PROJECT MAC
Computer Systems Research Division

October 14, 1975
Request for Comments No. 88

POSITION NOTE ON NEW DIRECTIONS FOR OPERATING SYSTEMS

by J. H. Saltzer

The enclosed letter is a first expression of some ideas about object-oriented systems.

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY

545 Technology Square
Cambridge, Massachusetts
02139

PROJECT MAC

Telephone: (617) 253-6016

October 9, 1975

Professor J. C. Browne
The University of Texas at Austin
College of Natural Sciences
Department of Computer Sciences
Austin, Texas 78712

Dear Jim:

Thank you for your invitation to participate in the workshop on new directions for operating systems. Unfortunately, at the present time, it looks like I won't be able to make it.

You asked for position papers, and while this doesn't constitute one, maybe you and the other participants will find something to argue about in the following thoughts:

It seems to me that the major new opportunities in computer system design come from two directions:

1. Pressure from users for more easily programmed systems.
2. Modern technology, which is providing minicomputers at negligible cost, but without much support for easier programming.

These two pressures can be absorbed best, I am convinced, by object-oriented systems, of the type pioneered in software by CAL and HYDRA. Object management provides a user interface with a high granularity of function and it also directly supports structured programming, reliable software, and protection, all of which are buzzwords reflecting contemporary concern for how systems are programmed and used. At the same time, modern technology may be able to provide the hardware support needed to create an effective object-oriented system.

And there, I would guess, is the hard problem: creating <u>effective</u> object-oriented systems. Both CAL and HYDRA, being software interpreters of the object concept within traditional hardware architecture, are not prepared for intensive use of large numbers of objects managed by hundreds or thousands of type managers.

One of the key problems of object oriented systems lies in discovering strategies for organizing the "map", or collection of hardware and software tables that the base-level system uses to keep track of the objects. In Multics, which is a primitive example of a system that implements objects--segments--the mapping tables are an area of frightening complexity, and because of the space they use and the management overhead they generate they inhibit free use of segments as computing objects. These tables support:

- mapping from unique names to physical representation of an object,
- ability to bind all manner of local names, both human-readable and machine loadable, to an object,
- connection among alternate physical representations of a single object to support multilevel memory management,
- connection between program accessible representations of an object and long term storage representation of the same object,
- connection to backup copies and earlier generations of an object, for reliability,
- collection of performance data, for use as input to resource management algorithms,
- connection to accounts, for charging for use of resources and enforcing quota restrictions,
- objects as the unit of authorization, for purposes of sharing and protection,
- revocation of authorization to use an object,
- distinct permission ("rights") for different operations on a single object,
- "type management", meaning that accessibility to the representation of an object requires authorization by the owner of the object and by the type manager programs.

This is a lot of function! And it is implemented in Multics by a collection of interconnected tables [1]. and giving each presumably modular system routine direct access to the tables so that it can work its magic. As a result, programming the catalog hierarchy manager is done taking into account special problems of the page manager, the traffic controller, access control, accounting, and backup. And vice-versa.

This lack of modularity in supervisor organization should perhaps be construed as bad design decisions by people who should have known better, but I think that the problem is deeper: the list of functions surrounding objects is long and growing. One should add user-extendible types as in CAL and HYDRA. There is continuous pressure to reduce the size of the smallest object that can be efficiently supported, thereby increasing the number of objects, the effort

---

[1] Bensoussan, A., C. T. Clingen, and R. C. Daley, "The Multics Virtual Memory: Concepts and Design," Comm. ACM 15, 4 (May, 1972) pp. 308-318.

of table management, and the strategic load on storage allocation and especially on multilevel memory management. It is simply not apparent how to do this function all at once, and efficiently.

All of these observations lead me to the conclusion that the design and implementation of a really practical object-oriented system that sensibly balances the objectives implied by the above list of table functions is a challenging research project.

There are, of course, other closely related issues. Can minicomputers be extended to object-oriented architectures without blowing their economics? What about the notion of building an apparently centralized system out of a collection of geographically distributed minicomputers? The meaning of "apparently centralized" is probably that there is logically a single object map for the system, so geographical distribution should be added to the list of complications for the object map. Should personal, desk-top computers be object oriented? If so, how should the objects of one personal computer refer or relate to the objects of another with which it is communicating or to a central library system? Again, these questions might be approached by considering the implications for the object mapping apparatus.

Good luck with your workshop.

Sincerely yours,

Jerome H. Saltzer
Associate Professor
Head, Computer Systems Research Division

JHS/mw