

**SUN WARS:
COMPETITION WITHIN A MODULAR CLUSTER, 1985-1990**

BY

CARLISS Y. BALDWIN AND KIM B. CLARK

Acknowledgments:

We thank Jack Soll, Ellen Stein, Todd Pulvino, Partha Mohandas and Brent Omdahl for invaluable assistance in preparing this paper. The financial support of the Division of Research of the Harvard Business School is gratefully acknowledged. Mistakes, errors, or omissions are ours alone.

© 1995 by Carliss Y Baldwin and Kim B. Clark.

The views expressed in this paper are preliminary conclusions subject to revision and correction. This paper is not to be copied, circulated, or quoted without written permission of the authors.

July 15, 2003

Sun Wars: Competition within a Modular Cluster, 1985-1990

1. Introduction: The Emergence of a Modular Cluster

Computers have been designed as modular systems with interoperable components since the advent of IBM's System/360 in 1964.¹ A modular design separates the design parameters of particular product into visible and hidden subsets. Visible design parameters are embodied in the system architecture and its interfaces – designers must take account of this information when designing individual components that function within the system. Hidden design parameters, in contrast, are "buried" in components. The designers of other parts of a system do not need to know in detail how a single component works so long as it conforms to system interface specifications.² As a corollary, hidden parts of a system can be changed to optimize their performance without compromising the system as a whole. Thus experimentation and improvement can proceed at the component level while the overall system architecture and interface designs remain intact.³

In the decades following the introduction of the System/360, the modular design of computers gave rise to a modular cluster of companies. In the early years (1966-1980), most new entrants to the cluster competed on the basis of hidden design information. That is, they offered superior hardware and software products designed to function within another vendor's system. From this era of plug-compatible peripherals comes the quote: "IBM is not the competition, it is the environment."

¹SPREAD Report; Ferguson and Morris, 1993

²Of course, no modularization is ever perfect. To partition and decouple design tasks requires knowledge, and thus modularization typically proceeds by stages as knowledge accumulates and is applied to system design. Some systems are technically easier to modularize than others: systems based on electricity, for example, involve a one-dimensional flow of electrons, and seem easier to break apart than mechanical systems (a car or an airplane) that contain two and three dimensional surfaces and must have structural integrity as well.

³Alexander, 1964; Parnas, 1972a, 1972b; Baldwin and Clark, 1993.

However, as fundamental technical knowledge expanded, it became possible to reconfigure known elements to make new systems. The computer workstation, for example, represented a fundamentally different conceptualization of the product and its relationship to the user. While it was based on much of the same componentry – electronic circuits, circuit boards, semiconductors, disk and tape drives, keyboards, and monitors – as traditional mainframes or minicomputers, it was smaller (and less expensive) than its predecessors, and it organized the elements of the system in a different way.

Such "architectural" innovations created new patterns of use for computers and, consequently, new markets. As a result of this evolutionary process, by the mid-1980s competition in the computer industry was essentially among "platforms."⁴ Then a new set of rivalries emerged, which set the stage for a "collision" between two contrasting paradigms of firm behavior.⁵ One pattern -- which we term the "mainframe paradigm" – was initially invented by IBM and was a mode of behavior appropriate to the early stages of a modular cluster's formation. Many other computer companies, including DEC, Data General and Prime, achieved great success by adopting this paradigm. But as the cluster of firms participating in the industry grew larger and denser, another pattern – which we call the "modular paradigm" – came into play. The purpose of this paper is to describe how these two paradigms collided in the competition between Sun and Apollo between 1985 and 1990.

The plan of the paper is as follows. Section 2 describes the two opposing paradigms. Section 3 describes the competition between Sun and Apollo between 1985 and 1987. Section 4 explains the rationale behind the Sun-AT&T technical alliance and describes the formation of the Open Software Foundation (the competitive response to

⁴Bresnahan and Greenstein, 1992

⁵Collisions in rapidly changing technologies and markets are especially interesting because so much happens so fast. What might have taken decades to work out in an industry such as steel happened in a few years in the computer industry.

the Sun-AT&T joint initiative) and its effect on AT&T's commitment to Sun. Section 5 concludes.

2. Exploiting Modularity: Two Paradigms

2.1 Modularity and Its Consequences

The workstation environment of the mid-1980s had its roots in the early 1960s, when IBM first tackled the problem of compatibility among computer systems. The challenge was to design a family of computers that spanned a wide range of size, cost, and operating performance parameters and yet could run the same programs and use the same printers and disk drives. The key to solving the challenge was modularity, a technical principle embodied in IBM's System/360 (a family of computer systems first conceived in 1961 and brought to market in 1965).⁶

Modularity is a principle of design in which a complex system is broken down into independent modules that are linked together through standard interfaces. The antithesis of a modular product is one in which every component – down to the smallest screw – is unique and can function in only one context. There are in fact several different types of modularity: a product may be *modular-in-production*, *modular-in-design*, or *modular-in-use*. Modularity-in-production rationalizes a product into components and allows parts to be standardized (e.g., all screws the same size) and produced independently before assembly into the final system. Modularity-in-design goes a step further: with an overall architecture and standard interfaces, the modules can be designed independently, and mixed and matched to create a complete system.

⁶The 360 achieved upward, downward, peripheral, and intertemporal compatibility, but it was a major engineering gamble at the time. Few in the industry thought it could be done. Notwithstanding the nay-sayers, the concepts of information hiding and interoperability of parts were rapidly developing in the technical literature of that time. If IBM's implementation of the 360 had failed, the same technical goals would almost certainly have been pursued at other companies, but the evolution of the computer industry in the 1960s and 1970s would have been different. (SPREAD Report; Pugh et al., 1991)

Finally, a product is modular-in-use if consumers themselves can mix and match components to arrive at a functioning whole.

The benefits of modularity in production are scale and scope economies, and the reduction in complexity of manufacturing and materials handling systems. The benefits of modularity in design are mix and match flexibility and efficient innovation, arising from the fact that all parts do not have to be redone to improve one element. The benefits of modularity in use are the ability to achieve customized designs plus ease of maintenance, reliability, and piecemeal upgradability.

For IBM, which in 1961 was faced with an overwhelming proliferation of incompatible computer models, the flexibility inherent in a modular product line held tremendous potential. Realizing that potential, however, required a strategy for marketing, operations, pricing, and finance that capitalized on the power inherent in the modular concept. In short, IBM needed a technical, financial and competitive paradigm suited to a modular family of computers.

The "mainframe paradigm" IBM adopted was not, as we shall see, the only paradigm it could have used to exploit modularity, but it did prove highly effective. It solved a fundamental problem inherent in all modular designs: how to capture (at least some of) the value generated by the underlying efficiency of modularity. In the short run, the originator of a modular design will be able to make the product more cheaply and with greater variety than is possible for those relying on less modular designs. But in the long run, an "architect firm" will have trouble earning superior returns. When the basic design is modular, there are simply too many ways for suppliers to link up with one another and with users without paying the architect's "fee."

IBM solved the problem (at least for a while) by using its traditional strengths in sales and operations, and by developing new capabilities in software, finance, and

component production.⁷ These initiatives together became the core of the mainframe paradigm.

2.2 The Mainframe Paradigm

Table 2.1 summarizes the main elements of this paradigm. Modularity is at its heart. The central processor, memory, printers, tape drives, and terminals that made up the product system were all modular in design and use. Furthermore, the manufacturing arm of the company had a long tradition of using common parts and of encapsulating components in hierarchical subassemblies. Such modularity in production proved invaluable as IBM began to cope with rapid innovation and cost reductions in electronic components and hardware.

⁷The paradigm that evolved around the 360 was deeply rooted in IBM's history. In its traditional tabulator business, IBM had developed a system of engineering design, finance, and sales that gave users the options they so valued while preserving their investments in processing methods and data. The system was based on modular, interoperable pieces of equipment that processed a universal, standard type of data (80-column punched cards). IBM also developed a contractual system that effectively limited users' options to those provided by IBM. The key elements of this "nexus of contracts" were: (1) IBM manufactured all physical parts of the system; (2) the IBM sales force was in constant contact with the customer's establishment; (3) IBM salesmen were technically competent and motivated to solve customer problems by incrementally reconfiguring their systems; (4) IBM owned all equipment and merely rented it to users.

Table 2.1: The Mainframe Paradigm

Key Elements	Impact
1. Compatible family of computer systems, modular in design and use, covering a wide product range	Modular upgradability; mix and match flexibility; one stop shopping
2. In-house manufacturing of components, peripherals, and systems; modular in production	Control over technical change and quality
3. Proprietary software (operating systems, compilers) and interfaces	Tie users to platform; preempt entry by producers of modules
4. Direct sales force, technically competent, focused on solving problems	Strong, enduring customer relationships; deep knowledge of customer's system
5. Own the equipment, rent it to customers	Facilitate product innovation; control over customer configuration of system
6. Preemptive cannibalization	Users get system upgrades as technology changes; preempts competitors
7. Conservative financial strategy	Capacity to cope with the unexpected; arsenal to withstand competitive attack

Although IBM's hardware systems were highly modular, its software was not. IBM's mainframes were designed around a monolithic core of proprietary software consisting of the instruction set, the operating system, and compilers that translated high level languages (like FORTRAN or COBOL) into machine-readable code. Thus whereas hardware improvements could be incorporated into the system rapidly and economically, instruction set and operating system changes and software conversions were extremely difficult and expensive. The opacity of the software core (and of most applications programs developed in this era) had an important economic effect: it locked users in to their existing systems and software.

In the mainframe paradigm, therefore, compatibility and hardware modularity created valuable user options, but once a user committed to a particular family of machines, a complex instruction set and operating system made it expensive to switch to a different platform. Moreover, IBM's excellent service and support and the option to upgrade piecemeal over time or reconfigure to meet changing needs made such a

switch unnecessary. Thus the mainframe paradigm tied the customer happily – but tightly – to IBM.

The paradigm also worked to deter competition on individual modules. In particular, selective pricing and product introductions were an effective disciplinary device and a deterrent to prospective entrants. Proprietary and secret interfaces made it technically difficult (and perhaps illegal) to attach a new module to the system. Finally, a conservative financial strategy gave IBM the wherewithal to cope with unexpected technical developments or competitive moves. In general, a firm's capacity to withstand competitive attack can be measured in terms of its financial slack. High internal cash flow, high cash balances, and low debt all signal potential competitors: "Don't tread on me!"

Over the past forty years, most firms that succeeded as system designers and manufacturers in the computer industry did so by adopting the mainframe paradigm.⁸ Firms such as DEC, HP, Fujitsu, and NEC combined modular hardware with monolithic software; they offered relatively extensive product lines and maintained proprietary interfaces; they practiced selective pricing⁹ and controlled cannibalization; and most have maintained a high level of financial slack. Partly to exercise control over their own systems and partly as an outlet for their high internal cash flow, these firms have tended to incorporate more components and interfaces within their own boundaries over time. With some modifications, this was the model Apollo Computer adopted as it pioneered in the workstation industry.

⁸Perhaps the most successful application of the paradigm outside IBM was accomplished, ironically, by DEC. DEC got its start in the business producing electronic modules and for many years defined itself as the antithesis of everything that was IBM. Yet in the late 1970s, with its VAX family of computers, DEC adopted the mainframe paradigm with great success. Its main competitors – Prime, Data General, and, to a lesser extent, Hewlett Packard – also pursued market success through modular designs, proprietary software, a broad line, direct sales, and preemptive cannibalization.

⁹Within the constraints established by antitrust law.

2.3 The Modular Paradigm: Choosing Where to Compete

A company operating under the mainframe paradigm expands outward, aiming to make enough parts proprietary to gain monopoly power over some group of users. Through vertical integration, the outward-bound innovator seeks to become "the environment" for a captive group. A quite different paradigm for competing as "the architect" in a modular cluster is to internalize the narrowest possible set of design decisions. If the cluster of firms providing design and production services to the architect is dense, then a system architect may reap significant performance gains by exercising its options to select the best implementation of each component of the system.

Table 2.2: The Modular Paradigm

Key Elements	Impact
1. Compatible family of computer systems, modular in design and use, covering a wide product range	Modular upgradability; mix and match flexibility
2. Little in-house manufacturing; reliance on a network of specialized suppliers	Need for dense network of specialists and well defined interfaces
3. Modular software, standard interfaces	Third-party software crucial; alliances on standards
4. Sales through resellers and distributors; some direct for large accounts	Reseller relationships crucial; need for other channels for customer information; alliances key
5. Sell, don't lease, equipment	Fosters customer creativity in system configuration (see #4)
6. Preemptive cannibalization	Push envelope on performance; preempt competitors
7. Aggressive financial strategy	Rely on unconventional sources and instruments; alliances, venture capital crucial
8. Speed in technical development, market entry, and growth	Fast and flexible systems; keen sense of opportunity and timing
9. Technical leverage through highly focused knowledge of system, interfaces, and hidden/visible information	Premium on gurus with specialized skills in architecture and interfaces; learning crucial

The modular paradigm laid out in Table 2.2 is virtually the polar opposite of the mainframe paradigm. While it is true that design modularity, compatibility, and

cannibalization are features of both paradigms, these ends are pursued by very different means. Under the modular paradigm, a system architect seeks to use divisible rather than monolithic software and standard rather than proprietary interfaces. Architects do not attempt to make everything under one roof; they assemble and test systems made of components supplied by specialist firms; they rely on resellers and distributors for sales; they sell, not lease, equipment; and they may pursue a more aggressive financial strategy.

In addition, two other elements – technical leverage and speed – play a critical role in the success of the modular paradigm. To see why this is so and how the elements of the paradigm fit together, consider the pressures the system architect faces in pursuing the modular strategy. To achieve high profitability without a significant degree of vertical integration, the modular architect's design must be attractive enough to convince suppliers of complementary parts of the system to invest in making components compatible with the new architecture.¹⁰ But to earn superior returns, the architectural innovator also must control some critical element of the overall system – a key module, subsystem, or interface.

To solve the problems of attraction and profitability, the architect firm must seek to create technical leverage. That is, it must develop knowledge of a few key elements of the design, and through these, it must substantially affect the performance the system as a whole. Such knowledge and insight into systemic behavior does not come cheap.

¹⁰There is an inherent conflict between the goal of acceptance and the goal of dominance within a cluster. The suppliers of complementary components do not want a disproportionate share of profits to flow to another firm; each would like to dominate the cluster itself (or, failing that, prevent anyone else from doing so). Thus one of the keys to having one's design accepted is to appear unthreatening. But it is also useful to be attractive: complementary suppliers would like to be part of a winning team. Acceptance of the architect's design thus rests on its attractiveness and the absence of threats to capture a disproportionate share of the rents. Operationally, this means that acceptance and dominance cannot be pursued openly at the same time; acceptance must be nailed down before dominance can be asserted. Intertemporal inconsistency is one of the things that makes the modular paradigm complex and hard to implement.

But even with such knowledge and insight, control over the critical design parameters is likely to be tenuous and short-lived. In a modular cluster there will be incessant pressure to find more powerful or lower cost solutions. Similarly, an architecture that succeeds in improving system performance will inspire relentless efforts to remodularize – that is, to carve the pieces up in different ways so that what was proprietary and hidden will be public and visible.¹¹

In sum, an architect firm operating in modular cluster can never slow down -- it must transform new knowledge into a sequence of new designs over time. With an ongoing stream of designs embodying highly leveraged insight and knowledge, the architect can push the envelope of performance and create momentum in the market and in the court of cluster opinion. By doing so, for a while at least, it may earn superior returns.

This is where speed becomes critical. The time it takes competitors to copy a design or equal its performance gives the architect a window of opportunity. In fast moving markets, however, competition is intense and windows close quickly – within 18-24 months in the workstation industry, for example. To succeed over the longer term the architect must move quickly to open successive windows with new, ever better designs built on leveraged technology. In effect, pursuing the modular paradigm is like riding a tiger – you've got to stay on or risk the consequences.

In summary, the mainframe and modular paradigms are business models that can be applied to products that are modular in design and use. In the mid 1980s, these two paradigms came into direct competition in the computer workstation market. Apollo adopted the mainframe paradigm, a natural step given the company's roots in the world of minicomputers and DEC's success. Sun, on the other hand, pursued the

¹¹When remodularization occurs (as when Compaq cloned the IBM PC), the hidden interior of the former interface becomes a component, and hence prey to competition from alternate versions that conform to the now-separate interface definitions. In this fashion a seemingly secure competitive advantage may evaporate very quickly.

modular paradigm, which emphasized standards, UNIX, and a dense network of component suppliers in Silicon Valley. The competition between Sun and Apollo is a natural episode in which to examine the two paradigms, their relationship, and their relative success.

3. Sun and Apollo

3.1 Similarities

Much has been written about the different strategies pursued by Sun and Apollo as they battled in the workstation market in the mid-1980s. In reading contemporary descriptions (particularly prospectuses), however, one is struck by how *similar* the two companies were. Both emerged out of the same technical, business, and financial environment.

From inception, Apollo and Sun pursued the same opportunity. Developments in semiconductor memory, microprocessors, disk drives, and operating systems had created the possibility of putting significant computer power – including eye popping graphics and hyperfast computation – on an engineer's desk. Moreover, with the emergence of low cost networking technology, it was possible to link the desktop machines (soon called workstations) into a network where engineers could share memory and peripherals and communicate rapidly and effectively – all at a fraction of the cost of a competitive minicomputer system.

Though Apollo was the leader in recognizing the opportunity, it was soon followed by a host of competitors, including Sun Microsystems. Both Apollo and Sun operated within a dense modular cluster. They each based their products on Motorola microprocessors. As one Sun engineer noted, they were equally hostage to Motorola's design and development cycle: if Motorola introduced a new, higher capacity chip, Apollo and Sun had no choice but to design new systems around it. Both companies also sourced peripheral equipment (disk drives, printers, file servers, and

monitors) from outside suppliers. Although there is some indication that Apollo did more to tailor peripherals (especially monitors) to its own operating system, neither company was eager to integrate backward into components or peripheral manufacturing. Their prospectuses contained the following statements (about manufacturing):

The Company intends to continue to buy standard parts and components (rather than to integrate vertically its manufacturing operations) because management believes that in this way the Company may more quickly integrate advances in technology into Apollo products and take advantage of the research and development efforts of component manufacturers. (Apollo, 3/4/83 and 2/7/86)

The Company intends to continue to purchase standard parts and components because it believes that this practice enables the Company to integrate advances in technology into its products more quickly and to take advantage of the cost reduction and research and development efforts of the component manufacturers. (Sun, 3/19/86)

The dynamics of a modular cluster were so taken for granted in the computer industry by the mid-1980s that the logic of constant improvement and cost reduction had found its way into the standard boilerplate of stock offering documents.

On the consumer side, both Apollo and Sun depended on third-party developers to write specialized application software. Their initial public offering prospectuses each

contained a list of applications packages operable on their machines.¹² Moreover large portions of the sales of both companies were to so-called "system builders" or OEMs. OEMs purchased systems (sometimes at the board level), added applications software and specialized equipment, and resold the systems (often under their own names) to end users. System builders were thus another link in the chain that stretched from components to end users.

The existence of this large set of system builders was further evidence of the density of the modular cluster Sun and Apollo both inhabited. As providers of complementary goods, the system builders, component and peripheral manufacturers, and software developers all wanted "their" workstation manufacturer to succeed, but not too well. Each member of the cluster sought to garner some share of a stable stream of rents for itself. To the extent that the relevant hardware and software interfaces were not perfect (hence there were costs of changing vendors or finding new customers), the members of the cluster depended on one another for reliable inputs or a stable demand for output. But firms making modules (whether hardware or software) were vulnerable to a reduction in rents if the provider of a critical, complementary piece of the system grew too strong. Therefore each was always looking to identify alternate sources of critical inputs and to diversify its customer base. Each was also looking for ways to become the sole supplier of one or more critical parts of the system as a whole.

Thus, in terms of their general position within the greater cluster, Apollo and Sun were very similar. Both designed and manufactured workstations and had no plans to integrate backwards into components or peripherals. Both sold large quantities to

¹²Interestingly, Apollo's 1986 convertible subordinated debenture prospectus did not contain such a list – perhaps because enough developers had dropped Apollo that publishing an updated list would have been embarrassing to the company. Sun's IPO prospectus, published one month later, listed available applications, large OEMs, and end users by name, indicating that such information was competitively relevant.

system builders: their products were components of still larger systems. Both were actively involved in the development of system software but relied on third-party developers to write specialized applications software. Both gave at least lip service to the need for standards and promised to conform with industry standards where they were relevant.

3.2 Apollo's Product Definition and Design

In spite of these similarities, Apollo and Sun exhibited distinctly different approaches to product definition and design. The differences were most evident in the areas of network design, the operating system, and modularity. Apollo presented users with an indivisible bundle consisting of (1) a network management system; (2) a single machine operating system; and (3) hardware. The first element was implemented through their proprietary DOMAIN network architecture; the second through their proprietary Aegis operating system software (although by 1986 they were supporting both Aegis and an internal version of UNIX); and the third through their workstations and associated peripherals, controllers, and storage devices. Each element required the other two in order to function; the DOMAIN architecture was inextricably tied up with the DN series of workstations, which in turn functioned best under the Aegis operating system.

This intertwining of network architecture, operating system software, and computer hardware was characteristic of the technical and commercial milieu from which Apollo emerged. Essentially all of Apollo's senior executives came from the minicomputer industry, and their explicit strategy was to exploit workstation technology to compete directly against the dominant players in that world: IBM, DEC, and HP. The established manufacturers not only served as targets, but also provided the paradigm for design, management, and competition. Edward Zander, Apollo's marketing vice president at the time, aptly summed up Apollo's position:

The competition I worry about is IBM, DEC, Hewlett Packard. The startups aren't a threat to our existence – which is not to say we don't look over our shoulder to see what technology they have. IBM has taught us over the last 20-30 years that the whiz-bang product isn't always the winner. What counts is having the best price and performance at the right time, with the sales and service to back it up, and a good understanding of applications and end users.¹³

The design of Apollo's workstations reflected its roots and primary targets: minicomputers. The new workstation technology – 16- and 32-bit microprocessors; high speed, low cost memory; small, high capacity disk drives – gave Apollo an inherent cost advantage over minis, but Apollo's product was no "bare-bones" machine. Indeed, Apollo's design philosophy was to offer a high level of functionality, elegance, and quality. The Aegis operating system, for example, was chosen over UNIX (a choice actively debated within Apollo) because it offered significant advantages in networking functionality. Aegis managed network resources so that any user could transparently access memory and processing power on other nodes and servers, which meant that a single user could run bigger programs faster than would be possible with a single workstation.

In addition to its proprietary network and operating systems, Apollo emphasized advanced graphics capability and high speed number crunching in a compact, desktop package with only a few printed circuit boards (typically 3-4). While Apollo did no component manufacturing, it built a state-of-the-art manufacturing facility to handle automated component insertion, wave soldering, assembly, and testing.

The key minicomputer manufacturers in Apollo's genealogy – Prime, Data General, and DEC – approached computer design in a similar fashion. IBM (except in

¹³*Electronic Business*, 1 April 1984, p. 79. Zander later left Apollo to join Sun.

the case of the PC) and DEC carried design integration even further – back to the level of chips. Many designers believed that this high degree of specialization and integration among the core elements of the machine was necessary to achieve high levels of processing speed and efficient management of memory and the CPU.

Moreover, proprietary operating systems and networking technology created user lock-in – a loyal base of customers guaranteed a continuing stream of rents. Apollo's focus on its proprietary operating systems was hailed in the financial community. The following comment by Peter Labe, an analyst with Barney Harris Upham & Co., was typical of Wall Street's view of Apollo in 1984:

Its proprietary operating system will keep users with Apollo. The company has responded to UNIX demand by running versions of UNIX as a subset. Users will get more performance with Apollo's operating system and it keeps it from being a commodity product. Some users prefer UNIX, and that segment Apollo won't get. But I don't see any problem; the market will grow 50% a year the next few years.

Thus Apollo followed all the major precepts of the mainframe paradigm. The company combined modular hardware with proprietary, monolithic software, and gave users options in the form of an extensive product line of compatible machines connected by proprietary interfaces. Apollo counted on users becoming locked in to their platforms. Finally, it charged high prices (relative to its costs), practiced controlled cannibalization, and tried to maintain high levels of financial slack. In 1984 Apollo had 60% of a market that was growing at 50% per year. By the conventional wisdom of the mainframe paradigm, its strategy was impeccable.

3.3 Sun's Product Definition and Design

It was Apollo's bad luck to be in a head-on collision with a different paradigm. The founders of Sun Microsystems viewed the market opportunity from a very different vantage point: their goal was to get a powerful but low cost workstation to market quickly. Sun pursued and – through the agency of Andreas Bechtolsheim and William Joy – managed to achieve two historically conflicting objectives of computer design: 1) low cost through standard off-the-shelf parts; and 2) superior speed and performance. Sun did not simply build a computer based on standards; it built a fast, high-performance computer based on standards. To do so, the hardware and software architects and designers at Sun reconceptualized and remodularized the design of the computer.

To achieve their low cost objective, Sun's designers adopted two principles: standard parts and a single board. Even at the outset of their work, Sun's architects tapped into a dense modular cluster of component suppliers and thus benefited from the large volumes, R&D, and expertise of very experienced electronics companies. Standard components, including a Motorola microprocessor and an Ethernet networking chip (not to mention power supplies, disk drives, monitors, connectors, and capacitors), were widely available, relatively inexpensive, and required little development investment. The single board design was chosen to simplify logistics, materials management, testing, and manufacturing, and to eliminate the cost of designing and producing multiple boards.

The pursuit of standard solutions had the virtue of minimizing up front investment and eliminating delays associated with developing proprietary components. Sun applied the same logic to choosing UNIX as the operating system. Bechtolsheim had used UNIX in developing a Sun workstation prototype; given the founders' strategy of simplicity and standardization, it made little sense to invest in a costly, time consuming development effort to develop a proprietary operating system.

Designing a computer with standard parts and a single board was one thing. Making it fast and powerful in order to appeal to a technical community was quite another. To get speed and power out of standard parts, Sun's architects built on recent research in computer science to modularize the design. In the spirit of the modular paradigm, they searched for a narrow set of design parameters they could control (all the rest would be standard). The set they hit upon was the interface between the central processing unit (CPU) and internal memory.

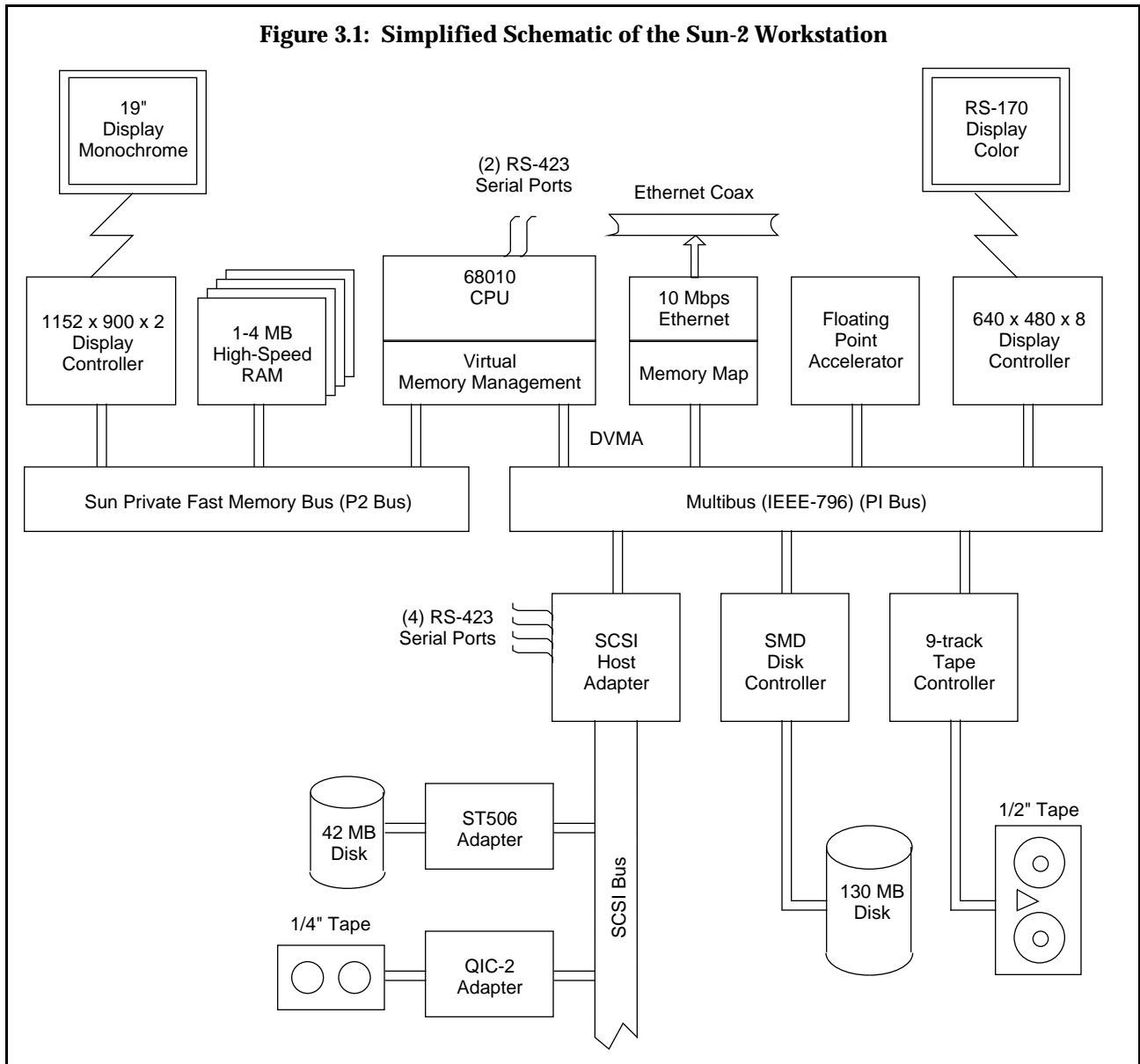
In the 1970s, John Hennessy and his colleagues at Stanford (where Andreas Bechtolsheim was a graduate student) helped to pioneer research on the frequency with which instructions in different computer architectures were actually used in running programs. Hennessy's work, along with work by Cocke at IBM and Patterson at Berkeley, led to the introduction of commercial RISC (reduced instruction set computer) architectures in the 1980s. Hennessy established the principle that instructions or operations used frequently should be very fast.¹⁴ In the kinds of applications Sun had targeted (e.g., computer-aided design and software engineering), implementing that principle meant running the CPU at maximum speed all the time. This in turn meant making operations to put things into and take things out of internal memory as fast as possible.

Sun developed two proprietary hardware components to couple the CPU and internal memory. The first was direct memory for the CPU, called the "memory management unit" (MMU). Sun patented a special "no wait state" MMU that eliminated situations where the CPU had to wait to access memory because memory was not available.¹⁵ The second proprietary component was a high speed 32-bit internal memory bus. Unlike its competitors, Sun put its internal memory chips (1-4 MB of

¹⁴This principle is sometimes referred to as "Amdahl's Law."

¹⁵Note the need for intellectual property protection for what was a fairly simple and visible change in the design.

DRAM) and the video controller chips for its bit-mapped graphics display on a proprietary, high speed internal bus. All other input-output operations used a standard Multibus setup (see Figure 3.1 for details).



The internal bus and the MMU were crucial not only in memory-CPU coupling, but also to the single board design. The bus, for example, allowed Sun's designers to closely link memory and video chips and locate them on the same board. The MMU allowed Sun to avoid using cache memory (a special set of very fast memory chips designed to eliminate wait states) and thus reduce chip cost and eliminate an additional

board. By effectively managing the CPU-memory connection, Sun achieved a no wait state design on a single board.

Software – particularly Sun's in-depth knowledge of UNIX – also played an important role. William Joy, the principal architect of the Berkeley version of UNIX, who joined Sun shortly after its founding, was a recognized master at manipulating UNIX to run efficiently on a variety of different hardware platforms. Joy and the other UNIX "kernel designers" at Sun formed a critical mass of UNIX hackers who not only understood the operating system, but could advise the hardware designers on how to take advantage of the software. When Sun encountered glitches as a result of using standard components, for example, the UNIX hackers simply wrote code to get around the problem. Furthermore, Sun exploited some of UNIX's idiosyncrasies to enhance the machine's speed. For example, the operation of the no wait state MMU fit perfectly with UNIX's conventions for establishing the hierarchy of operations (e.g., determining which address gets accessed first): the MMU produced exactly what UNIX wanted, when it wanted it. The result was a meshing of hardware and software critical to overall performance.

3.4 The Impact of Remodularization on Competitive Performance

Sun's approach to computer design produced a fast, high performance, surprisingly affordable workstation. Moreover, it was a product architecture in which traditional sources of dominance in the modular cluster (the operating system, high profile components, critical applications software) were either standard products themselves, or based on standards that allowed others to hook into the architecture and reap the rewards outside of Sun's control. Indeed, in the area of software, Sun went out of its way to make it easy for third parties to develop new products for the Sun platform. A crucial step was the introduction of a suite of powerful software tools that hid details of the hardware and eliminated much of the drudgery in software

development. While writing for Sun meant writing for UNIX-based products more generally, having great tools meant that developers had an incentive to write for Sun first.

Reconceptualization and remodularization gave Sun three important competitive advantages. First, Sun workstations were capable of running a full-fledged version of UNIX – a significant selling point to an important group of users.¹⁶ Apollo offered a version of UNIX on its workstations from the outset (see the 1983 prospectus), but its systems did not run well under UNIX until much later, and not all features of UNIX were offered. Second, because they ran UNIX, Sun workstations could participate in networks of heterogeneous machines. This meant that users could link Sun machines into their existing DEC minicomputer networks at low cost.¹⁷ Therefore, in stark contrast to a fundamental tenet of the mainframe paradigm, purchasing Sun workstations did not entail an irreversible commitment to Sun's product line. These two advantages were important selling points for Sun.

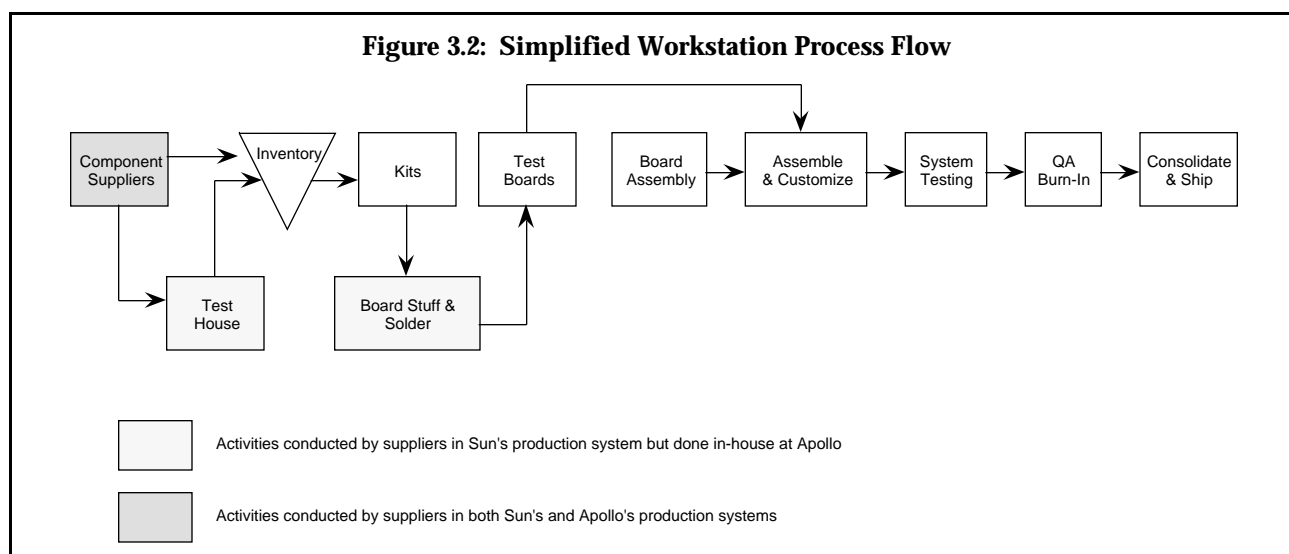
The third advantage was often overlooked by the trade press and investment analysts. Sun's remodularization of workstation design and its pursuit of the modularity paradigm in production as well as engineering design greatly increased its efficiency and reduced its capital requirements. Thus a dollar of funding at Sun went approximately twice as far as a dollar at Apollo. In the long run, Sun's efficiency was as important to its success as its espousal of open systems or its implementation of UNIX on a desktop machine.¹⁸

¹⁶One of UNIX's advantages seems to have been its high degree of modularity relative to other operating systems, which made it somewhat portable across hardware platforms. Also very important was the fact that new features could be added to UNIX very easily without compromising the basic system (this is typical of a modular design). Thus UNIX itself came to have a cluster-type of structure, with many small modifications and improvements being made everywhere with little central control.

¹⁷Most UNIX-based networks were implemented on DEC hardware.

¹⁸The appeal of open systems and UNIX is explained by complementary asset externalities typical in networks. But existing network theories do not explain how a different design approach can yield a dramatic advantage in terms of overall efficiency. An understanding of the efficiency gains from modularity is needed to complete the theory.

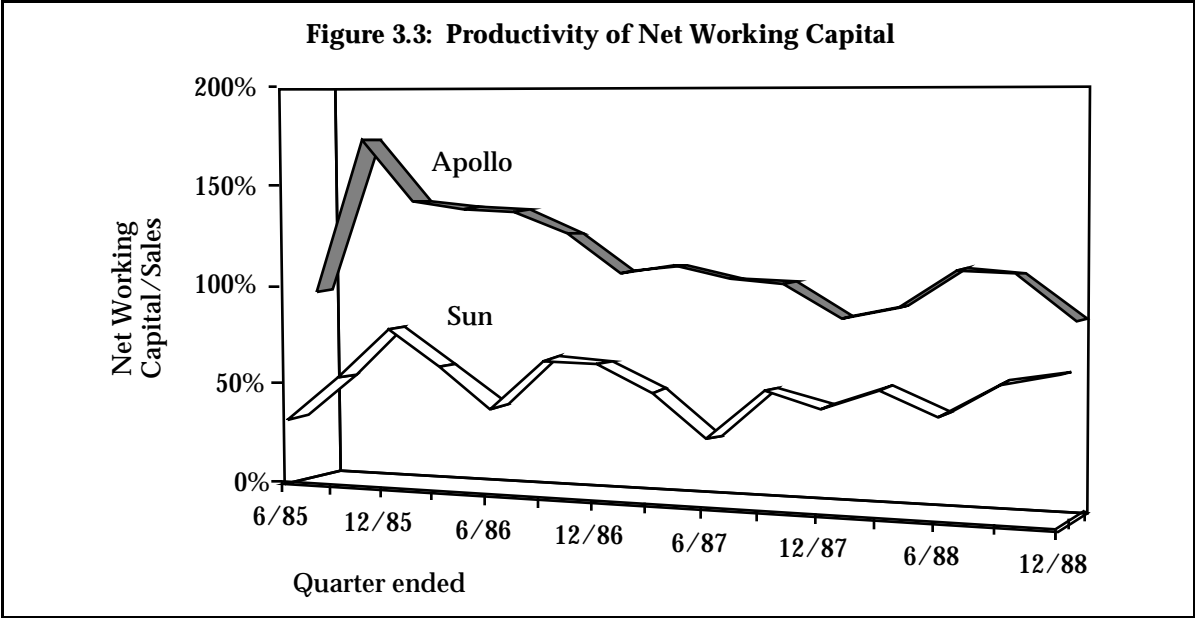
Part of Sun's efficiency advantage was rooted in its strategy of relying on the dense modular cluster of suppliers for almost every aspect of manufacturing. Capitalizing on its reliance on standard components, Sun did even less in-house manufacturing than Apollo (see Figure 3.2 for a comparison). The single board design accentuated that difference. With a single board, Sun greatly simplified the materials flow it had to manage, reduced the amount of work-in-process, made testing boards and systems much easier, and eliminated the need for associated overhead and facilities. The impact was to substantially increase Sun's capital efficiency relative to Apollo.



For example, Figure 3.3 shows that from mid-1985 (the earliest period for which comparative figures are available) through the end of 1988, Sun's net working capital¹⁹ averaged about 50% of quarterly sales, while Apollo's was never less than 100%. A detailed breakdown of the components of working capital reveals that the main difference was in inventory turnover: Sun produced and shipped its product in about half as much time as Apollo. Since the evidence suggests that Sun's delivery

¹⁹For purposes of the comparison, we defined net working capital as current assets excluding cash minus current liabilities excluding short-term debt. Levels of cash and short-term debt reflect the firms' strategic and financial choices, and are not indicative of the operating efficiency of these businesses. Detailed financial comparisons are available on request from the authors. See also Baldwin and Soll (1990).

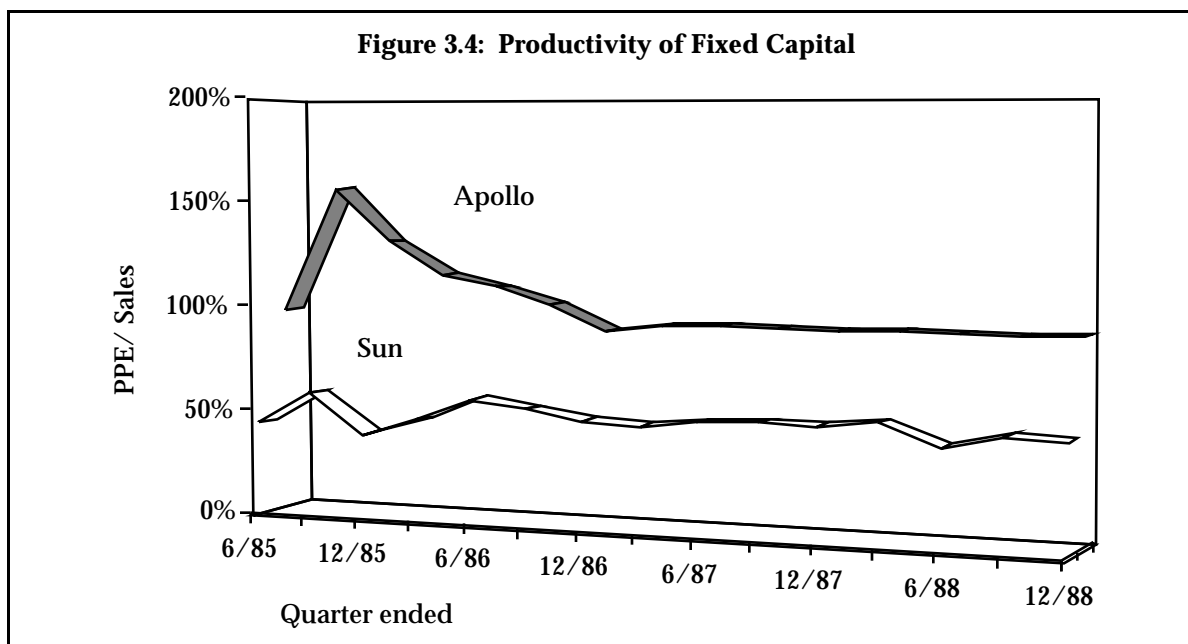
performance was equivalent to Apollo's, this difference in throughput time underscores the power of the modular paradigm. Indeed, during this time Sun built "to order" while Apollo built "to stock." Apollo's intent was to optimize production flow at the new, highly efficient factory it brought on line in 1985. But, interestingly, Sun suffered no cost disadvantage from its "inefficient" production method: the companies' gross margins were almost equal.²⁰



Turning to fixed assets in Figure 3.4, Sun's property, plant and equipment never exceeded 60% of quarterly sales, while Apollo's peaked at 157% of quarterly sales in the third quarter of 1985 and declined to 95% of sales by the fourth quarter of 1988. Part of the explanation here may be due to Apollo's new plant. When lumpy capacity is brought onstream, there is often a period of excess capacity with correspondingly low productivity. In fact, some of Apollo's lower productivity was caused by being at less than full capacity from 1985 through the end of 1987. But the *increases* in property, plant and equipment at Apollo from June 1985 to March 1987 amounted to 89% of

²⁰Sun's net income over sales was higher than Apollo's because of Apollo's higher depreciation charges.

incremental sales, while increases at Sun were only 53% of incremental sales.²¹ Thus, even allowing for the underutilization of capacity, the figures clearly show that Sun had a system which allowed it to produce computers with substantially less fixed capital than Apollo.



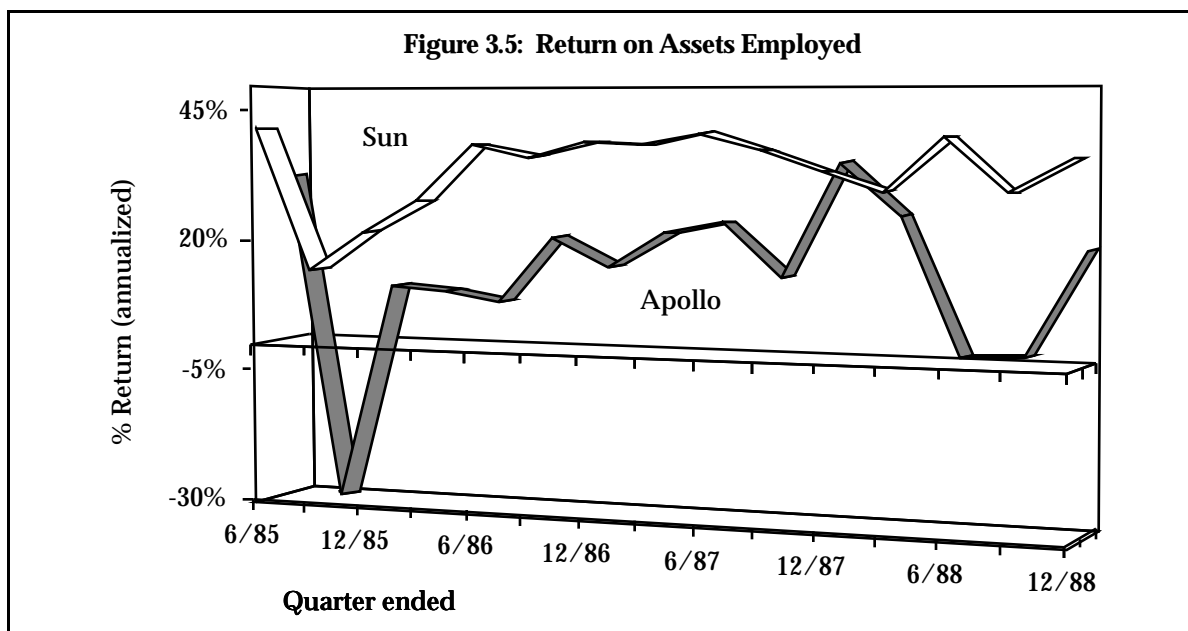
What Sun's efficiency meant in financial terms is summarized in Figure 3.5, which plots the return on assets employed for the two companies. The numerator of each series equals internally generated free cash flow;²² the denominator equals the sum of net working capital, fixed, and other assets. The ratio is a rate of return on the operating assets in each business. (Quarterly rates of return have been annualized so that the scale is comparable to an ordinary interest rate.)

Figure 3.5 shows both companies to be in very attractive businesses. After suffering a loss in the third quarter of 1985, Apollo recovered to earn between 15% and

²¹This period approximates the product life of the Domain 3000 and Sun-3 families. It includes the initial ramp-ups of plant and equipment in anticipation of the introductions of the Domain 4000 and Sun-4. After mid-1987, Apollo's spending was constrained by cash conservation measures.

²²Free cash flow equals net income plus depreciation expenses plus the change in deferred taxes. It approximates the cash flow available at the end of each quarter that can be reinvested in the business or, at the discretion of management, deployed elsewhere.

20% on operating assets by the second half of 1986.²³ But in the same time frame, Sun was earning 35% to 40% on its operating assets. Sun's greater productivity was evidenced by the fact that although its margins were similar to Apollo's, it returned about twice as much on assets actually used in the business.



Here then is the power of the modular paradigm. By modularizing around a narrow but powerful subset of design elements, using standard components and operating software, and exploiting deep knowledge of the system and interfaces, Sun could offer the market a superior product at an attractive price, earn excellent margins, and employ much less capital in production. It was a sharp departure from the mainframe paradigm, and it spelled trouble for Apollo.

4. Sun's Growth and the Sun-AT&T Alliance

4.1 Hypergrowth: Using an Aggressive Financial Strategy to Exploit Modularity

²³Apollo earned 36% on assets employed in the fourth quarter of 1987. However, both the income and asset figures in that quarter were influenced by year-end asset sales and related adjustments (increases) in depreciation expense. Thus the high rate of return shown should be viewed as anomalous; in reality, Apollo's profitability was deteriorating, as the very low returns in the second and third quarter show. (In terms of net income, Apollo lost money in the second and third quarters of 1988.)

What did Sun choose to do with the capabilities derived from its more efficient systems of design and production? In principle, upon reaching a certain market share (say 50%), it could have adopted the same pricing and product innovation targets as Apollo. The two companies would then have grown at approximately the same rate, but Sun would be twice as profitable. The two might then have sought to differentiate their customer bases in order to maintain margins and minimize head-to-head competition. If both companies had been competing under the mainframe paradigm, their respective customers would soon have been locked into different proprietary architectures, and competition would occur only at the margin (with new customers).

But Sun was promoting open systems and a universal, non-proprietary operating system.²⁴ And rather than settling for high margins and positive cash flow, Sun held its margins down and grew very fast, putting continual competitive pressure on Apollo and other companies seeking a foothold in the workstation market at this time. Sun financed its "hypergrowth" with massive amounts of funding raised in the public capital markets and later from AT&T.

4.2 Financing Hypergrowth

Finance played a crucial role in the interaction between Sun and Apollo in the mid-1980s. Both firms had access to the capital markets, but each approached its capital suppliers very differently.²⁵ Indeed, in 1986 and 1987, Sun embarked on a financial

²⁴UNIX was as close to a universal, non-proprietary operating system as existed at this time.

²⁵Those differences were not due to differences in the way the markets valued the companies. This is somewhat surprising, since Figure 3.5 suggested that a dollar of revenue at Sun would generate cash flow that could be reinvested at about two times the rate of return as a dollar of revenue at Apollo. By this reasoning, Sun's market value per dollar of revenue should be higher than Apollo's – but it was not. In early 1986, before Sun went public, net market capitalization [defined as the market value of equity plus the book value of debt (which is close to the market value) minus cash and investments] of the two companies was almost identical. This is the appropriate measure, because it approximates the total market value (debt plus equity) placed on the *operations* of the business, net of its cash balances. The adjustment is necessary because Sun and Apollo had very different policies with respect to debt and cash balances.

strategy radically different from that of Apollo. Sun went to the capital markets five times in the course of 18 months. The money it raised was used to finance growth at the rate of 25% to 30% per quarter. As a result, in less than two years, Sun's revenues went from 60% to 144% of Apollo's.

Throughout this period, Sun put enormous downward pressure on prices, making it hard for anyone to make a profit or achieve positive cash flow in the workstation industry. In essence, Sun combined its technical productivity and efficiency with very aggressive financial tactics to buy market share in its business.

Sun's opportunity to unseat Apollo arose partly as a result of the latter's own mistakes. In 1985, Apollo greatly overestimated the demand for its products in the face of a general industry slump. It brought a new high-capacity production facility onstream just as demand for its products was evaporating. In the third quarter of that year, Apollo was forced to write off \$30 million in obsolete inventory and underutilized capacity, and suffered an \$18 million after tax loss. Moreover, operating losses combined with capital expenditures on the new plant obliterated the \$50 million cash cushion Apollo had at the start of the year. The company survived by using sale and leaseback financing and drawing down a \$75 million line of credit it had negotiated early in the year.

These events caused a major shakeup in management. In the fourth quarter, William Poduska, Sr., the founder, left, and David Lubrano, the chief financial officer, was ousted. Three (out of seven) members of the board resigned and four new ones were added in the wake of the crisis. The result was to consolidate the position of Thomas Vanderslice, who had been hired as president and CEO in 1984. With Poduska's departure, Vanderslice became Chairman of the Board as well as president and CEO.

One of Vanderslice's first moves was replenish the company's cash by issuing \$115 million in convertible subordinated debentures. About a third of this money was

used to repay existing debt, and thus Apollo ended the first quarter of 1986 with \$75 million in cash and investments and a \$75 million unused line of credit. The 1985 Annual Report (published after the debenture sale) stated:

[Apollo] anticipates that the proceeds from the sale of the Debentures, together with internally generated funds, will be sufficient to meet the Company's capital spending requirements and liquidity needs until mid-1987.

In other words, Apollo was not contemplating the sale of equity, or further borrowing, in the near term. It was already in the process of implementing major cost-cutting measures and planned to live on the cash it had in hand. The issuance of convertible debentures also indicates that Apollo was reasonably confident of its short-term profitability, and optimistic enough to bet on the future conversion of the debentures to equity.

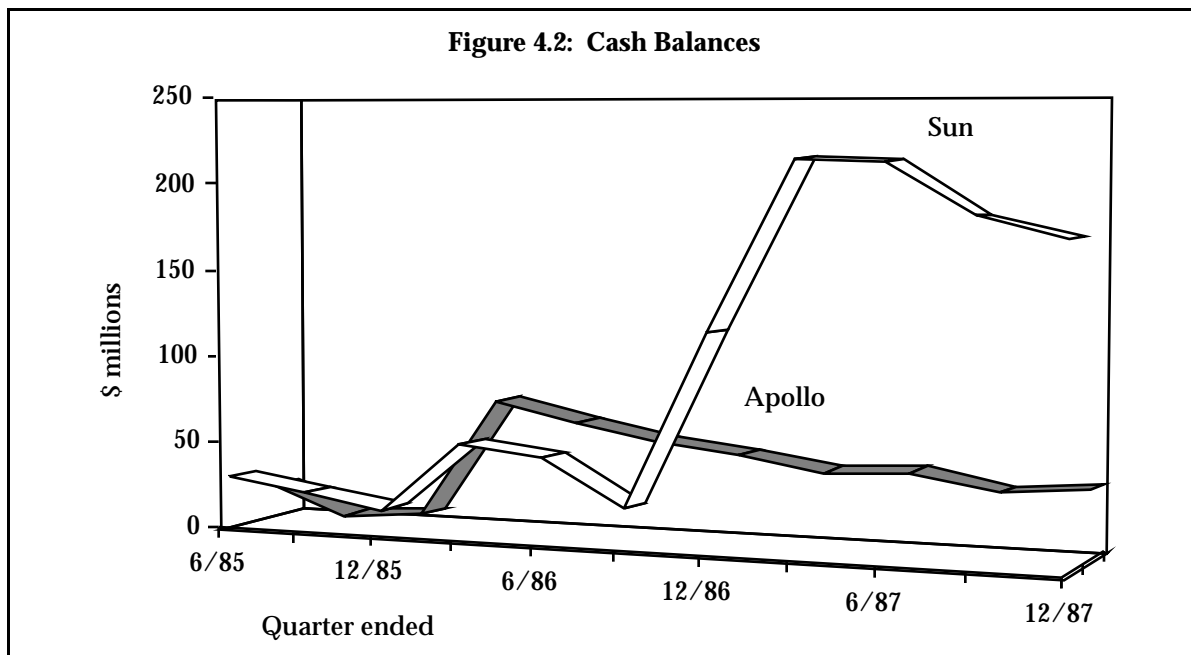
Thus, in early 1986, Apollo's managers did not anticipate they would soon be fighting a battle for survival against a superior rival. Even then Apollo was losing sales to Sun in head-to-head competition and was being forced to change its stance on UNIX and open systems. But the threat posed by Sun's superior technical capabilities was not yet apparent. Sun was still a privately-held, venture-capital-backed startup, its revenues were 60% of Apollo's, its cash balances were a mere \$15 million, and it had \$18 million of senior debt outstanding. In almost every way imaginable, Apollo appeared the stronger, better-capitalized company.

Sun went public in March 1986, raising \$46 million of new equity in its initial public offering. It used this infusion of funds to grow: its cumulative internal funds deficit in the first three quarters of 1986 was almost \$50 million, but sales more than doubled. At this point, the power of Sun's more productive business model became apparent: each dollar invested in Sun supported twice as much revenue as the same dollar invested in Apollo. Thus although Sun's and Apollo's internal funding deficits were about the same in the first nine months of 1986, Sun's quarterly sales went from

\$42 million to \$92 million, while Apollo's quarterly sales climbed from \$71 million to \$100 million. In other words, the same amount of money financed a sales ramp-up of \$50 million per quarter (\$200 million annualized) at Sun, but only \$29 million per quarter (\$116 annualized) at Apollo.

By the third quarter of 1986, Sun's sales were almost equal to Apollo's. But internal deficits caused by high growth eroded its cash balances: by September, Sun was down to \$23 million in cash and investments and consuming money at the rate of \$10 million per month. Confronted with a potential liquidity crisis, Sun arranged four financings in quick succession. In October, it borrowed \$25 million from two banks; in November, it issued \$90 million of new common stock; in December, it arranged a \$50 million revolving line of credit; and in February 1987, it sold \$100 million of convertible subordinated debentures.

Thus in six months, Sun's and Apollo's financial positions were dramatically reversed. In September 1986, Apollo had twice as much cash as Sun (\$52 million to \$23 million) and a substantial unused line of credit. By March 1987, Sun had cash in excess of \$216 million, and another \$70+ million available in bank lines. At this time, almost half of Sun's reported assets consisted of cash and marketable securities. Sun was now not only twice as profitable as its chief rival, but had *five times* as much cash to spend on building market share (see Figure 4.2).



But Sun's war chest was only large relative to Apollo's. Compared to its own potential need for cash, Sun's cash balances in mid-1987 were sufficient for only 12 to 18 months of continued growth. Of course, Sun's rate of cash consumption – colloquially known as its "burn rate" – depended on its pricing strategy. If Sun were to raise prices, its growth rate and correlated need for cash would diminish. The \$200 million war chest would then last a long time, and Apollo and the rest of Sun's competitors would breath easier.

However, Sun's executives built the war chest for a purpose. They proceeded to use it, and the balances began to decline. New financing would be needed before very long. But new cash was unlikely to come from the public capital markets for two reasons. First was the stock market's reaction to the introduction of the Sun-4 product family in June 1987. Consistent with the modular paradigm, Sun introduced its fourth generation product long before the third generation had run out of steam. The Sun-4 clearly cannibalized the Sun-3, and its introduction required significant infusions of cash for phase out/phase in expenses. Wall Street's reaction was immediate: Sun's stock price dropped 18% in the two trading days surrounding the Sun-4 announcement. In

contrast to the steady upward progress of the stock from September 1986 to May 1987, the stock fell from its peak of 44 and settled in the mid-30s by October 1987. And then came the second problem: Black Monday. Following the crash of the stock market on 19 October 1987, Sun's stock dropped to the low 20s and then rebounded to the low 30s. At this point the outlook for new equity issues by small, risky companies was quite uncertain. But at the same time, following a very enthusiastic response to the Sun-4, Sun was consuming cash at the rate of \$15 million per month.

In the midst of this uncertain financial environment Sun sought new cash from a familiar source: the corporate investor.²⁶ Sun had already worked out a technical agreement with AT&T covering the development of UNIX; an equity investment would complete the package. But this was to be no "plain vanilla" deal. Not only were the amounts of cash sizable, but the deal was structured to give Sun valuable flexibility in its battle for supremacy in workstations.

In late 1987, Sun and AT&T completed a financial agreement that allowed Sun to "put" its common stock to AT&T at a premium to the market price (25% over the 20 day trailing average). The agreement gave Sun the right to demand that AT&T purchase up to 6 million new shares of Sun stock over the next six years. (Sun had 34 million shares outstanding at the time.) Sun could trigger a sale by sending a notice to AT&T, which then had 30 days to complete the transaction.²⁷

²⁶In 1984, when Sun was faced with rapid growth and the need for additional capital, it had negotiated a deal with Kodak (an important OEM buyer of Sun workstations that it incorporated into its printing products) for \$20 million in equity. After holding the equity for 30 months, Kodak cashed out at \$97 million, an annualized return of 88%.

²⁷Sun 8-K. There were a number of other provisos in the agreement. In a single sale notice, Sun could not demand that AT&T purchase more than 25% of the 6 million shares reserved, and Sun could not send more than one notice in any 90-day period. Finally, Sun had to place half the direct placement shares before 5 July 1989 (2-1/2 years after the start of the agreement) or lose its right to require purchase of the shares. The rest of the shares had to be sold by 5 January 1993, the date the agreement expired. AT&T also had the right to purchase 5% of Sun's shares in the open market, and to participate in up to 17.65% of any other offering of Sun's stock. However, AT&T's voting power was strictly limited: it received a seat on Sun's Board, but could not vote against management except in special circumstances. Furthermore, AT&T could not acquire more than 20% of Sun's voting shares without Sun's consent unless Sun became the object of a third-party takeover bid.

The direct effect of the AT&T financing was to increase Sun's war chest by an amount proportional to its stock price. At the time the deal was announced, a conservative estimate of the total amount of money Sun could obtain was 140% of the value of 6 million shares. Sun's shares were then trading at around \$38 per share, and thus the AT&T financing gave Sun access to more than \$300 million of additional cash. (The technical details of the valuation of the agreement are found in a separate appendix available from the authors.)

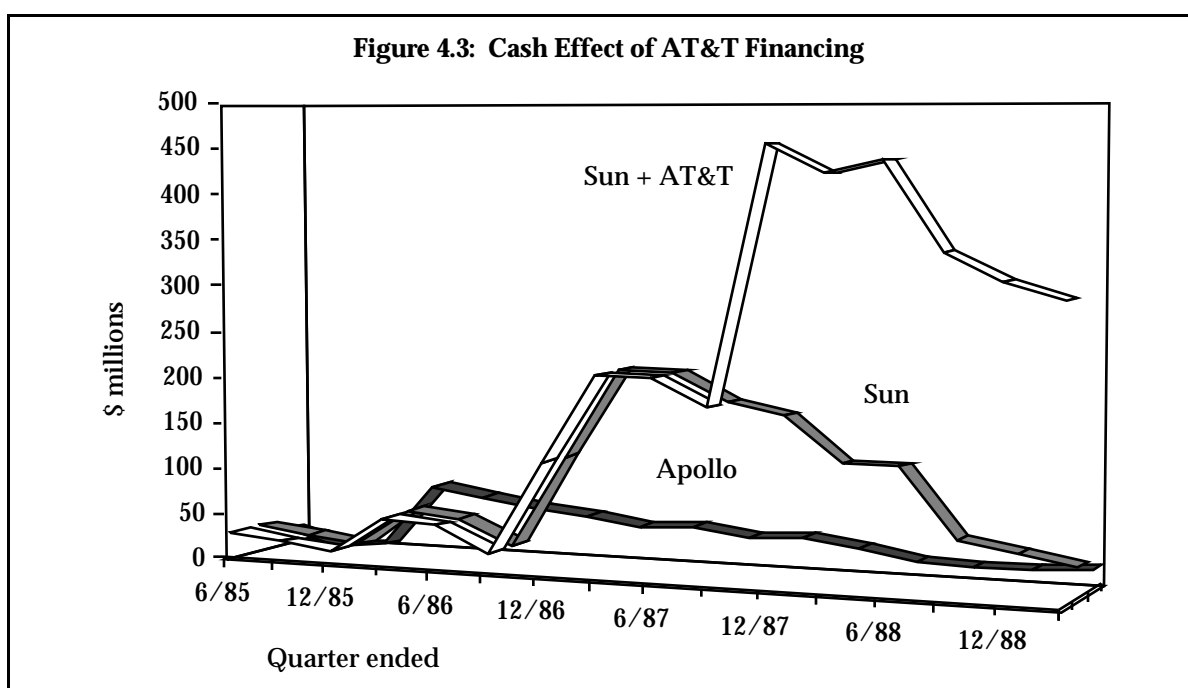


Figure 4.3 shows the implicit effect of the AT&T financing on the relative cash balances of Sun and Apollo through the first quarter of 1989. (This was the last quarter Apollo existed as a freestanding company.) The difference in resources between the two companies is staggering. By the end of 1987, Apollo's cash resources had dwindled to \$40 million – a small sum compared to the \$177 million in cash on Sun's balance sheet, and dwarfed by the \$458 million Sun could raise by exercising its rights to issue shares to AT&T.

The difference in financial resources was mirrored in the two companies' spending rates. By the first quarter of 1989, Sun's internal quarterly cash deficit was in

excess of \$130 million, and sales were increasing by about \$50 million per quarter.²⁸ In stark contrast, Apollo was attempting to grow with no new cash infusions at all. The pressure Sun was putting on industry-wide margins made it hard for Apollo to show a profit, and to break even it was forced to cut back on selling, general, and administrative expenses. Throughout this period Apollo managed to maintain its R&D spending in proportion to sales. Unfortunately, Apollo's engineers had to design and engineer a line of products as complex and extensive as Sun's on about 40% of Sun's R&D budget. Toward the close of 1988, it became clear that Apollo could not survive as an independent company without a massive funds infusion, but by then the total market value of its equity had dropped to around \$300 million. At this point, a 100% dilution of Apollo's existing equity would have allowed it to match Sun's spending for two or three quarters at a maximum.

5. The Aftermath: Conclusions and Implications

The battle between Sun and Apollo is testimony to the power of modularity, both as a principle of design and as the basis for a new paradigm for competition. Sun's experience shows the decisive impact on productivity when a modular design creates a dense modular cluster of companies competing on independent modules. As the architect, Sun could exploit the variety of parallel initiatives to push knowledge of components deeper, performance higher, and cost lower. Moreover, by relying on standard interfaces, Sun could mix and match components to achieve what we have called the "best of the best."

²⁸ This indicates that competition was taking its toll. In 1986, \$50 million of incremental spending at Sun had supported about \$50 million of incremental sales per quarter (\$200 million annualized). Now the same \$50 million of incremental sales per quarter needed outlays of \$130 million. This erosion of performance was caused by a reduction in margins and by a decrease in Sun's efficiency with respect to net working capital and property, plant and equipment.

In principle, this flexibility was available to Apollo -- it was even touted in their public statements and pronouncements. But Sun took the principle of modularity much further than Apollo extending it to the operating system, networking hardware, and software. And Sun's bet on the capabilities inherent in the modular cluster paid off in one respect: in terms of customer acceptance and machine performance, Sun's full blown modularity beat Apollo's limited modularity. Mixing and matching designs across the full modular cluster delivered systems with lower cost, higher performance and more user options than mixing and matching designs within a single firm. This was true even though Apollo's proprietary solutions were regarded as "elegant" and "superior."

Modularity was the source of power behind Sun's strategy, but it was Sun's specific choice of a design (the memory-CPU interface and a single board), its approach to manufacturing and its aggressive financial strategy that allowed it to exploit that power in competition. However, Sun's victory over Apollo turned out to be no guarantee of supranormal profits, nor even of a quiet life with average returns. Indeed, what happened to Sun illustrates precisely the difficulty an architect has when it attempts to convert a successful design into a dominant market position within a modular cluster.

Sun's first problem was that the niche it sought to dominate (networked workstations) was being invaded from all sides. Large, well financed, capable players -- Hewlett Packard, IBM, and DEC -- realized that Sun's products would not only be attractive to engineers and software designers, but would be competitive in business applications as well. They had to develop directly competitive products or lose the business. At the same time, personal computer manufacturers, designers and distributors had already discovered the power of the modular paradigm, and were rapidly increasing the power and range of their products. They too saw servers, networks, and workstations as targets of opportunity. Framed in these terms, what

Sun won in its battle with Apollo was the opportunity to play in a larger game against bigger players.

The good news was that the big game was potentially very valuable. The bad news was that many important players had good reason not to let Sun achieve a dominant position in UNIX-based, RISC-powered, high-performance workstations. This became crystal clear soon after Sun announced the AT&T financing agreement. Although the technical agreement with AT&T announced in October 1987 brought little reaction (almost everyone had some kind of technical agreement with AT&T), the announcement of the financing agreement in January 1988 set off a firestorm of protest from other UNIX licensees. This group feared that Sun's deal with AT&T would cut them out of UNIX development and give Sun both a lead time and a performance advantage. Eventually they believed UNIX would be optimized for Sun's processors and systems.

AT&T tried to placate the licensees, but to little avail. In May 1988 DEC, Apollo, Hewlett Packard, IBM and others established the Open Software Foundation (OSF) to create an independent UNIX standard. To Sun's dismay, AT&T then abandoned the idea of setting a single standard with Sun. In December 1988, AT&T and other companies that had supported the AT&T/Sun alliance set up their own independent UNIX development group called UNIX International (UI). In 1989, UI forced Sun and AT&T to shut down their joint development lab. Sun (which joined UI) would have no exclusive access to any part of the development process.

The concerted reaction of competitors prevented Sun from dominating UNIX, which would have given the company control over a powerful element of the modular cluster. But even without such control, Sun was in a relatively strong position. Hewlett Packard was occupied with trying to absorb Apollo (which it acquired in May 1989) and was attempting to develop a coherent product line. DEC was struggling with multiple platforms in its workstation lineup (VAX-VMS and MIPS-UNIX), and IBM was trying to

develop a real workstation product after the embarrassing failure of its RT entry. Thus, the big players were strong enough to prevent Sun from dominating UNIX, but they were not particularly formidable in the marketplace.

But the dynamics of the modular paradigm implied that Sun had to move quickly and hit successive windows of opportunity. Instead Sun lost its technological focus – a not uncommon occurrence in the late 1980s. At the time Intel's x86 line of processors was expanding in power and PCs were invading workstation turf. RISC technology (to which Sun was closely connected in history and in technical philosophy) was emerging from the laboratory. And Sun had built a huge business around Motorola's 68000 product line.

Sun's answer to this quandary was to pursue all options at once. It first tried to develop a RISC chip jointly with Motorola (Sun would bring ideas on RISC architecture, Motorola would bring expertise in design and processing). But Motorola was not interested and Sun then proceeded to design its own processor (SPARC), which it licensed to chip makers and other computer companies. Meanwhile, another group at Sun developed a workstation built around Intel's 80386 processor and, of course, there was the mainline product family built around Motorola's CPU. The upshot was that Sun had to support three platforms, all the while confusing its customers.

Sun executives soon realized this strategy was unsustainable, and refocused the company around SPARC. (Internally this move was described as "we need to get all our wood behind a single arrow.") But then, Sun stumbled again. After initial success with its line of SPARCstations, Sun was late to market with the next generation of its processor. This gave Hewlett Packard and IBM an opportunity to re-enter the competitive arena with their own products. Sun could no longer claim price/performance leadership. With its installed base, its strong reputation, and its ability to market and price aggressively, it was able to maintain a sizable market share

in workstations. But its opportunity to lead and dominate the workstation market evaporated, and its financial performance and market value stagnated.

In summary, Sun's experience underscores both the power and the peril of the modular paradigm. Moreover, it raises fundamental questions about the longer term evolution of an industry such as computers where modularity is fundamental, and modular clusters emerge as the primary mode of competition.

REFERENCES (incomplete)

- Alexander, Christopher (1964) *Notes on the Synthesis of Form* (Cambridge, MA: Harvard University Press).
- Baldwin, Carliss and Kim B. Clark (1993) "Modularity and Real Options" Harvard Business School Working Paper #93-026.
- Bhide, Amar (1989) "Vinod Khosla and Sun Microsystems (A)" (Boston, MA: Harvard Business School Publishing Division).
- Besen, Stanley M. and Garth Saloner (1989) "The Economics of Telecommunications Standards," in R.W. Crandall and K. Flamm (eds.), *Changing the Rules: Technological Change, International Competition, and Regulation in Communications* (Washington, D.C.: Brookings Institution).
- Black, Fischer and Myron Scholes (1973) "The Pricing of Options and Corporate Liabilities," *Journal of Political Economy*, 81: 637-654.
- Bresnahan, Timothy F. and Shane Greenstein (1992) "Technological Competition, and the Structure of the Computer Industry," CEPR Publication No. 315, Center for Economic Policy Research, Stanford University, Stanford, CA.
- Brooks, Frederick W. (1975) *The Mythical Man Month* (New York: Addison Wesley).
- Christensen, Clayton M. (1993, Winter) "The Rigid Disk Drive Industry: A History of Commercial and Economic Turbulence," *Business History Review* 67 (4): 531-38.
- Clark, Kim B. (1985) "The Interaction of Design Hierarchies and Market Concepts in Technological Evolution," *Research Policy* 14 (5): 235-51.
- Clark, Kim B. (1988) "Knowledge, Problem Solving, and Innovation in the Evolutionary Firm," Harvard Business School Working Paper.
- Constantine, L.L., G.J. Myers, and W.P. Stevens (1974) "Structured Design," *IBM Systems Journal* 13 (2): 115-39.
- Dixit, Avinash K. and Robert S. Pindyck (1994) *Investment Under Uncertainty* (Princeton, NJ: Princeton University Press).
- Dorfman, Nancy S. (1986) *Innovation and Market Structure: Lessons from the Computer and Semiconductor Industries* (Cambridge, MA: Ballinger).
- Eppinger, Steven D. (1991) "Model-based Approaches to Managing Concurrent Engineering," *Journal of Engineering Design* 2 (4).
- Eppinger, Steven D., Daniel E. Whitney, Robert P. Smith, and David Gebala (1994) "A Model-Based Method for Organizing Tasks in Product Development," *Research in Engineering Design* 6 (1).

- Eppinger, Steven D. and Kent R. McCord (1993, August) "Managing the Iteration Problem in Concurrent Engineering," MIT Working Paper 3594-93-MSA.
- Farrell, Joseph and Garth Saloner (1986) "Economic Issues in Standardization," *Telecommunications and Equity: Policy Research Issues* (New York, NY: North-Holland).
- Farrell, Joseph, Hunter K. Monroe, and Garth Saloner (1993) "Order Statistics, Interface Standards and Open Systems," mimeo, University of California, Berkeley, CA.
- Farrell, Joseph and Carl Shapiro, "Optimal Contracts with Lock-in," *American Economic Review* 79 (1): 51-68.
- Ferguson, Charles H. and Charles R. Morris (1993) *Computer Wars: How the West Can Win in a Post-IBM World* (New York, NY: Times Books).
- Freeze, Karen and Kim B. Clark (1986) "Sun Microsystems, Inc. (B)" Harvard Business School Publishing Division, Boston, MA.
- Garud, Raghu and Arun Kumaraswamy (1993, July) "Changing Competitive Dynamics in Network Industries: An Exploration of Sun Microsystems' Open Systems Strategy," *Strategic Management Journal* 14 (5): 351-69.
- Gill, Geoffrey and Steven C. Wheelwright (1989) "Sony Corporation: Workstation Division," Harvard Business School Publishing Division, Boston, MA.
- Greenstein, Shane (1993, December) "Markets, Standards and the Information Infrastructure," *IEEE Micro*, 36-51.
- Harrison, J.M. (1985) *Brownian Motion and Stochastic Flow Systems* (NY: John Wiley and Sons).
- Henderson, Rebecca M. and Kim B. Clark (1990), "Generational Innovation: The Reconfiguration of Existing Systems and the Failure of Established Firms," *Administrative Sciences Quarterly* 35: 9-30.
- Iansiti, Marco (1993) "Technology Integration: Managing Technological Evolution in a Complex Environment," Harvard Business School Working Paper 93-057.
- Iansiti, Marco and Kim B. Clark (1993) "Integration and Dynamic Capability: Evidence from Product Development in Automobiles and Mainframe Computers," Harvard Business School Working Paper 93-047.
- Katz, Michael L. and Carl Shapiro (1985) "Network Externalities, Competition and Compatibility," *American Economic Review* 75 (3): 424-40.
- Katz, Michael L. and Carl Shapiro (1992) "Product Introduction with Network Externalities," *Journal of Industrial Economics* 40 (1): 55-83.

- Kogut, Bruce and Nalin Kulatilaka (1992) "Options Thinking and Platform Investments: Investing in Opportunity," mimeo, Wharton School, University of Pennsylvania.
- Langlois, Richard N. and Paul L. Robertson (1992) "Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries," *Research Policy* 21: 297-313.
- Langowitz, Nan and Steven C. Wheelwright, "Sun Microsystems, Inc. (A)," Harvard Business School Publishing Division, Boston, MA.
- Lindgren, B.W. (1968), *Statistical Theory* (New York, NY: MacMillan).
- Marples, David L. (1961) "The Decisions of Engineering Design," *IEEE Transactions on Engineering Management* 2: 55-71.
- Mason, Scott P. and Robert Merton (1985) "The Role of Contingent Claims Analysis in Corporate Finance," in E. Altman and M. Subrahmanyam, *Recent Advances in Corporate Finance* (Homewood, IL: Richard D. Irwin).
- Merton, Robert C. (1973) "Theory of Rational Option Pricing," *Bell Journal of Economics and Management Science* 4: 141-83.
- Merton, Robert C. (1990) *Continuous Time Finance* (Cambridge, MA: Basil Blackwell).
- Milgrom, Paul and John Roberts (1990) "The Economics of Manufacturing: Technology, Strategy and Organization," *American Economic Review* 80 (3): 511-28.
- Morris, Charles R. and Charles H. Ferguson (1993, March-April) "How Architecture Wins Technology Wars," *Harvard Business Review*, 86-96.
- Myers, Glenford J. (1975) *Reliable Software through Composite Design* (New York: Van Nostrand Reinhold).
- Nelson, Richard R. and Sidney G. Winter (1982) *An Evolutionary Theory of Economic Change* (Cambridge, MA: Harvard University Press).
- Nevins, James L. and Daniel E. Whitney (1989) *Concurrent Design of Products and Processes* (New York, NY: McGraw-Hill).
- Parnas, D.L. (1972, May) (1972a) "A Technique for Software Module Specification with Examples," *Communications of the ACM* 15: 330-36.
- Parnas, D.L. (1972, December) (1972b) "On the Criteria to Be Used in Decomposing Systems into Modules," *Communications of the ACM* 15: 1053-58.
- Parnas, D.L., P.C. Clements, and D.M. Weiss (1985, March) "The Modular Structure of Complex Systems," *IEEE Transactions on Software Engineering*, SE-11: 259-66.

- Pugh, Emerson W., Lyle R. Johnson, and John H. Palmer (1991) *IBM's 360 and Early 370 Systems* (Cambridge, MA: MIT Press).
- Reinganum, Jennifer F. (1989) "The Timing of Innovation: Research, Development and Diffusion," in Richard Schmalensee and Robert D. Willig (Eds.), *Handbook of Industrial Organization, Volume 1* (New York, NY: Elsevier Science Publishers B.V.).
- Rothwell, Roy and Paul Gardiner (1988) *Journal of Marketing Management* 3 (3): 372-87.
- Sahlman, William A. and Howard H. Stevenson (1985) "Capital Market Myopia," *Journal of Business Venturing* 1: 7-30.
- Sanchez, Ronald A. (1991), "Strategic Flexibility, Real Options and Product-based Strategy," Unpublished Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA.
- Schach, Stephen R. (1993) *Software Engineering, 2nd Ed.* (Burr Ridge, IL: Richard D. Irwin).
- Simon, Herbert A. (1969) *The Sciences of the Artificial* (Cambridge, MA: MIT Press).
- Shaked, Avner and John Sutton (1983) "Natural Oligopolies," *Econometrica* 51: 1469-84.
- Soll, Jack and Carliss Baldwin (1990) "Sun Microsystems, Inc. – 1987 (A), (B), and (C)" Harvard Business School Publishing Division, Boston, MA.
- Stigler, George (1951, June) "The Division of Labor is Limited by the Extent of the Market," *Journal of Political Economy* 59: 3.
- Stulz, Rene M. (1982) "Options on the Minimum or the Maximum of Two Risky Assets," *Journal of Financial Economics* 10: 161-85.
- Sutton, John (1991) *Sunk Cost and Market Structure: Price Competition, Advertising and the Evolution of Concentration* (Cambridge, MA: MIT Press).
- Ulrich, Karl T. (1992) "The Role of Product Architecture in the Manufacturing Firm," Working Paper No. 3483-92-MSA, Sloan School of Management, MIT, Cambridge, MA.
- Ulrich, Karl T. and Karen Tung (1991) "Fundamentals of Product Modularity," *Proceedings of the 1991 ASME Winter Annual Meeting Symposium on Issues in Design/Manufacturing Integration, Atlanta, GA.*
- Ulrich, Karl T. and Steven D. Eppinger (1994) "Product Architecture," *Methodologies for Product Design and Development* (New York: McGraw-Hill).
- Von Hippel, Eric (1990) "Task Partitioning: An Innovation Process Variable," *Research Policy* 19: 407-18.