8-25-1995

Distributing Object-Oriented Systems

Sandeep Purao
Georgia State University, cissrp@gsusgi2.gsu.edu

Hemant Jain
niversity of Wisconsin-Milwaukee

Derek Nazareth
University of Wisconsin-Milwaukee

Distributing Object-Oriented Systems

Sandeep Purao, Dept of CIS, Georgia State University Atlanta, GA 30302. cissrp@gsusgi2.gsu.edu
Hemant Jain, Department of MIS, University of Wisconsin-Milwaukee, Milwaukee, WI 53201
Derek Nazareth, Department of MIS, University of Wisconsin-Milwaukee, Milwaukee, WI 53201

Object Distribution

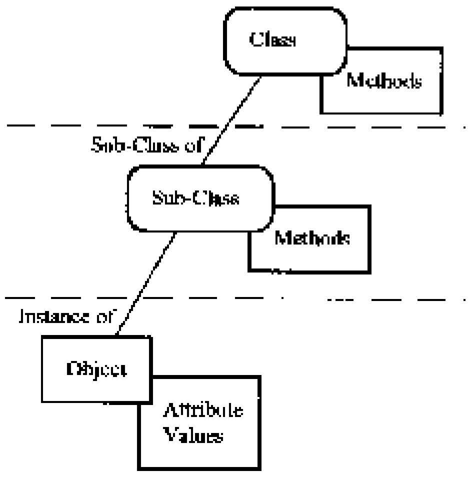

Figure 1: Object Distribution Possibilities

Effective distribution of system components has been a concern of system designers for many years. As systems design paradigms have evolved, techniques for system distribution have attempted to keep pace. The most recent challenge is the object-oriented paradigm. While introducing interesting challenges (due to encapsulation and inheritance), the object-oriented paradigm offers, for the first time, the opportunity to consider integrated approaches to distribution of data and behavior. After some early efforts, researchers have started to investigate the object distribution problem with an implementation-independent perspective. Figure 1 below (adapted from [1]) shows natural separations in the object-oriented models that researchers are beginning to exploit.

The first: object/class separation presents the opportunity of treating object instances and class implementations as distinct units of distribution. The threat to encapsulation, arising from such approaches, is averted by routing access to objects through the class interface. The class itself contains a myriad of methods, which can be considered as distinct units of distribution. The second: subclass/superclass separation represents inheritance, which allows behavior sharing. Realization of this may, however, involve dynamic binding, requiring run-time probes through the class hierarchy. The expense associated with these searches has prompted some researchers to state that '.. inheritance is inherently incompatible with distribution'[2].

Significant players and consortia in the industry have also been active. Many *de facto* and planned standards are emerging to address object distribution at different levels. Prominent among these are Object Management Group's [3] (OMG) Common Request Broker Architecture (CORBA) (see figure 2), and Object-Oriented Database DataBase Task Group's (OODBTG) Object Data Model (ODM) [4].

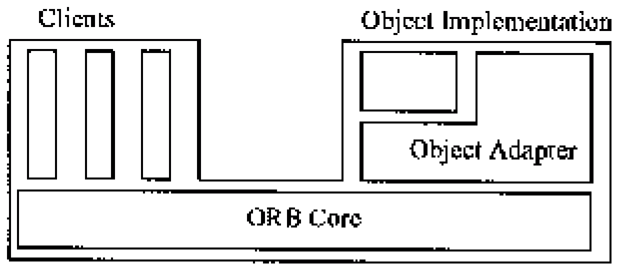

Figure 2: CORBA Reference Model by OMG

The mechanisms suggested by OMG include Object Adapters that can act as managers at different locations, and with which objects and classes can register themselves. By disengaging interface from implementation, these mechanisms allow partitioning and allocation of the implementation. OODBTG's ODM covers a larger problem space compared to that of OMG's CORBA. In an ODM system, things which may be distributed include objects, operations, classes, ODM system functions, ODM system processes, and user processes. Distribution of ODM objects may involve any of the following: making objects the unit of partitioning, such that an object or replica of an object exists on one computer system; spreading implementation of an object across multiple computer systems; or grouping objects together [5].

A Framework for Object Distribution

It appears that object distribution possibilities envisioned by academia and planned by industry have much in common. Both communities agree that distributable elements in the object-oriented paradigm include: the partial or complete state of an object, collections of objects, the class template, and methods implemented in the class template. It is obvious that these elements belong to different levels of abstraction. Platforms across which these may be distributed include geographically dispersed sites, and heterogeneous processors within each site. These platforms too, present different concerns, and clearly belong at different levels. It is clear that multi-echelon framework that recognizes these differences is needed. Of the decomposition strategies suggested by Schoeffler [6], decomposition on the basis of influence appears to be the most appropriate for object distribution. Schoeffler describes it as:

"a ... problem is partitioned or structured in such a way that it can be solved sequentially in levels or strata with the result or output of one stratum (a higher-numbered one) serving as partial input to another lower-numbered stratum. Thus, from the top level down, the decision-making process is similar to a staged process rather than a completely interacting one."

Table 1 shows the Object Distribution Framework we propose that utilizes the 'decomposition by influence' technique.**Table 1: Object Distribution Framework**

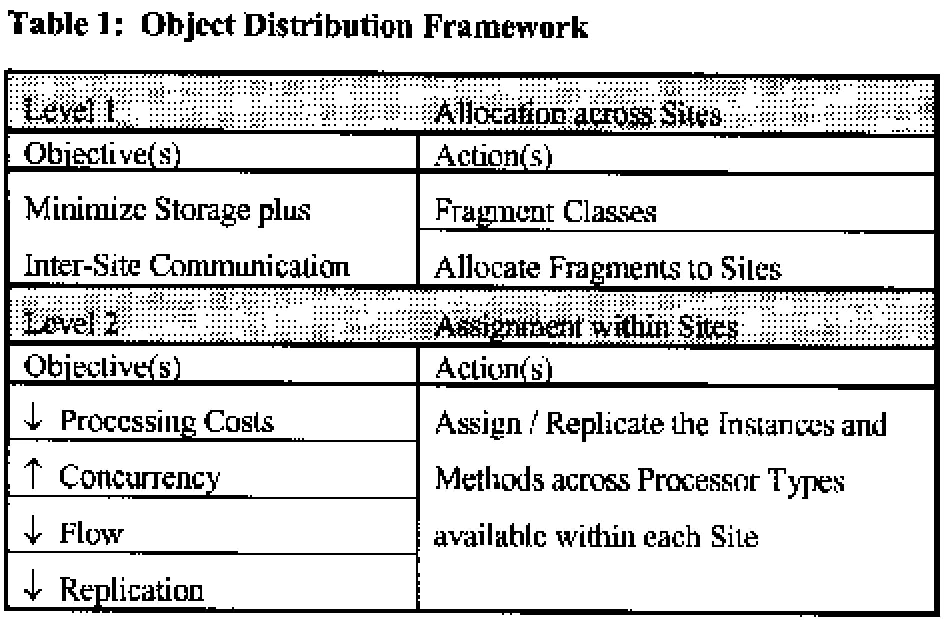

Table 1: Object Distribution Framework

Level 1	Allocation across Sites
Objective(s)	Action(s)
Minimize Storage plus Inter-Site Communication	Fragment Classes
	Allocate Fragments to Sites
Level 2	**Assignment within Sites**
Objective(s)	Action(s)
↓ Processing Costs	Assign / Replicate the Instances and Methods across Processor Types available within each Site
↑ Concurrency	
↓ Flow	
↓ Replication	

At level 1, the problem is modeled and solved, using the Class as a conceptual tool [7]. The units of distribution at this level represent collections of object instances, along with the class template, which are allocated to the appropriate sites. Following Wegner [2], consideration of inheritance is postponed to level 2. The results of level 1 are used at the next level for distribution of object instances and methods over the heterogeneous architecture within each site. The objectives mentioned at each level are dictated by choice of distributable units and significant concerns at each level. The framework has been instantiated in the form of a comprehensive methodology for distribution of object-oriented applications [8] that squarely addresses the question:

How should an object-oriented application be distributed over existing heterogeneous architectures at geographically dispersed sites?

At level 1, the methodology extends some research from distributed relational databases, proposes new algorithms for fragmentation of classes, and formulates allocation models. At level 2, it utilizes and extends research from the MCDM area for intra-site distribution of object instances (considering inheritance), and methods. The proposals have been implemented in a working prototype called Object Distribution Environment (ODE), and verified by using ODE for distribution of a moderate-sized marketing information system from a midwestern utility company.

References

[1] Blair, G. and R. Lea. 1992. The Impact of Distribution on Support for Object-Oriented Software Development. In *Software Engineering Journal* (March) pp. 130-138.

[2] Wegner, P. 1987. Dimensions of Object-Based Language Design. In *Proceedings of Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA 87)*. pp. 168-182.

[3] Object Management Group. 1992. *Common Object Request Broker : Architecture and Specifications Rev. 1.1 OMG Document Number 91.12.1*. Object Management Group, Framingham, MA.

[4] Ballou, M. 1994. Standards Set to Take Hold. In *Computerworld, 28(1)*. December 27, 1993/January 3, 1994. Pp. 91, 93.

[5] Fong, E., W. Kent, K. Moore, and C. Thompson. Ed. 1991. *X3/SPARC/DBSSG/ OODBTG Final Report*.

[6] Schoeffler, James D. 1971. Static MultiLevel Systems and On-line Multilevel Systems In *Optimization Methods for Large-Scale Systems* ed. David Wismer, McGraw-Hill.

[7] McGregor, J. and T. Korson. 1990. Object-oriented Design. In *Communications of the ACM, 33, 9*. (September). Pp. 39-59.

[8] Purao, S. 1995. *A Methodology for Distribution of Object-Oriented Applications*. Unpublished Ph.D. Dissertation. University of Wisconsin-Milwaukee.