Addison-Wesley

DENNIS LINNELL

The SAA Handbook

A Practical Approach to IBM's Systems Application Architecture

The SAA Handbook

A Practical Approach to IBM®'s Systems Application Architecture

Dennis Linnell

Addison-Wesley Publishing Company, Inc.

Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Wokingham, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan

Library of Congress Cataloging-in-Publication Data

Linnell, Dennis.

The SAA handbook : a practical approach to IBM's Systems Application Architecture / Dennis Linnell.

p. cm.

ISBN 0-201-51786-8

1. IBM computers--Programming. 2. Computer architecture.

I. Title.

QA76.8.I1015L56 1989

004.2'525--dc20 89-17826

CIP

ISBN 0-201-51786-8

Managing Editor: Amorette Pedersen

Set in 10.5-point Palatino by Benchmark Productions

ABCDEFGHIJ-MW-943210

First Printing, October 1990

To Victoria

Preface

Systems Application Architecture (SAA) is IBM's sweeping effort to standardize nearly all facets of application software design. Though comprehensive, SAA is complex, obscure, and potentially difficult to learn. This book describes the key architectural features and the products that implement them. I try to demystify the architecture by explaining the SAA lexicon in plain English, without using much hyperbole or marketing rhetoric.

If you need a technical understanding of SAA, this book is for you. I've assumed you are familiar with computer programming and application design. If you have experience with IBM products, you may appreciate the subtle humor hidden in the text. Otherwise, IBM knowledge isn't necessary; the book provides the necessary background and context.

The first chapter speculates about the reasons for SAA and the final chapter speculates about the future. Chapter 2 contains my analysis of the past two decades of IBM computers. The rest of the book is mostly factual. You should know that my forecasts are correct about 50% of the time; my opinions should be taken with a grain of salt.

The book covers the architecture at three levels of detail. Chapters 1 and 20 are at the executive level; they mostly discuss technology trends, problems, and strategies. Chapter 4 provides a succinct technical overview of the architecture. Chapters 5 through 19 burrow into progressively greater detail about the SAA components. The lowest possible level of detail, concerning the mechanics of pro-

gramming, is absent from this book. The bibliography lists the dozens of IBM manuals that cover the programming details.

Our IBM consultant relations representative, Alfred B. Jackson, cheerfully furnished photographs of equipment. I thank IBM Corporation for providing the IBMLink on-line service, which was invaluable for researching and verifying facts. Conversations with Stephen L. Gray formed the basis for Chapter 1. Victoria Linnell deserves special recognition for drawing most of the pictures in the book and reviewing the manuscript many times. I am grateful for the patience and diligence of Amorette Pedersen in the production of the book. James A. Shields and Michael T. Wilcox contributed many improvements to the manuscript and helped me in many other ways.

Table of Contents

Chapter 1

IBM Strategies and Issues

This book is about a new standard for application programming, Systems Application Architecture (SAA), which IBM Corporation introduced in March of 1987. In the highly competitive computer industry, it is hard to find two people who agree on any particular topic, and many people are suspicious of new standards. IBM wants to "sell" the SAA standard to its customers and to others in the computer industry. The mission of this book is to explain the thinking behind the standard to a skeptical audience that wants an objective view.

IBM is the largest computer vendor in the world, with revenue five times that of its nearest rival. The company's strategies, both stated and implied, influence the entire computer industry, including customers and competitors. IBM's broad product lines meet a wide range of customer needs while competing with almost every other vendor's offerings.

IBM tries hard to explain its product strategies to its customers. This process is complicated by the sheer number of products offered and the technical nature of each product. Because most people in the computer industry are specialists, few people fully understand the "big picture" of the IBM world. Until IBM developed SAA, nothing explained how the pieces of the big picture fit together.

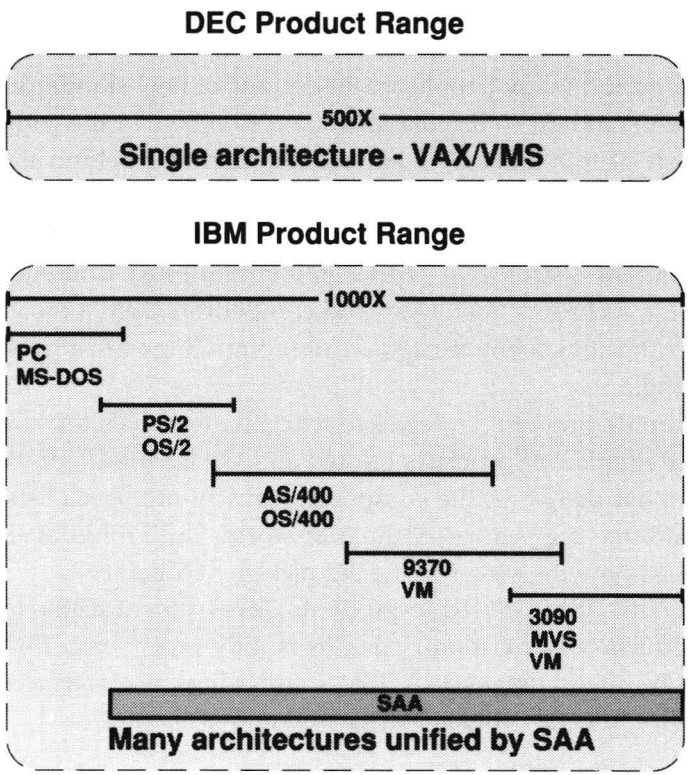
The pieces don't always fit. IBM needed to guide its development teams to build well-integrated products. Furthermore, customers buy many non-IBM products, which also must be integrated. In SAA, IBM standardizes the interfaces of many key products, solving several problems discussed in this chapter.

Single Versus Unified Architectures

In the 1970s, IBM could have decided to use the successful System/370 architecture in all its computers. Instead, the company offered several diverse computer architectures, each intended to serve different customer needs. This decision made it difficult for customers to switch from one computer family to another. Software developers had trouble building products that worked with multiple IBM computer families because each family was different.

While many IBM computer product lines were successful in the marketplace, others ultimately failed. Although IBM support sometimes seems everlasting, no company can support a failing product forever. Some notorious products, such as the 8100 information system, had unique architectures that many customers adopted. When IBM phased out these products, they became orphans. Customers had difficulty converting their applications to newer systems.

Figure 1-1: Single Versus Unified Architecture



IBM's archrival, Digital Equipment Corporation, took a different approach. Digital reasoned that a single architecture could be used in its entire product line, spanning the full spectrum of computing power. Introduced in the late 1970s, the VAX architecture propelled Digital to industry leader status. This architecture is comparable to, but not compatible with, IBM System/370 architecture.

The single architecture strategy worked well for Digital and other vendors such as Wang and Hewlett-Packard. Customers wondered whether IBM's motley collection of architectures would meet their long-term needs. Competitors argued that IBM's strategy was fragmented, leaving customers stuck with orphaned products. They pointed out several product lines, such as the 8100, where this had happened.

The single architecture strategy is flawed, but few vendors admit it. This strategy dictates putting all their eggs in one basket. How do vendors take advantage of the latest technology, which may not fit into their architecture? What happens when an architecture becomes obsolete? How do vendors tell customers it is obsolete? Some vendors who were using a single architecture failed, and others might fail in the future.

IBM could not easily switch to a single architecture strategy because many of its disparate product lines had been so successful. The company sold thousands of System/370, System/36, and Personal Computer products. Although System/38 wasn't a great sales success, customers liked the product. If IBM abandoned these architectures, customers might abandon IBM. Nevertheless, something had to be done.

IBM picked the most important architectures and unified them through a common user interface, programming interface, and communications. The new AS/400 computer family is the fusion of the old System/36 and System/38 products. The Personal System/2 followed the Personal Computer, and Enterprise Systems Architecture (ESA) added new functions to System/370. SAA pulls these hardware environments together under one umbrella. Although SAA doesn't mandate a single computer architecture, it is IBM's best response to its competitors.

Portability

Portability has been an issue ever since engineers built the world's second programmable computer. SAA is concerned with portability of three assets:

- Application programs
- Programmer skills
- User skills

Portability makes it easy to move these assets from one computer environment to another. If all SAA hardware had the same architecture, portability wouldn't be difficult. However, the three SAA hardware architectures are very different, so SAA's portability features must address many problems.

Many customers outgrew the capacity of their IBM systems and learned how difficult conversion could be. This was a special problem for System/36 customers, who often are small businesses. They could choose between two IBM product lines, but both required tricky conversions. System/370 offered plenty of growth potential, but had few application software products intended for small businesses and System/38 offered suitable software, but much less capacity than System/370. Customers who outgrew System/38 faced an arduous conversion to System/370.

SAA is supposed to cut software conversion effort when growth dictates moving to another hardware platform. At first, this was a primary selling point for the new architecture. After customers asked many pointed questions about software portability, IBM retreated. SAA won't provide full portability of application source code, so software writers must tailor their code to run on each platform, which might raise development cost by 10 to 20 percent. Programmers should avoid using unique features of each SAA hardware platform. This "least common denominator" approach may simplify application design, but it also may degrade performance.

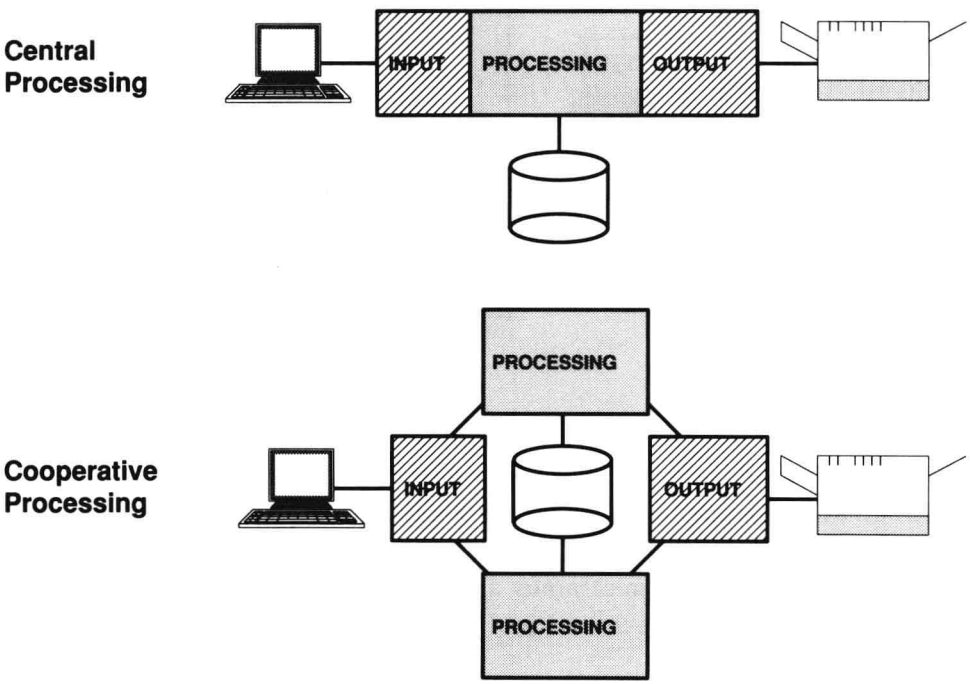
Each IBM computer architecture demands different sets of programmer skills. A System/38 programmer must be retrained to work on a System/370, which could take months and cost thousands of dollars. By providing a standard application programming interface, SAA lets developers focus on the logic of applications, rather than the details of the computer environment. Today, SAA falls short of isolating the programmer from each system's foibles. If IBM succeeds in its goal of programmer portability, the possible benefits are huge for both users and vendors.

User skill portability may be SAA's most important benefit. The standard user interface makes it easier to move users between computer environments. Real users of applications often are not computer professionals; computers tend to intimidate many people and resistance to change is a common problem. People make more errors during the "learning curve" for applications; such errors can be devastating or merely expensive. SAA's objective is to reduce the time spent retraining people to use new applications.

Cooperative Processing

In pure mainframe environments, nearly all application processing takes place in the central processors. Users are located at non-intelligent terminals, that are connected through communication links to the central processors. Although this approach has worked well for many years, it isn't always the least expensive or most efficient way to run applications.

Figure 1-2: Central Versus Cooperative Processing



Mainframe computers always have been expensive. Minicomputers were cheaper, based on the cost per computation, but they could not be used for many applications because they were not powerful enough. In the late 1970s, IBM and other vendors tried to divide the processing between mainframes and minicomputers. This effort, called *distributed processing*, worked well for some applica-