OMOS — An Object Server for Program Execution

Douglas B. Orr                  Robert W. Mecklenburg
Department of Computer Science, University of Utah
Salt Lake City, UT 84112
dbo@cs.utah.edu, mecklen@cs.utah.edu

Abstract

The benefits of object-oriented programming are well known, but popular operating systems provide very few object-oriented features to users, and few are implemented using object-oriented techniques themselves. In this paper we discuss a mechanism for applying object-oriented programming concepts to program binding (linking) and execution. We describe OMOS, an object/meta-object server that embodies a flexible object framework. The OMOS framework projects an object-oriented structure onto programs and shared libraries that may not have been originally developed for use within an object-oriented environment. This framework provides natural facilities for inheritance, interposition, and overloading of operations, as well as development of classes with dynamically evolving behavior.\(^1\)

1 Introduction

In recent years object-oriented programming has gained widespread support due to its facilities for controlling modularity, division of responsibility, support for code reuse, and scalability[21]. We believe these features can be profitably applied to the problem of program binding and execution to achieve a more elegant solution than is currently available, while also providing increased functionality. We present a mechanism for applying object-oriented programming concepts to program binding (linking) and execution. We also describe the implementation of OMOS, an object/meta-object server which implements these concepts as a process in the Unix\(^2\) operating system.

Current technology for constructing programs from modules is clumsy and lacks structure[24]. This clumsiness results in inefficiency which manifests as poor use of programmer skills, poor locality of program reference, poor reuse of existing code, poor use of virtual address space, and poor use of cpu time (i.e., avoidable recalculations). This lack of structure is, of course, not without reason. Historically, there has not been a structure which seemed sufficiently comprehensive and robust to encompass current techniques while providing the increased functionality of the object-oriented paradigm. Also, the large investment in existing techniques makes moving to an incompatible structure costly.

We believe that object-oriented programming concepts can be applied to existing techniques such that many of these inefficiencies can be overcome while still taking advantage of existing technology. We begin the discussion with a review of object concepts and how they relate to programs and name binding. We then discuss recent work which clarifies the relationship between modules and inheritance. After these preliminaries we will describe the features an object server must possess and a sample architecture. This discussion is followed by a more detailed discussion of a prototype implementation. Finally, we will review some results and make some observations on the concept of the OMOS system.

2 Objects and modules

An object is a collection consisting of some member data (called slots) and some member functions operating on that data (called methods). In this way an

\(^{1}\)This research was sponsored by Hewlett-Packard’s Research Grants Program and by the Defense Advanced Research Projects Agency (DOD), monitored by the Department of the Navy, Office of the Chief of Naval Research, under Grant number N00014-91-J-1046. The opinions and conclusions contained in this document are those of the authors and should not be interpreted as representing official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the U.S. Government, or Hewlett-Packard.

\(^{2}\)UNIX is a trademark of AT&T.