omniORBpy and omniidl—CORBA for Python

Duncan Grisby

AT&T Laboratories Cambridge
24a Trumpington Street
Cambridge CB2 1QA, UK
dgrisby@uk.research.att.com

Introduction

omniORBpy is a free (LGPL and GPL) implementation of the new CORBA to Python mapping. It is built on top of omniORB, one of the most efficient and robust C++ ORBs available.

Why CORBA?

In our laboratory, many of the projects are based upon a CORBA framework. CORBA allows developers to think about application functionality without having to spend time worrying about the details of network communication and interoperability. We started developing our own ORB, omniORB, since other ORBs had some serious shortcomings, and were not available for all of the platforms we use. omniORB is now available on all major platforms, and is in widespread use outside, as well as inside, our laboratory.

Why Python?

With a large and expanding collection of complex CORBA-based systems, we have increasingly found ourselves having to write simple programs to 'glue' existing systems together, and wishing to write more extensive prototype applications. Python is an ideal language for both. The fact that CORBA operations can be invoked from Python's command line also provides a valuable debugging tool.

In addition, we find that many of our applications have some portions which are best implemented in C++, and others which would be neater if they were written in a scripting language. By providing a Python binding for our C++ ORB, we are able to have C++ and Python CORBA objects coexisting in a single address space, sharing the ORB runtime.

omniORBpy design

omniORBpy has been designed to be as fast as possible, without compromising Python's dynamic properties. The majority of the ORB runtime is implemented in C++, relying on Python only where necessary. In particular, arguments are marshalled and unmarshalled by a set of generic functions which are able to read and write Python data structures, through the Python C API, without executing any Python code. These functions are driven by type descriptors which can either be statically created by the IDL compiler, or dynamically created at run-time.

Interoperability

One of the primary goals of the omniORB project is complete compliance with the CORBA specification. To this end, omniORB is one of only three ORBs to have passed the official CORBA branding programme run by the Open Group. omniORB completely adheres to the IIOP (Internet Inter-ORB Protocol) specification, so it interoperates with any other compliant ORB. In practice, omniORBpy communicates seamlessly with the majority of other ORBs. There are a number of known bugs in some ORBs, for which omniORB contains work-arounds.

omniidl

CORBA types and interfaces are declared in an Interface Definition Language (IDL), which must be compiled into declarations in the target programming language. IDL compilers are traditionally very poorly structured, mixing compiler logic with fragments of code to be output. This makes compiler maintenance extremely difficult.

omniidl is a new IDL compiler which is used to produce both Python and C++ stubs for omniORB. The front-end parser is written in C++ (with help from flex and bison); back-ends for the different target languages are all written in Python.

Back-ends store the code to be output as a set of template strings, containing tags marking those parts which vary according to the input IDL. There is therefore a clean separation between the logic which decides what code to output, and the fragments of code themselves. This approach, combined with the use of Python, makes writing and maintaining back-ends far simpler than it is with other compiler designs.

Having an easy-to-use IDL compiler has enabled us to experiment with generating special-purpose code based upon IDL definitions. Work so far has explored automatic notification of attribute changes, and caching of attribute values.

More information

Further information about omniORBpy can be found at http://www.uk.research.att.com/omniORB/