.mico.org)	Yes	Yes	No	No	Yes	Yes	Planned	Yes	Yes	Planned
Orbix 2000 (www.iona.com)	No	Yes	Yes	Yes	Yes	No	Yes	No	No	Yes
Tao (www.theaceorb.com)	Yes	Yes	Planned	No	Yes	Yes	Yes	Planned	Yes	Yes

Author Guidelines for Future Columns

Select one open source software component, tool, product, or product group that you've used. Focus on why you chose it, what its benefits are, how it contrasts with competing components (OSS or commercial off-the-shelf), and what lessons learned might aid other practitioners. For your column to be useful, your report must come from in-depth experience rather than a superficial, one-weekend evaluation.

You may review a single tool or product (such as a GNU or Eclipse item) or a vertical product group (such as defect-tracking tools, middleware, or workflow management tools).

Evaluate the OSS components in a concrete, fairly broad application context and present the information in a neutral style. If authors follow a similar style, readers will have easier access to the information.

Preferably, you are an OSS user rather than a primary author or key contributor. Typically, authors don't work for an independent software vendor or packaging company. You can't be zealous nor hostile toward OSS, as this would bias the evaluation and reduce credibility.

Present feature comparisons of different OSS products and other data in chart format, if possible.

Send your column proposal and author qualifications to Christof Ebert at christof.ebert@alcatel.com.

using? Some CORBA implementations support only certain languages and therefore aren't suitable for projects that require some other programming language.

- Which compiler are we using? The quality of compilers (especially C++ compilers) varies greatly. So, be sure that the CORBA implementation supports the compiler you want to use.
- Which operating system are we using? Different operating systems offer significantly different features. Be sure that the CORBA implementation supports the operating system you want to use. This is particularly true for embedded systems.
- What features do we need? Different projects have different requirements. Not all CORBA implementations support such features as the CORBA Component Model, Objects by Value, or multithreading.
- What license is acceptable? Open source CORBA implementations are released under different licenses that might affect a project. For example, all modifications to an ORB under the General Public License (a "GPLed ORB") must be published; for BSD-style licenses, this isn't necessary.

I maintain a CORBA product matrix on the Web (www.puder.org/corba/matrix) that lists several dozen commercial and open source implementations for different languages and platforms. The matrix provides a quick overview of which features are supported by or planned (by vendors) for each CORBA implementation. Table 1, an excerpt from the Web page, notes the key differentiators of some popular CORBA implementations. I hope these charts, here and online, help you choose the right implementation for your project.

Arno Puder is an assistant professor of computer science at San Francisco State University. Contact him at arno@puder.org.

IEEE Software

How to Reach Us

Writers
For detailed information on submitting articles, write for our Editorial Guidelines (software@computer.org) or access www.computer.org/software/author.htm.

Letters to the Editor
Send letters to

Editor, *IEEE Software*
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
software@computer.org

Please provide an email address or daytime phone number with your letter.

On the Web
Access www.computer.org/software for information about *IEEE Software.*

Subscribe
Visit www.computer.org/subscribe.

Subscription Change of Address
Send change-of-address requests for magazine subscriptions to address.change@ieee.org. Be sure to specify *IEEE Software.*

Membership Change of Address
Send change-of-address requests for IEEE and Computer Society membership to member.services@ieee.org.

Missing or Damaged Copies
If you are missing an issue or you received a damaged copy, contact help@computer.org.

Reprints of Articles
For price information or to order reprints, send email to software@computer.org or fax +1 714 821 4010.

Reprint Permission
To obtain permission to reprint an article, contact the Intellectual Property Rights Office at copyrights@ieee.org.