Towards an OMG IDL Mapping for Common Lisp

Tom Mowbray
Kendall White
The MITRE Corporation
Workstation System Engineering Center
7525 Colshire Drive, MS Z253
McLean, VA 22102
(703)883-6000
FAX: (703)883-3315
Internet: mowbray@mitre.org, klwhite@mitre.org

Abstract

The Object Management Group (OMG*) is adopting a set of standards for distributed computing, including multiple programming language bindings to the Interface Definition Language (OMG IDL). This paper explains the need for an OMG IDL mapping to Common Lisp, and shows some examples of such a binding and its implementation, based upon the authors' work on integrating Common Lisp applications using OMG-compliant object request brokers.

Introduction

Several significant trends in computing technology are complicating the task of Common Lisp programming. Among these trends are the need to maximize utilization of distributed resources, heterogeneous computing, and software interoperability. These are key issues that the OMG is addressing in its standards, and this paper proposes a straightforward way for Common Lisp to benefit from OMG initiatives.

Distributed Processing is increasingly important, given that LAN networks are pervasive and WAN networks, such as the Internet and the future National Information Infrastructure, have increasing visibility and importance. Technologies for distributed processing (such as RPC and middleware) are difficult and expensive to use, particularly when accessed through foreign language bindings indirectly from Common Lisp.

Heterogeneous Computing is commonplace in application environments, such as end-user systems with multiple operating systems and platforms. Common Lisp has an important role in heterogeneous environments for certain types of applications, and there is a need for better mechanisms to integrate Common Lisp seamlessly. Multiple programming languages are often required for applications involving Common Lisp. For Common Lisp developers, heterogeneity is an important challenge that is supported selectively by existing technologies.

Software Interoperability between applications is a capability increasingly demanded by end-users. Application environments such as DDE/OLE, MacApp, OpenDOC, NeXTSTEP/OPENSTEP, and Taligent have shown the potential benefits and feasibility of application frameworks for software interoperability. Although the advantages of Common Lisp are desirable for certain applications, its position in the marketplace does not demand high priority support from most of the major innovators of these application development technologies.

Object Management Group

The OMG is addressing these issues through a consensus standards process, that has enlisted the participation and support of virtually all the major software and hardware platform vendors (Apple, DEC, HP, IBM, ICL, Microsoft, Sun, etc.). The OMG has identified as its mission, the need for consistent access to services as explained in their architecture guide [OMAG]:

> *What is missing is only a set of standard interfaces for interoperable software components. This is the mission of the OMG.*

OMG uses two consensus processes for adopting specifications. The established process uses a request for proposal (RFP) that solicits specifications from multiple submitters who can merge their proposals to obtain consensus. This process can be completed in about one year. A new fast track process is based upon a single submission and a request for comment (RFC) issuance. A fast track adoption could be completed within 6 months. This compares to the 3 or more years in a typical international standards process. Our recommendations for the OMG adoption of a Common Lisp mapping is presented in the conclusions section below.

CORBA

The OMG has produced a standard called the Common Object Request Broker Architecture (CORBA) specification as its model for a distributed communication infrastructure. [CORBA] CORBA is a standard for object request brokers (ORBs) that resolve distributed processing and heterogeneity issues transparently. CORBA-compliant ORB's hide the distributed nature of the computing environment from

---

* The following words and phrases are registered trademarks of the Object Management Group Inc.: OMG, Object Management, ORB, Object Request Broker, OMG IDL, and CORBA.

the programmer, such that requests are handled consistently whether library calls, local calls, or remote invocations. The ORB is responsible for transparently resolving differences between operating systems and hardware representations when software communicates across boundaries. ORBs can automatically activate servers without client intervention. They also handle requests reliably with an extensible exception handling mechanism.

There are many productized implementations of CORBA (from DEC, HP, IBM, Iona, NCR, etc.), as well as major projects underway at SunSoft, HP, and IBM to develop implementations bundled with their operating systems. DEC and Microsoft are jointly developing the Common Object Model, comprising productized support for CORBA/OLE2 integration across a dozen operating system platforms. A more extensive introduction to CORBA is published in [DPOM].

OMG IDL

A key component of CORBA is the Interface Definition Language (OMG IDL). OMG IDL is a syntax for formal specification of object-oriented interfaces. Its principal benefit is its independence from implementation. OMG IDL specifications are programming language independent, operating system independent, and physical process allocation independent.

OMG IDL has been called a "standard to define other standards," and it is being used for this purpose by the on-going OMG activities and groups such as the X Consortium, the OSF and the ISO ODP Trader working group. OMG IDL benefits standards authors and architecture designers because it can be mapped to multiple language bindings.

OMG IDL specifications are compiled directly into application program interfaces (API). For example, an OMG IDL specification can be compiled into a C header file which contains all the type definitions, constants, and function prototypes defined indirectly in IDL. Similarly, OMG IDL could be compiled into Ada specification code, C++ classes, Common Lisp definitions, and so forth. This paper describes an OMG IDL mapping for Common Lisp.

OMG IDL specifications are equally applicable to Common Lisp as they are to any other language, given the existence of an OMG IDL mapping to Common Lisp. The OMG IDL mapping to the C programming language was adopted as part of CORBA. Standard bindings for C++, and Smalltalk are active in the OMG process. Ada and OO Cobol mapping standardization efforts will begin soon.

Example OMG IDL mapping to Common Lisp

The authors have defined a preliminary specification for an OMG IDL mapping to Common Lisp. We used the Common Lisp standard with CLOS extensions as our language guidelines to ensure portability between Lisp implementations. [STEELE][CLOS] We also considered previous work on Common Lisp bindings to ORB-like facilities. These previous projects include NEC's Dynamic Invocation Interface for C and CLOS, BBN's CRONUS, and Xerox's ILU [NEC] [CRONUS] [ILU]. The significance of this specification is primarily as a guideline and leverage to Common Lisp vendors who wish to pursue the OMG process.

Developers can use this specification to layer OMG IDL interfaces to Common Lisp on top of language bindings, RPC calls, and ORB interfaces. The authors are using this binding with a commercial CORBA-compliant ORB by layering these interfaces on top of the C language binding. OSF has provided an example of how OMG IDL specifications map into their DCE RPC environment in [DCEIDL].

An OMG IDL compiler is responsible for creating several outputs based upon an input IDL specification. The first output is a set of client-side API definitions in Common Lisp. These definitions provides the client's API to user defined interface classes in IDL. A second compiler output is the client-side stub functions that implement the client-side APIs and deliver messages to the ORB. The third output is object implementation skeletons that the server needs for its interface to the ORB. By convention, a standard language binding focuses on the first output (the client-side definitions) leaving the ORB-dependent details to the ORB implementors. A portable ORB-independent implementation approach is described in the next section. (see Implementation Approach)

OMG IDL provides a set of data structure definition constructs similar to C++. In our example OMG IDL mapping to Common Lisp, we have mapped data structure types in a straightforward way. As our first example, consider the definition of a structure type in OMG IDL:

```
// OMG IDL
struct example1 {
     long field1;
     float field2; };
```

The mapping into Common Lisp comprises this definition:

```
; Common Lisp
(defstruct example1
   (field1 0 :type long)
   (field2 0.0 :type float))
```

We used a corresponding approach for most data types. Basic types in OMG IDL map into comparable basic types in Common Lisp, IDL array types map into

Common Lisp arrays, strings map into strings, and so forth.

The interface construct in OMG IDL is the principal mechanism for defining object encapsulations. A simple example follows, comprising an interface `example2` with a single operation `op1` and a user defined parameter `arg1`:

```
// OMG IDL
interface example2 {
     void op1(in long arg1); };
```

The mapping into Common Lisp comprises these definitions:

```
; Common Lisp
(defclass example2 () () )
(defmethod op1 (
     (o example2)
     arg1)
   (declare (integer arg1))
   (example2-op1-stub o arg1) )
```

In example 2, we see that the IDL interface is mapped into a Common Lisp class. The class `example2` has a method definition `op1` with a parameter pattern list that includes a typed object class parameter (which was implicit in IDL) followed by the user defined parameter `arg1`.

The following example, builds upon the previous example by using inheritance. A new interface `example3` is defined as a subtype of interface `example1`. The subtype inherits the operation `op1` definition from `example1` and adds an additional operation `op2`:

```
// OMG IDL
interface example3: example2 {
     void op2( in long arg2,
        out long arg3,
        out float arg4); );
```

The mapping into Common Lisp comprises these definitions:

```
; Common Lisp
(defclass example3 (example1) () )
(defmethod op1 (
     (o example3)
     arg1)
   (declare (integer arg1))
   (example3-op1-stub o arg1) )
(defmethod op2 (
     (o example3)
     arg2)
   (declare (integer arg2))
   (example3-op3-stub o arg2) )
```

The arguments `arg3` and `arg4` are output parameters that can be accessed through a `multiple-values-bind` in Common Lisp.

The remaining features of the example mapping are documented in our specification. Retrieval instructions are included in the conclusions section.

Implementation Approach

The implementation of the OMG IDL mapping to Common Lisp can be highly leveraged upon existing work.

The OMG is distributing a public domain shareware OMG IDL compiler toolkit. It is being distributed via FTP from omg.org, contact request@omg.org for details. This toolkit has a working version of all phases of OMG IDL compilation except code generation. In order to apply it to Common Lisp, an additional code generator needs to be written to generate the Common Lisp definitions and stub functions that call the ORB.

A strategy for the ORB integration layer is to use an existing standard mapping that multiple ORB products support. For example, the C binding is stable and has been implemented by several ORB products.

Conclusions

The benefits of OMG adoption of a OMG IDL mapping to Common Lisp include:

- reduction of risk for commercialization and end-user support for the OMG IDL mapping to Common Lisp
- extended capabilities of Common Lisp
  - support for distributed processing using ORB's
  - heterogeneous multiplatform transparency through ORB's
  - multiple foreign language bindings (all consistent)
- foreign language bindings consistent between Lisp environments
- consistent interfaces to ORB's and other CORBA compliant software components
- mapping does not change Common Lisp or CLOS language specifications

The next step is for one or more commercial sources of Common Lisp to pursue the OMG process. This includes obtaining an OMG Corporate Membership (a modest dues payment), and initiating the OMG process either through an RFP or an RFC.

The fast track RFC process has some advantages and a risk. It's faster than an RFP process and it can be initiated at any time; whereas an RFP requires more resources and must be timed with respect to the task force's schedule. The risk is that the RFC can be withdrawn by the OMG if it receives "significant comment" during the comment period.

How to Obtain the Draft Specificiation

The authors have drafted an OMG IDL mapping to Common Lisp specification, summarized above, which

we would like to release for general comment. This specification can be obtained from the OMG server by sending the email message:

```
get docs/94-3-11.ps
```

to the Internet address: `server@omg.org`

Acknowledgments

We appreciate the help and input from Jim Veitch and Stuart Elliot at Franz, Neil Feinberg and Scott McKay at Harlequin, Cameron Kemper and Jay Mellman at Lucid, Steve Strassman at Apple, Bill Janssen at Xerox, and MITRE contributors including: Diane Mularz, Ron Zahavi, Melony Katz, Webster Anderson, Kurt Louis, Wes Hamm, Greg Whittaker, Jeff Graber, Paul Silvey, Tana Reagan, Malcolm McRoberts, Martha Farinacci, Vic Giddings, Richard Tucker, Chris Elsaesser, Hans Tallis, Scott Musman, and Raphael Malveau.

Bibliography

[CBOS] TJ Mowbray and T Brando. Interoperability and CORBA-based open systems, Object Magazine, October 1993.

[CLOS] Sonya E. Keene, *Object-Oriented Programming in Common Lisp: A Programmer's Guide to CLOS,* Addison-Wesley, New York, 1989.

[CORBA] Object Management Group. *Common Object Request Broker Architecture And Specification,* Document 91.12.1, Framingham, MA, 1991.

[COSS] Object Management Group. *Common Object Services Specification,* Volume I, Framingham, MA, 1993.

[CRONUS] BBN Systems and Technologies, *CRONUS Programmer's Reference Manual (Lisp),* Release 3.0, Cambridge, MA, 1992.

[DCEIDL] DEC, HP, HyperDesk, IBM, NEC, and OSF. Joint submission on interoperability and initialization, OMG TC Document 93-3-5, March 7, 1994.

[DISCUS] J. Fleisher and TJ Mowbray, Integrating tools and data sources using the DISCUS Framework, Proceedings of the AFCEA-ITEMS Conference, Washington DC, 1993.

[DOE] TJ Mowbray. Distributed Objects Everywhere: An early assessment, Object Magazine, January 1994.

[DPOM] TJ Mowbray and R. Zahavi, Distributed processing with object management, ConneXions--The Interoperability Report, Interop Corporation, December 1993.

[ILU] Bill Janssen and Mike Spreitzer, *Using ILU with Common Lisp,* Technical Report, Xerox Corporation, January 1994.

[NEC] NEC, *NEC CORBA Sample Implementation,* Document 93-4-28, Object Management Group, Framingham, MA, 1993.

[OMAG] Object Management Group. *Object Management Architecture Guide,* Document 92.11.1, Framingham, MA, 1992.

[STEELE] Guy L. Steele Jr, *Common Lisp: The Language, Second Edition,* Digital Press, Digital Equipment Corporation, USA, 1990.

Authors

Tom Mowbray, PhD is a Lead Scientist in the MITRE Workstation System Engineering Center in MITRE-Washington. He is the architect of the DISCUS Framework, an OMG IDL-compliant software architecture which is being reused on several testbeds both inside and outside of MITRE. [DISCUS] He has written an OMG IDL compiler, and has training and experience on several CORBA-compliant ORB's including: SunSoft's DOE, HyperDesk's HD-DOMS, and DEC's ObjectBroker. [DOE][CBOS] He is also MITRE's principal representative to the OMG and Chairperson of the OMG Common Facilities Task Force, a group responsible for all vertical market and specialty area framework specifications.

Kendall White is a Member of Technical Staff in the MITRE Workstation System Engineering Center at MITRE-Washington. He is the lead developer and integrator of Common Lisp with CORBA. His other responsibilities include developing client/server interfaces between C and Smalltalk, and Object-Oriented Database schemas. He is also an organizer of object technology workshops and panels at the TOOLS USA and OOPSLA conferences.