Vacillation Made Easy: Distribution, Re-distribution, and Un-distribution of DOPL-based Processing

Scott Burleigh
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, M/S 301-270
Pasadena, CA 91109
scottb@puente.jpl.nasa.gov

Extended Abstract

Distributed Objects Protocol Layer, or DOPL, provides a simple and general data communication abstraction that can support the distribution of C++ applications software functionality among an arbitrary collection of processors. The purpose of the abstraction is to minimize the cost of revising processing distribution decisions throughout the software development cycle, including after software has been delivered to users.

DOPL comprises an application-level protocol for communication among C++ objects and an application programming interface (API) that implements this protocol. No extension to the C++ programming language is required to implement DOPL, and the API is platform-independent.

Specifically, the DOPL abstraction enables the application code in the definition of class X that specifies transmission of data to object A to be the same regardless of whether A is instantiated within:

- the same UNIX process as the transmitting object;
- a different process on the same processor;
- a different processor in a common hypercube;
- a different processor on another continent, connected by TCP/IP, OSI, SNA, whatever;
- an MS-DOS executable on a personal computer elsewhere on a NetWare LAN;
- a flight computer aboard a spacecraft orbiting Mars, 11 light minutes away.

An experimental system for tracking expected changes in spacecraft state over time, called SST, has been implemented at JPL using DOPL. A software model of part of the Galileo spacecraft's attitude and articulation control subsystem was developed by designing C++ classes that could instantiate 40 DOPL-capable C++ objects. An additional DOPL-capable C++ object managed a

graphical user interface built on the X Window System and Motif widget set. To investigate the degree to which parallelizing SST might increase its execution speed, the system's performance was measured in various serial and parallel configurations. Parallelizing SST proved not to be advantageous. More significantly, however, the DOPL implementation enabled us to develop all of these configurations at run time merely by editing a simple initialization file.

The original aim of DOPL was merely to simplify the reconfiguration of distributed applications, as in the experiment cited. During development, however, DOPL gradually evolved into an effort to take "object-oriented" thinking to a logical extreme, augmenting the native capabilities of C++ objects to the point at which objects, rather than operating system processes or tasks, could be regarded as the atoms of software organization.

Implicit in this view is a change in the way C++ is used. Perhaps the prevailing archetype for C++ applications today is the fairly conventional ANSI C program that operates on data aggregations -- objects -- by invoking the member functions defined for those objects. DOPL encourages the developer instead to view a C++ application as a collection of objects that exchange messages with one another and, possibly, the wider processing universe. Objects, rather than programs or processes, are anthropomorphized. A "system" takes the form of a population of conceptually autonomous objects, rather than a hierarchy of procedures for manipulating data.

As a result:

- change in the distribution of application functionality is facilitated, because inter-object communication is the same for both local and remote destination objects;
- application integration is accelerated, because inter-object communication protocol is standardized;
- application portability is enhanced, because inter-object communication is platform-independent;
- software re-use is simplified, because the interface to an instance of a given DOPL-capable class is universally accessible to any DOPL-capable object.

Applying DOPL in a given venue does require adjustment to a non-traditional system style, but we contend that the advantages justify this investment.

Acknowledgement

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.