# Application-oriented Object Architecture: A Revolutionary Approch

**Ramesh K. Karne**

Department of Computer and Information Sciences, Towson University
Towson, MD 21252 USA
rkarne@towson.edu

**ABSTRACT**

Since last fifty years computer systems and architectures followed an evolutionary path resulting in rapidly changing hardware and software. This paper proposes a revolutionary approach, application-oriented object architecture (AOA), which is based on Application Object (AO) and self managed hardware units. We have demonstrated sample applications such as sorting and editing using the above approach. Currently, we are in a process of understanding a Web server application to be run under AOA. When AOA becomes feasible, it provides a revolutionary approach that will change the information technology horizon so that computer systems can be built similar to engineering buildings.

**Keywords**

Application Object(AO), Application-oriented Object Architecture (AOA), and Embedded OS.

**INTRODUCTION**

Current computer systems are rapidly changing environments including: hardware, operating systems, compilers, tools, programming languages, and so on. *We must build computer applications that are independent of operating systems and environments, yet extensible, secure, and stable*. The OSKit [2] approach attempts to avoid traditional operating systems and provides a necessary kernel for applications. Exokernel [1] offers an application level kernel interface for fast server applications bypassing the traditional operating system. The Oxygen project at MIT [4], and other distributed OS projects [5], offer hope to build pervasive and distributed computing with embedded building blocks and natural language interfaces. Our goals are similar to the above research, but it differs in its approach. The application-oriented object architecture (AOA) [3] focuses on applications instead of computing environments. It is based on primitive computer applications such as sending a message, editing a document, sorting data, computing FFTs, performing a purchase online, and so on. That means, these applications are abstract and will never change, however, they may need to be extensible. For example, they should not depend upon any OS or any devices attached to a system. Once we understand these primitive applications, the application object (AO) can be designed, and it can be run in the hardware without any need for a particular environment. For example, a bubble sort program written today in a given programming language such as C++ should be able to run on any computer as long as the hardware provides execution environment for this AO. This forces hardware vendors to provide upward compatibility and keep the AO interfaces stable so that old and new AOs can be run without any need for operating environment. When an AO can run without any operating environment on a bare machine, then it offers total flexibility in AOs and stability in hardware and software evolution and impedes computer obsolescence and waste.

As the operating environment is bundled or embedded into application program, now the hardware has to manage itself and should provide direct interfaces to AOs. This embedded OS approach into applications forces the hardware to be self manageable and controllable. Initially, we can build these AO interfaces outside the hardware, however, once these interfaces are well understood, they can be directly implemented in the hardware.

**AOA ARCHITECTURE**

The AOA architecture is shown in Figure 1. It has AOs and self managed hardware units. A sample AO is shown in Figure 2. The AOs communicate to hardware directly through sample interfaces as shown in Figure 3. These are AO interfaces written using the lowest level of hardware interfaces such as (Basic Input Output System, BIOS) and should be standard across all hardware elements. That is, all hardware units such as processors, memory, disks, network cards, and so on should provide AO interfaces.

At any given time, there is only one AO running in the hardware. An AO is a self-controllable and self-executable unit that consists of all its program and data. When an AO is not running in the hardware, there is no code resident in the hardware (it is a bare machine). Thus, an AO performs, its own boot, load, and execution. It carries with it the only necessary code required for its execution. When an execution is done, it exists by carrying all its code and data with it to be stored outside the hardware. It may be stored on a network server for later processing.

An AO can communicate with another AO on a different machine using message objects (MOs). These message objects are traditional TCP/IP messages that can travel on the network and can be sent and received by a pair of AOs.

The AOs can also be designed to be secure in storage and in execution, but it is not addressed in this paper.

## AO IMPLEMENTATION

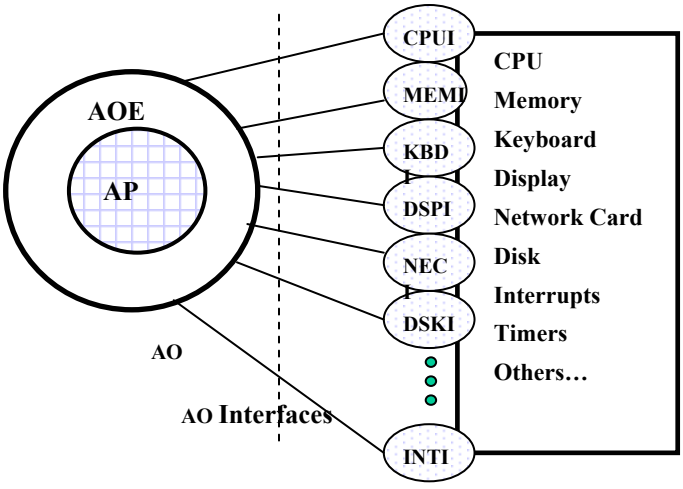
AOs can be implemented as executable modules by application-specific vendors. They can distribute these objects through a network server. There should be a standard programming language and compiler to be used to implement AOs. We have used the C++ language and assembly language to demonstrate the AOA. We have implemented AOs in C++ and the AO interfaces in assembly language. Using the current processor and memory technologies, we can build robust interfaces that can serve many applications. When these interfaces are matured, hardware vendors can implement them directly in hardware.

## CONCLUSIONS

We have presented a novel AOA architecture that is in preliminary stages of the development. When it becomes feasible, this may bring revolution in information technology. As the architecture is not traditional, it requires different mind set to design and implement computer hardware and software applications and its impact is not eminent at this point.


## ACKNOWLEDGMENTS

We thank Dr. Frank Anger and Dr. Spencer Rugaber at NSF for their support and suggestions and funding for this research as SGER grant (CCR-0120155).


**AOE: Application Operating Environment**

**AP: Application Program**

**AO: Application Object**

**Figure 3.** AO Interfaces

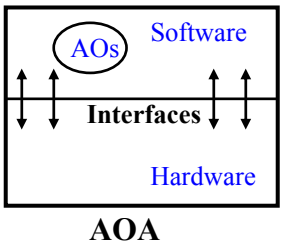



**Figure 1**. Architecture

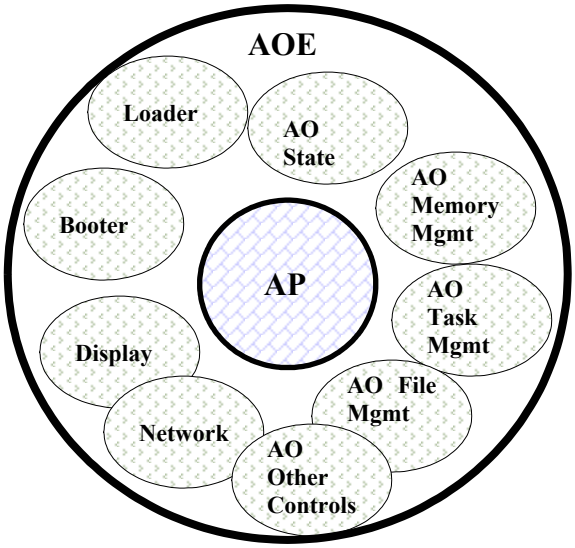



**Figure 2**. Sample AO Structure

## REFERENCES


1. Engler, D.R. The Exokernel Operating System Architecture, Ph.D. Thesis, MIT, October 1998.
2. Ford, B., Back, G., Benson, G., Lepreau, J., Lin, A., and Shivers, O. The Flux OSKit: A Substrate for OS and Language Research. In Proc. of the 16nth ACM Symp. on Operating Systems Principles, St. Malo, France, Oct. 1997, 38-41.
3. Karne, R.K, Gattu, R., Dandu, R., and Zhang, X. Application-oriented Object Architecture: Concepts and Approach. In IASTED Networks, Parallel and Distributed Processing, and Applications Conference, Tsukuba, Japan, October 2-4, 2002.
4. MIT Project Oxygen, Pervasive, Human-Centered Computing, http://oxygen.lcs.mit.edu.
5. Vaughan-Nichols, S.J. Developing the Distributed-Computing OS, IEEE Computer, September 2002, Volume 35, No. 9, 19-21.