Dynamic reconfiguration with CORBA and Common Lisp: A position paper

Lewis Stiller[1]
Franz Inc.

1.0 Introduction

We are developing a CORBA-compliant Object Request Broker for Common Lisp. This project comprises three phases: design of the mapping, implementation of the ORB, and validation in applications.

Our original goal in designing this ORB was to provide CORBA connectivity to existing Lisp users. As we have gained experience in CORBA development, we are also exploring possibility that our ORB will offer dynamic reconfiguration features that are independently useful.

2.0 Motivation

By dynamic reconfigurability of an ORB we mean support for functionality changes in deployed clients and servers on an adaptive basis without service interruption.

Dynamic reconfigurability is useful in a number of contexts.

2.1 Incremental development

CORBA applications typically comprise multiple autonomous processes interacting via an ORB. In development it is often the case that one of the constituent programs needs to be modified in order to correct a programming error or to test a new feature.

In languages such as C, C++, and Java such modifications can entail recompiling the IDL, recompiling the application, restarting all of the processes that interact, and restoring the state of the system to the situation of interest; this is true in many cases for Java applications as well. This process results in a slower development cycle than would be the case if the server could be modified on the fly.

1. Address correspondence to stiller@franz.com. Mailing address: Franz Inc, 1995 University Avenue, Berkeley, CA 94704. The Franz Inc. CORBA group comprises Lewis Stiller, Steve Haflich, Bob Rorschach, and Jim Veitch.

2.2 Adaptive applications

The second difficulty with many ORBs is that it is difficult to write applications that modify and adapt to their environment. Most languages require a clear demarcation between the SII, or static invocation interface and the DII, or dynamic invocation interface (similarly for the static vs. dynamic skeleton interfaces). If large parts of the application are configurable, much of it must be written using the DII and DSI, which are typically slower and much more cumbersome than their static analogues.

3.0 Dynamic reconfigurability

In this position paper we would like to explore two related themes.

The first is that dynamic reconfigurability—the ability to modify an application without recompilation on the basis of run-time information—is a useful feature in CORBA application development. The second is that Common Lisp is well-suited to supporting this functionality.

In a dynamically reconfigurable environment, modification of server functionility does not require that the server be restarted. Instead, its functionality is modified on the fly.

Unless the client makes use of the DII server modification will require the new IDL to be propagated to the clients. However, in a dynamic environment the clients do not have to be recompiled and restarted; they can maintain persistent state.

4.0 Common Lisp and reconfigurability

Common Lisp provides support for certain features facilitating the development of reconfigurable programs. These features include dynamic modification of class and function definitions, making an object be an instance of a class different from its original class, multiple inheritance, first-class closures, and interpretive semantics.

The Franz Common Lisp ORB represents an attempt to meet the challenges described in the previous section by exploiting the pre-existing dynamic reconfiguration of CLOS, the Common Lisp Object System.

CLOS allows classes and methods to be redefined interactively. The semantics of the modification of a class definition—for instance, the behavior of objects that gain or lose slots—is an integral part of the language definition. In addition, the class of an object can be modified at run-time, and the semantics of class dispatch can be modified using the meta-object protocol. Because CLOS supports multiple inheritance, new interfaces can be added to an implementation object at run time. Such features facilitate the evolution of programs in a controlled and well-defined manner.

For example, in order to add an interface or an operation to the server, the server’s copy of the IDL file would be modified and the implementation code recompiled (without taking

down the server). This would automatically make the new interfaces and operations available to clients. The new interfaces could be published via the IDL file itself or, more generally, via the interface repository.

There would be a potential loss of type-safety, however, if the signature of an operation were modified without informing the client proxy of the change; this would normally result in an exception being raised.

5.0 Status

We have circulated several versions of a mapping proposal from the URL http://www2.franz.com/~darpa/corba. This proposal is being reviewed by several of our industrial partners. After revising based on their comments we intend to initiate the OMG formal adoption process.

Our ORB, which uses native IIOP, has been tested in internal prototype, but is not complete. Our schedule is for the ORB to be available in alpha pre-release in December of 1997 and in general availability in the first quarter of 1998.

Thus, at the time of this writing we have not been able to verify the utility of our ORB for the development of dynamically reconfigurable applications on non-trivial applications, although we have found it useful already in small test cases.

On the other hand, it is worth noting that several industry groups, such as the ILU group, Ascent, and some CAD firms, have independently implemented some level of CORBA connectivity for Lisp. In general, the published experience of these companies, for example that of Ascent has tended to support the feasibility of our claims.

6.0 Conclusion

We have introduced a CORBA-compliant object request broker for Common Lisp, and we have shown how it lends itself to the design of dynamically reconfigurable CORBA servers.

A promising area for future work is the integration of this dynamic ORB into two comparatively recent CORBA facilities: the meta-object facility and the mobile agents facility.

The former could be used to provide applications with dynamic information that is more complete and more useful than the interface repository alone. The latter could provide new target applications particularly well-suited to dynamic reconfiguration.