Architectural Strategy for Digital Platforms: Technological and Organizational Perspectives

Authors

Jason Woodard, Singapore Management U., jwoodard@smu.edu.sg
Joel West, San Jose State U., Joel.West@sjsu.edu

Architectural Strategy for Digital Platforms: Technological and Organizational Perspectives

Abstract

Digital systems are malleable in the sense that their components can typically be reconfigured at low cost. Malleability confers unprecedented freedom to arrange components in new ways, easing the creation of innovative designs but also complicating the task of making good design choices. This paper explores this tension from the perspective of a firm that aspires to platform leadership in an industry that creates digital products or services. We frame the firm's challenge as a design problem aimed at changing the way a system is decomposed into components, the way the components are linked together by interfaces, or both. Building on the concept of architectural innovation, we integrate prior research to yield a new framework for studying strategic architectural decisions that span the domains of technology and organizations. Using examples from the computing and communications industries, we show how our framework sheds light on four distinct types of design choices faced by firms that create or participate in digital platforms. Further analysis reveals a variety of interactions among these choices, which create both opportunities and threats in the form of mutually reinforcing or conflicting design decisions.

1. Introduction

We are surrounded by objects of astonishing complexity, many made possible by advances in digital technology. Software programs containing tens of millions of instructions are deployed in billions of increasingly interconnected devices, including cameras, phones, cars, and computers. These devices enable a large and growing fraction of the world's population to interact with each other, further driving demand for innovative ways to share and manage information.

This paper examines the design of digital artifacts and the organizations that create them, focusing on changes in their architecture — that is, the way they are decomposed into component parts, nested within larger systems, and linked together by interfaces. We first synthesize the literatures on innovation management and organizational design to develop a view of *architectural strategy* that encompasses design decisions about both technological and organizational components and the interfaces between them. Second, we apply this framework to analyze three episodes of architectural innovation involving digitally enabled platforms. Our analysis suggests that managing interactions between the technological and organizational aspects of a firm's architectural strategy is a pressing and difficult challenge.

While the importance of architectural strategy is most apparent in software-based systems, the growing mobility and scale of these systems — along with the pervasive embedding of digital technologies into non-digital artifacts — increasingly warrants the attention of technology and innovation scholars more generally, as well as organizational researchers (Lyytinen and Yoo 2002). Digitalization is blurring the boundaries between previously distinct industries such as computing and communications, which in turn increases market turbulence and perpetuates technological ferment (Bogner and Barr 2000). In such hypercompetitive

environments, architectural change is an important phenomenon not only in the wake of a technological discontinuity (Anderson and Tushman 1990), but at all stages of industry evolution. Moreover, since complex artifacts evolve through a nested hierarchy of technology cycles (Murmann and Frenken 2006), multiple architectural changes may happen concurrently, posing critical challenges for organization design.

Some architectural changes may follow predictable technological trajectories, but many others arise through deliberate strategy and focused innovative effort. The fact that Bill Gates appointed himself "chief software architect" after stepping down as Microsoft's CEO in 2000 is but one example of the importance of architectural decisions to suppliers of digital products and services. More broadly, a 2008 survey of IT executives across 18 industry sectors found that 77% of responding organizations employed a full-time "enterprise architecture" team, and in 62% of these organizations the head of this team reported directly to a "C-level" executive (CIO, CTO, CFO, or CEO) or to the board of directors (Obitz and Mohan 2008). Yet, despite the growing number of practitioners who identify themselves as architects, few research studies have addressed the role of architectural strategy in the evolution of digital systems.

This paper provides a preliminary framework to help designers of products, services and organizations answer three kinds of questions. First, how are the problems faced by designers in different domains related to each other? Second, what kinds of tensions can arise in developing an integrated architectural strategy across these domains? And third, how can firms successfully navigate these tensions to achieve and sustain a position of platform leadership?

These questions are especially important in complex, fast-moving markets like mobile telephony, where firms are grappling simultaneously with a shift to third-generation wireless data standards, consumer demand for integrated multimedia features, and platform competition

between embedded operating systems. Our aim is to advance the research frontier by shifting attention from the general characteristics of these settings (e.g., turbulent, hypercompetitive, or network-structured) to the specific sequences of design moves by which firms shape their technological and organizational environments.

The paper is structured as follows. Section 2 motivates our framework by reexamining the concept of architectural innovation from the perspective of an innovator in an industry shaped by digital technology. Section 3 presents the framework itself, and uses it to review and synthesize the literatures on product and organization design. Section 4 demonstrates the use of our framework to contrast and compare the architectural strategies of IBM, Sun, and Nokia for their respective digital platforms. Section 5 reviews the implications of our analysis for firms that create or participate in platform architectures, and highlights opportunities for future research.

2. Architectural Innovation as a Design Process

In their seminal paper, Henderson and Clark (1990, p. 10) define an architectural innovation as a type of technological discontinuity that "change[s] the way in which the components of a product are linked together, while leaving the core design concepts (and thus the basic knowledge underlying the components) untouched." They note that incumbent firms often find it difficult to adapt to these seemingly minor changes, citing as examples RCA in portable radio receivers and Xerox in small copiers, as well as the photolithographic alignment equipment makers featured in Henderson's landmark study.

While Henderson and Clark develop a detailed theory to explain the *effects* of architectural innovation on established firms, they elaborate little on the *process* of architectural innovation from the innovator's perspective. They acknowledge the selective focus of the paper,

and invite further research into the proactive use of architectural innovation for strategic advantage (pp. 28–29):

> Since architectural innovation has the potential to offer firms the opportunity to gain significant advantage over well-entrenched, dominant firms, we might expect less-entrenched competitor firms to search actively for opportunities to introduce changes in product architecture in an industry … As an interpretive lens, architectural innovation may therefore prove quite useful in understanding technically based rivalry in a variety of industries.

Here we take up their invitation by viewing architectural innovation as a sequence of strategic design decisions that are endogenous to the innovating firm, resulting in the reconfiguration of technological and/or organizational components, which may in turn reshape the firm's internal and external environment.

This process-oriented view is consistent with a long tradition of design research. Simon (1962) explained the ubiquity of hierarchical structures as a natural consequence of their evolutionary stability. In his classic treatise on the synthesis of form, Alexander (1964) characterized design as a process of creating forms that fit their context. Clark (1985, p. 237) extended this idea to the industry level by exploring "the sequence of design decisions that emerge over time … [which] determines the pattern of change in product and process technology." More recently, scholars have noted that the traditional tools of decision theory are of limited value for reasoning about these kinds of decisions, because they tend to assume a fixed and enumerable set of alternatives to choose from (Boland and Collopy 2004). In contrast, design research emphasizes the emergent nature of these alternatives (Orlikowski 2000) and the iterative process of generating and evaluating them (Simon 1969, Hevner et al. 2004).

The "design attitude" (Boland and Collopy 2004) is especially valuable in the context of digital systems, which are *malleable* in the sense that their software-intensive components can typically be reconfigured at low cost relative to systems composed of components like

mechanical parts, chemical molecules, or biological tissues. This property arises from the fact that digital components, which are linked by informational interactions rather than physical ones, can be more loosely coupled (Weick 1976) than other kinds of system elements.[1] Loose coupling means that designers of digital innovations typically have even more alternatives available — that is, operate in a larger design space — than designers of physical artifacts. As a result, the structure of a digital system is determined less by exogenous physical constraints (e.g., the number of features that can be etched onto a piece of silicon) than by decisions that are endogenous to the design process (e.g., the choice of how to represent a document as a string of binary digits).

Even with the best tools at their disposal, designers of malleable systems face a daunting task: as hard as it is to design a good system architecture, its value may be affected by the decisions of other stakeholders, including suppliers of components, complementary products, and competing systems — all of whose fates are intertwined in a shared business ecosystem (Iansiti and Levien 2004). Architectural innovation is thus a continuous challenge that requires the active engagement of both technologists and organization designers.

3. Linking Technological and Organizational Perspectives on Architectural Strategy

Henderson and Clark (1990) define architectural innovation with respect to product architectures. This section extends the concept of architectural innovation into the domain of organizations by considering changes in organizational components (e.g., firms, divisions, and departments) and their linkages (e.g., contracts, reporting relationships, and information flows).

[1] In a typical engineered system, components may interact with each other through physical adjacency and/or exchanges of material, energy, and information (Pimmler and Eppinger 1994). In digital systems, informational interactions usually dominate the concerns of system designers.

The resulting framework provides a basis to bridge the literatures on product and organization design and to develop the concept of architectural strategy, which was introduced by Ferguson and Morris (1993) in an influential article but remains surprisingly underexplored in the academic literature.

Here we consider architectural strategy across two dimensions. The first dimension considers two fundamental design elements, components and interfaces. The second dimension subdivides the technical domain from the organizational one. Together, these two dimensions suggest four different quadrants of interest (Figure 1). The remainder of the section briefly reviews the literature related to each quadrant, then explains how the four quadrants yield complementary perspectives on architectural strategy for systems that comprise both technological and organizational components.

3.1 Quadrants I & II: Product and System Architecture

Ulrich (1995, p. 419) defines product architecture as "the scheme by which the function of a product is allocated to physical components." This scheme includes "(1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components." Components may be hierarchically nested, from systems and subsystems all the way down to individual parts (Simon 1962, Murmann and Frenken 2006). Components with strong internal interdependencies but relatively weak linkages to the rest of the system are called modules (Baldwin and Clark 2000).[2] The large and vibrant literature on product modularity explores the ways in which modular designs enable decentralized innovation, rapid product evolution and economies of scale and scope (Langlois and Robertson 1992, Garud and Kumaraswamy 1995, Sanchez and Mahoney

[2] Since our framework applies to both modular and integral architectures, we use the term component to avoid confusion. Otherwise, "component" and "module" are interchangeable for our purposes.

1996, Schilling 2000). This literature also warns that modular architectures can facilitate imitation by competitors (Ethiraj et al. 2008), and finds that re-integrating such an architecture can also yield competitive advantage (Fixson and Park 2008).

Interfaces describe how components interact, "including how they will fit together, connect, and communicate" (Baldwin and Clark 1997, p. 86). This visible information is used by designers of complementary components and compatible substitutes (also known as "clones"), giving interfaces an important role in both *de facto* and *de jure* standardization processes (Saloner 1990, David and Greenstein 1990). The ability to influence the design of key interfaces and control the disclosure of these designs can either encourage or deter both complementors and competitors, with profound effects on the value of a system and its architecture (Farrell and Saloner 1992, Morris and Ferguson 1993).

While Baldwin and Clark classify interfaces and architectures as different types of design rules, the recent engineering literature follows Ulrich in treating information about both components and interfaces as part of a system's architecture (e.g., Maier et al. 2001). We adopt this broader view while affirming the fundamental distinction between architectural and component knowledge articulated by Henderson and Clark (1990). An architectural innovation may change the arrangement of components in a system, and thus the pattern of linkages between them. In our framework, such changes are the focus of the architectural perspective we label Quadrant I. It is also possible to change an interface between components without changing the overall structure of their interdependencies, such as by changing the shape of a physical connector or the format of a communication protocol, which we label Quadrant II.

Interface changes are especially common in digital systems, where the details of an interface — down to the exact sequence of binary digits needed to invoke a particular function

— are often as important as the larger pattern of structural relationships in the system. Designers may also add interfaces to a system, for example to provide compatibility between otherwise incompatible components (West and Dedrick 2000). New architectural layers are created to encapsulate existing functionality and provide access through a common interface, which Baldwin and Clark (2000) label inversion and porting. Such layering tends to be more extensive in software-based digital systems than in physical ones, because rapid increases in computing power allow new layers to co-exist with existing interfaces with minimal impact on the performance or production cost of the system.

3.2 Quadrants III & IV: Organization and Industry Architecture

Not coincidentally, organization scholars have long employed architectural concepts to describe the structure and evolution of human social systems. Although the concept of architectural innovation was first applied in the context of product development, the term itself was suggested by Michael Tushman (Henderson and Clark 1990, p. 10), whose own work on organizational architecture stems from the observation that "social organisms display many of the same characteristics as mechanical and natural systems" (Nadler and Tushman 1997, p. 26). More recently, researchers have extended this approach to the industry level by studying the architecture of inter-firm networks that support the production of complex multi-product systems (Prencipe et al. 2003, Jacobides et al. 2006).

The idea of treating organizational structure as a design problem was widely explored in the 1960s and 1970s (e.g., Thompson 1966, Galbraith 1973) and continues in the modern literature on organization design (Weick 2004, Yoo et al. 2006). Even the early work on this topic recognized an explicit parallel with the design of technological systems (Haberstroh 1965), as it sought to uncover principles for achieving an optimal design in a given environment. More

recent research has applied engineering techniques such as computational simulation to expand our understanding of organizational design parameters such as coupling, centralization and hierarchical control (Marengo et al. 2000, Ethiraj and Levinthal 2004).

A key concern of organization design is the division of tasks among organizational units such as firms, departments, and teams. Although the terms "module" and "component" are used less frequently to describe units of organizational structure, the organization design problem is analogous to the modular decomposition of a product or service. In research on this topic, the term *organizational architecture* has been used to describe the formal structure of an organization, such as a multi-divisional hierarchy, as well as the linking mechanisms that coordinate interactions between individuals and groups, such as interdepartmental liaisons or matrix reporting relationships (Nadler et al. 1992, Nadler and Tushman 1997). The architectural perspective we label Quadrant III focuses on the former ("organizational components"), while Quadrant IV emphasizes the latter ("organizational interfaces").

The study of organizational architecture is complicated by the fact that different levels of analysis present different design issues, which in turn have attracted the attention of different scholarly communities. For example, problems related to interfaces between individuals and teams are well described by the literature on boundary objects (Star and Griesemer 1989). Carlile (2002) found that artifacts such as drawings, databases, and process descriptions can mediate interactions across functional groups with disparate knowledge bases; the "shared syntax" established by these objects facilitates knowledge transfer in the same way that a software interface facilitates the transfer of information between digital components.[3]

[3] Carlile's semantic and pragmatic approaches to coordination are also relevant to digital interfaces, as evidenced by two recent trends in computer science: semantic web services (McIlraith et al. 2001), in which interactions between software components are facilitated by annotations describing what they do; and autonomic computing

Other work examines the design of coordinating structures at the level of a firm and its value network (Brusoni et al. 2001, Christensen et al. 2002, Maula et al. 2006), and even an entire industry (Jacobides et al. 2006). At these levels, "components" are typically business units or whole firms, and "interfaces" often take the form of contracts that mediate transactions and information exchange (Baldwin 2008).

3.3 Cross-quadrant Challenges in Architectural Strategy

The four quadrants of our framework provide a convenient way to organize a large and diverse body of literature on the architecture of socio-technical systems. But some of the most challenging issues in the design of such systems span the technological and organizational domains, as well as the component and interface perspectives within each domain. Therefore, as the case discussions in Section 4 will show, system architects must be alert to the possibility of tensions between quadrants. Existing research has identified some of these cross-quadrant concerns, most notably in the literature on the duality between product and organizational architectures. After briefly reviewing this work, we highlight the importance of integrating the four perspectives in the context of digital platforms.

Henderson and Clark (1990, p. 27) suggested an intriguing relationship between what we would label Quadrants I and III: "We have assumed that organizations are boundedly rational and, hence, that their knowledge and information-processing structure come to mirror the internal structure of the product they are designing." This assumption has come to be known as the "mirroring hypothesis" (Colfer 2007). Von Hippel (1990) examines the more general issue of task partitioning in product development, and notes that while problem-solving tasks are not always most efficiently partitioned according to the structure of the product being developed, in

(Kephart and Chess 2003), in which complex computing systems manage themselves to achieve specific goals rather than following a pre-defined set of instructions.

practice this is often the case. Sanchez and Mahoney (2006, p. 64) extended this argument to the inter-firm level, arguing that while integral products are best developed within a single firm, "the standardized component interfaces in a modular product architecture provide a form of *embedded coordination* that greatly reduces the need for overt exercise of managerial authority to achieve coordination of development processes," allowing components to be developed by loosely coupled organization structures.

Empirical evidence on the mirroring hypothesis has been mixed. Staudenmayer et al. (2005) studied inter-firm product development processes and found that interdependencies emerged repeatedly despite *ex ante* agreement on component interfaces. Brusoni and Prencipe (2006) examined a transition from an integral production process to a modular one, and found that it required integrating — rather than separating — the previous activities of design and production. But in a study of open source software projects, MacCormack et al. (2008) found that larger, more distributed teams tended to develop products with more modular architectures. Taken together, these studies indicate that the appropriate mapping between technological and organizational components remains an open question in general, and thus a design problem to be solved anew by every architectural innovator.

Compared to the mirroring literature, relatively little research examines the relationship between technological and organizational interfaces (Quadrants II and IV). However, the literature on technology standardization addresses this relationship implicitly by studying situations in which individuals work across organizational boundaries — either within or between firms — to coordinate on visible design rules (technological interfaces) that are supported by license agreements, standardization processes, or other coordination devices (organizational interfaces). For example, Rosenkopf et al. (2001) showed that the individual-

level social structure forged through the creation of mobile phone standards enabled knowledge flows and strategic alliances between their corresponding employers — in other words, that technological interfaces can enable the formation of organizational ones. The case of Sun Microsystems and Java illustrates the opposite relationship: Sun engaged in extensive institutional entrepreneurship (Garud et al. 2002) to win acceptance of its Java platform as a *de facto* technology standard, even after withdrawing twice from *de jure* standardization processes (Egyedi 2001).

Although firms in many industries face strategic decisions about technological and organizational architecture, we believe these concerns are especially salient in the context of digital systems. Digitalization not only makes products and services more malleable, but also makes it possible to radically reconfigure their design and production (Yoo et al. 2008). This flexibility confers unprecedented freedom to arrange both technological and organizational components in new ways, but existing theory offers limited advice on how to use this freedom effectively. Interfaces play a heightened role in digital systems for a similar reason: in contrast to analog systems, which typically tolerate a certain amount of variance in component interactions, every bit is potentially significant in a digital system — which means that creators of digital interfaces work in a vast and largely uncharted design space. While a comprehensive theory of architectural strategy lies well beyond the scope of this paper, our framework draws attention to a novel set of concerns, namely the interactions between the four architectural perspectives. These interactions are the focus of the next section.

4. Architectural Strategy in Computing Platforms

To illustrate how architectural strategy can shed light on the interactions among design choices in digital systems, we apply our framework in the context of a particular type of system architecture, namely computing platforms (Gawer and Cusumano 2002, Gawer 2009).

There are at least two paths for platform leaders to profit from the success of their platforms: by serving as a systems integrator, as IBM did with its mainframes, or supplying key system components, as Intel and Microsoft did for personal computers (Bresnahan and Greenstein 1999, Gawer and Henderson 2007). Platform architectures create value by supporting the distributed production of components whose integration is governed by publicly documented interfaces (West and Dedrick 2000). A recurring challenge in platform stewardship is to stimulate the production of enough complementary components to sustain a vibrant "ecosystem" of users and component developers (Iansiti and Levien 2004, Evans et al. 2006). A key tension arises from the fact that platform sponsors need to attract enough outside complements to benefit from network effects, while maintaining sufficient architectural control to capture economic value and coordinate the evolution of the platform (Morris and Ferguson 1993, West 2003, Boudreau 2006).

We use our framework to contrast and compare critical episodes in the evolution of three influential platforms over the past 30 years: the IBM PC, Sun's Java technology, and Nokia's smartphone products. There are important parallels across the three cases. All three involve a systems integrator that was the clear leader in its respective field: desktop computers, Internet servers, and mobile phones. In all three cases, the focal firm created and evolved a digital architecture both to win adoption for its products and reshape the industry structure to its advantage. All three firms balanced proprietary control against the strategic use of "openness"

to attract complements and adopters, and all three adjusted their product and organizational designs in response to competitive pressures.

At the same time, there are crucial differences. The cases are drawn from different decades, each providing lessons for players in the next. The original IBM PC was targeted at small businesses, while Nokia's smartphones were sold to affluent consumers and large enterprises; Sun's Java was a "middleware" technology that allowed other firms to create applications for Web browsers and Internet servers. All three platforms succeeded in achieving widespread adoption, but had varying degrees of success in generating financial returns for their original sponsors.

The cases are presented in chronological order with respect to the critical incidents we focus on: the 1981 introduction of the IBM PC and subsequent entry of IBM-compatible "clones," Sun's efforts to standardize the Java platform from 1997 to 1999, and Nokia's 2008 decision to acquire and partially spin off the maker of its Symbian operating system.[4] Table 1 summarizes the specific design decisions made by each firm. The decisions are classified into quadrants based on the key architectural elements involved in each. Our preliminary analysis of these three cases is presented below.

4.1 The IBM PC Revisited

The well-known case of the IBM PC provides an opportunity to revisit the tensions between technical and organizational design decisions in the period leading up to the release of the original Model 5150 PC in August 1981. These tensions played a role in the emergence of the PC "clone" market in the mid-1980s, which in turn led IBM to reverse a number of key

[4] Each case is drawn from news articles and (in the case of the IBM PC) book-length accounts of the product and organizational strategies of the focal firms. Due to space limitations, only quoted sources are cited in the text; the full list of references is available on request.

design decisions in its ultimately unsuccessful PS/2 product line. To show how these decisions are coded in our framework, we label them with bold Roman numerals indicating quadrants in Figure 1 and the corresponding cells in Table 1.

Developed in less than a year due to a mandate from IBM's Corporate Management Committee, the IBM PC's product architecture was based almost entirely on off-the-shelf components (**I**). IBM designed the PC around an existing processor chip, the Intel 8088, and contracted with external suppliers for disk drives, power supplies, circuit boards, and other critical components. As a result, most of the PC's design could be easily replicated by competing firms (**II**), with two notable exceptions: the IBM BIOS, a custom-designed chip that provided an interface between software programs and hardware components, and the computer's operating system, PC-DOS, which was sourced under contract to Microsoft but actually derived from software Microsoft acquired from a third party.

For IBM, the PC also reflected a novel set of organizational design choices. The product was developed far from the company's headquarters — and outside its normal processes — by a small team based in Boca Raton, Florida. Even for parts that were supplied by other IBM units, such as the keyboard, the PC team demanded competitive bids and treated these units no differently than outside suppliers (**III**). Microsoft was an exception: IBM engaged in extensive collaboration to help the tiny Seattle firm meet its exacting quality requirements, including fixing code errors and writing documentation. Conversely, Microsoft contributed to key design decisions concerning the BIOS interface, on which its code depended. Frequent interactions between the two development teams occurred both face to face and via an electronic mail system established for the project. Although IBM's interactions with its suppliers were covered under extensive nondisclosure agreements and detailed procurement contracts, IBM owned few

intellectual property rights to the PC design, with the exception of copyright and trade secrets related to the BIOS (**IV**). In particular, Microsoft retained control over the DOS source code and the right to market its own version of the product, MS-DOS, which it soon licensed to IBM's competitors.

Taken together, IBM made a distinctive set of design choices: modular architecture (**I**), public interfaces (**II**), outsourced components (**III**), and permissive licensing (**IV**). Echoing the consensus view among technology strategists, Brandenburger and Nalebuff (1996, p. 155) suggest that IBM erred not in any of these choices individually, but in their combination:

> IBM's real error was pursuing the outsourcing [**I** + **III**] and open-architecture [**II** + **IV**] stories together. Had it stopped at bringing in Intel and Microsoft, and not given up control of the hardware portion of the business, it would have remained in a strong position. Had it kept control over the chip and operating system technologies, then, despite cloning of the hardware, it would still have been in a strong position. Either approach might well have been effective. But outsourcing together with opening the architecture was a mistake. It's a case of two rights making a wrong.

Note that this tension is *not* the stereotypical clash between engineers and their pointy-haired bosses. The fault line spans the technological and organizational domains, dividing interfaces from components. In other words, IBM's design decisions were well aligned with respect to technological and organizational components (modular, outsourced) and also with respect to technological and organizational interfaces (public, permissive). The problem was that in pursuing both pairs together, IBM made it hard to maintain architectural control and thus to sustain its ability to capture value from the system.

Indeed, IBM overestimated its ability to deter rivals from producing compatible substitutes, and legal IBM-compatible "clones" became available soon after the PC's release, most notably the Compaq Portable in 1982. Over the next five years, the rise of the clones pushed IBM's share to less than half of the market for IBM-compatible PCs. Determined to

reassert control of the IBM PC ecosystem, in 1987 IBM sharply reversed course for its PS/2 line of computers. The company introduced a range of innovations that were incompatible with the de facto standards that had emerged around the original PC design, including the OS/2 operating system, the Micro Channel system bus, the Token Ring network protocol and the "PS/2" keyboard interface (**II**). All components except OS/2 were developed inside IBM (**I**), and competitors could employ the technologies only under royalty-bearing licenses (**IV**).

Ironically, these design moves not only failed to resolve the original tensions in IBM's architectural strategy, but created new tensions as well. The more expensive IBM-controlled components were rejected by PC makers, suppliers of complementary products, and ultimately by computer buyers. IBM eventually abandoned most of the PS/2 technologies, reverting to the *de facto* standards used by its rivals. In doing so, it effectively ceded platform leadership to Microsoft and Intel, as the "IBM PC" platform became known as the "Wintel" platform. IBM finally exited the market with the sale of its PC division to China's Lenovo Group in 2005.

4.2 Sun: The Java Wars

Our second case example concerns Sun Microsystems and its Java technology, an architectural innovation that changed the linkage between software applications and operating systems. Like IBM's PC, Java posed a complex design problem spanning the four architectural perspectives of our framework. To illustrate the cross-quadrant tensions that arise later in a platform's evolution, we focus not on the initial release of Java in 1995, but on the period from 1997–99. During this time, Sun initiated and then abandoned two efforts to standardize the Java platform, finally creating its own Java Community Process to govern the evolution of the technology.

Java is both a programming language and a software platform that allows developers to write programs that run without modification on a variety of computer systems — a feature Sun called "Write Once, Run Anywhere." Sun introduced a component, called a Java Virtual Machine (JVM), which functioned as a new architectural layer that mediates interactions between application programs and lower-level hardware and software platforms (**I**). Through the company's own engineering efforts and agreements with licensees, Sun ensured that JVMs were available for all major operating systems, including Microsoft Windows. Java programs could also be run inside (or alongside) web browsers and servers, as well as a range of small devices including smart cards, phones, and PDAs. Java achieved a high degree of platform independence by providing a common set of application programming interfaces (APIs) across these diverse environments (**II**). Although Java was by no means the first attempt at decoupling programs from their surrounding hardware and software components, its wide acceptance — accelerated by the growth of the commercial Internet in the late 1990s — posed both opportunities and threats for the rest of the computer industry.

Sun Microsystems was also known for its loosely coupled organizational architecture, which featured semiautonomous operating units ("planets") revolving around a central coordinating organization. Consistent with this pattern, Sun established a standalone unit called JavaSoft to develop and market its Java technologies and products (**III**). This arrangement buffered JavaSoft from conflicts with Sun's hardware and software businesses, whose enthusiasm for Java's cross-platform value proposition was tempered by proprietary interest in their own product lines. In addition to its product development responsibilities, JavaSoft served as the nexus of Sun's licensing agreements with virtually every major hardware and software vendor in the industry, including Microsoft, Netscape, IBM, Oracle, Apple, and Hewlett Packard

(**IV**). Sun used these licenses, along with its ownership of the Java brand, to retain control of the Java platform and stave off fragmentation — both accidental (due to incompatibilities introduced while porting the JVM to a new operating system) and deliberate (as in the case of Microsoft, which added its own APIs that tightly coupled Java applications to Windows while omitting some of Sun's cross-platform interfaces).

In contrast to the cross-quadrant tensions IBM experienced with the PC, which arose largely from design decisions made within the company, Sun's challenges with Java stemmed from opposing external forces. On one hand, Sun promised as early as 1996 to submit Java to an international standards body. This move was intended to assuage fears among Java licensees that Sun would in effect become another Microsoft by retaining control over the key interfaces of an important new platform (**II**). But Microsoft itself was determined to "embrace and extend" Java by tying its own JVM implementation closely to the Windows platform. Sun argued that its contracts with licensees offered a more effective way to protect the integrity of the platform than the weak enforcement mechanisms available to standard-setting organizations (**IV**). To this end, Sun sued Microsoft for breach of contract in October 1997.

By late 1999, with the Microsoft litigation still pending, Sun had withdrawn both of its Java standards submissions (first to ISO, the International Organization for Standardization, then to a European organization called ECMA). But ending its formal standardization efforts led to increased the pressure from other licensees, notably IBM and HP, to relax control over Java in other ways. Sun's first efforts in this direction, the Java Community Process (JCP) and Sun Community Source License (SCSL), were widely viewed as insufficient when they were announced in 1998. However, a second version of the JCP introduced in 2000 gave outside expert groups more autonomy to influence Java's evolution, effectively creating new

organizational components in the Java ecosystem (**III**), and Sun gradually moved toward full open source licensing, culminating in the release of the core Java platform under the GNU Public License in 2006 (**IV**).

Even while it was grappling with issues of platform governance, Sun faced an overlapping set of tensions in its efforts to make money from Java. Here again, architectural decisions in different quadrants often undermined Sun's objectives rather than reinforcing them. Although Sun experimented with various licensing models for Java (**IV**), the JVM was always free to end users, and corporate license revenues never covered Sun's development costs. The company also made several acquisitions in 1998–99 in an effort to benefit from the growth of complementary product categories (**III**). In particular, Sun acquired a startup company, NetDynamics, and formed an alliance with America Online to jointly develop Java-based products based on Netscape's application server software. Also in 1999, Sun consolidated its software-related business units under a single software division to help increase their profitability. But integrating so many products with overlapping functionality proved to be a formidable challenge (**I**), and Sun was never able to capture market leadership from BEA (acquired by Oracle in 2008) and IBM. Thus, despite the fact that Sun's ability to capture value from Java was not undermined by cloning in the same way as the IBM PC, Sun was similarly unable to monetize its control of the Java interfaces (**II**).

4.3 Nokia: Symbian Inside

Our third case illustrates the contrasting challenges of managing a platform across firm boundaries. To develop products for a new category of mobile devices, Nokia sponsored the creation of Symbian, a new company intended to supply software platforms for leading mobile phone vendors. This strategy helped Nokia to lead the product category, but in response to the

technical and organizational design choices of its fragmented competitors, Nokia acquired Symbian and dramatically liberalized the licensing terms for its technology.

In the mid-1990s, a new class of mobile computers emerged, variously called PDAs (Palm Computing), handheld computers (Psion PLC), or handheld PCs (Microsoft). Each of the companies built a platform and managed an ecosystem similar to those created by earlier computer makers. Beginning in 1996, these and other platforms formed the basis for experiments marrying a mobile phone and computer, creating a new category of mobile phones that came to be known as "smartphones." To develop a new platform specifically for smartphones, in 1998 Psion transferred software and staff to a newly created London-based company, Symbian Ltd.

Perhaps unique to any commercial computer platform, the operating system and its user interface were developed as separate components by separate companies (**I**). In fact, three different Symbian customers each developed their own interface: Nokia (the S60 UI for internal use), NTT DoCoMo (MOAP), and UIQ, a Symbian spinoff acquired by Ericsson (later Sony Ericsson). Each of these companies provided APIs to third-party software developers, in addition to the APIs Symbian provided to interface developers and other makers of complementary hardware and software (**II**).

Symbian was co-owned with major mobile phone vendors, who provided both equity capital and royalties for using the Symbian OS (**III**). Symbian OS was licensed on a per-unit royalty basis to makers of mobile phones. Fearing knowledge leakage and competition from its customers, Symbian originally withheld source code for key components from its owners and other licensees, but over time migrated to providing complete source code under trade secret restrictions (**IV**). In 2006, the Symbian OS platform held 67% of the global smartphone market, well ahead of second-place Windows Mobile (14%).

By 2007, Nokia had consolidated its mobile phone development onto two platforms: S40 for its mass-market phones, and S60 (plus Symbian OS) for its high-end phones. It was the world's top mobile phone maker with a 40% global share. Nokia was both Symbian's largest shareholder (47.9%) and customer, with 53% of all smartphone sales and more than 80% of Symbian handsets.

However, the company faced unexpected market challenges. On the one hand, its vertically integrated rivals — Research in Motion (BlackBerry) and new entrant Apple (iPhone) — could more easily coordinate operating system software and handset design under one roof. On the other hand, two rival organizations — the LiMo Foundation and the Google-led Open Handset Alliance — had recently announced competing Linux-based smartphone platforms available to handset makers as royalty-free open source software.

In response, in 2008, Nokia announced it was spending €264 million to buy the shares of Symbian that it did not own. The majority of Symbian's 10,000+ employees, the software development group, became part of Nokia's handset R&D group (**III**) and continued to be the main developers of S60 and Symbian OS components (**I**). In early 2009, Nokia created the Symbian Foundation as a nonprofit open source foundation (**III**), which would combine the APIs of Symbian OS and the three UIs into a new integrated set of APIs (**II**). Nokia, Sony Ericsson and DoCoMo assigned all source code rights to the foundation, which promised to release the entire platform as royalty-free open source software (**IV**).[5]

[5] Efforts to convert previously proprietary software platforms to open source normally entailed a delay of several years, as developers worked to obtain redistribution rights or developed unrestricted replacements. Examples of such delays included OpenSolaris from Sun Microsystems and Eclipse from IBM.

4.4 Analyzing the Architectural Strategies

Each of the three platform sponsors made a series of architectural design choices and took concrete steps to implement those choices. Previous research has emphasized both the choices themselves (e.g., IBM's decision to use outside suppliers) and their realization through subsequent actions (e.g., Sun's efforts to standardize Java). However, we believe our framework provides a new perspective on these architectural strategies, illuminating both obvious and subtle interdependencies across the two dimensions of Figure 1: technology vs. organizations, and components vs. interfaces. In addition, the comparisons suggest both common themes across multiple platform contests and variation particular to individual contests.

One kind of interdependency is commonly known as alignment, congruence, or fit. Design choices are aligned if their effects are complementary or mutually reinforcing. Conversely, they are in tension if they undermine each other or are mutually opposing. A common source of tension in all three cases was the need to balance architectural control with openness, in order to achieve both widespread adoption and adequate profit.

In the case of the original PC, IBM's design choices were well aligned to benefit the company's rivals, but undermined its own attempts to sustain the profitability of the platform (**I** + **III** vs. **II** + **IV**, as discussed above). By the time IBM reversed course with the PS/2 line, it had lost the ability to dictate the design parameters of an "IBM-compatible" PC, causing its new proprietary interfaces to further undermine the company's market position. In the Java case, Sun faced a tension between fending off Microsoft's attempts to "pollute" the technology — which required tight architectural control and aggressive contract enforcement (**II** + **IV**) — and the demands of allied but independently powerful licensees to cede control to a neutral standards

body (**III** + **IV**). Managing this tension, in turn, undermined Sun's ability to develop a coherent product strategy in a market that the company itself created (**I** + **III**).

In contrast, Nokia's strategy with Symbian reflects lessons learned from both Sun and IBM. After facing pressure from two kinds of competitors — those more open (with royalty-free open source distribution policies) and more integrated (with phones and operating systems produced by the same company) — Nokia invested heavily to acquire the Symbian unit and integrate it into the company (**I** + **III**), while releasing the Symbian source code to competitors and complementors alike without royalties or disclosure restrictions (**I** + **IV**). While too soon to judge the results of this move, it appears likely to give Nokia the benefits of both integrated and open approaches, albeit at a high cost.

A more subtle type of interdependency relates to the sequence of strategic design decisions rather than design outcomes. To identify these patterns, we classified the platform design decisions summarized in Table 1 into 16 permutations, corresponding to single (atomic) actions in each of the four quadrants, plus 12 ordered pairs of actions across quadrants. Table 2 provides a stylized description of the pattern corresponding to each permutation of actions.

Some of these patterns are well understood, both in theory and practice. For example, platform sponsors often publish modular APIs, both to enable the independent provision of missing components (**II → I**) and also to facilitate the division of labor between the sponsor and a key supplier or complementor (**II → III**). Others are hard to generalize, but clear in the context of the cases. For example, IBM's permissive licensing regime made it easy for both rivals and complementors to enter the market (**IV → III**); IBM then tightened its control in the hope of reducing competitive pressure, but the industry dynamics proved too strong to reverse. Following the same design sequence, Symbian licensed is OS to partners to promote the growth of

Symbian-compatible smartphone suppliers; the change to an open source license was intended to facilitate further entry and adoption of Symbian OS in preference to competing technologies.

A common way that platform vendors seek to grow their ecosystem and influence is through formal standardization, as Sun attempted with Java. Most *de jure* compatibility standards are about institutionalizing interfaces through existing standards organizations (**II → IV**), but in some cases a new organization is created to create and control the dissemination of interfaces (**IV → II**). Similar dynamics exist in the open source world, such as when open source communities create nonprofit foundations to manage the interfaces between communities and firms (O'Mahony 2003; **III → IV**).

In summary, we believe the value of the framework is twofold. First, it provides a systematic way for researchers and managers to analyze the interdependent design choices among technological and organizational components and interfaces. Second, by enumerating the types of relationships that are logically possible, it can suggest patterns to look for which may not have been previously identified.

5. Discussion: Organizing for Architectural Strategy

In this paper, we combine a review of prior research and three brief cases to suggest two major changes to the extant understanding of product and organization design. First, linking to the principles of design research, we approach architectural innovation as an intentional set of design choices rather than an exogenous change. Second, we integrate four prior perspectives across two dimensions of architectural strategy: components vs. interfaces (as focal design elements) in the technological vs. organizational domains. Together, these contributions suggest

new opportunities for theory development, as well as ways for managers to harness architectural strategy for platform leadership.

5.1 Architectural Innovation as a Design Process

Complementing Henderson and Clark's (1990) conception of architectural innovation as an exogenous change to which an incumbent firm must respond, we consider such change as an endogenous process and focus on the larger set of firms that participate in it. Consistent with Simon's (1969) view of design processes, the outcome is not fully determined by optimizing behavior or market forces, but instead reflects the creativity of human designers as well as biases introduced by historical path dependence and heuristic search methods. Although these ideas have influenced theories of product and organizational architecture, efforts to synthesize across the quadrants of our framework have been limited, especially in the organizational domain (Galunic and Eisenhardt 2001 is an exception).

More broadly, this work is positioned within — and influenced by — the resurgent tradition of design thinking in research on organizations (Dunbar and Starbuck 2006). As Yoo et al. (2006, p. 227) concluded from their study of building architect Frank Gehry:

> Organization designing challenges the beliefs that managers need to adapt to or interpret uncertain environments, which sets constraints for their goal seeking. In organization designing, leaders see the environment as a constraint that must be accommodated in the design as a set of conditions that must be overcome or reinterpreted to enable form giving, and as a set of opportunities that can be taken advantage of in putting a remarkable artifact into the world.

A key benefit of viewing architectural innovation as an endogenous design process is that it provides a way to reconnect with the equally powerful tradition of evolutionary thinking in research on innovation (Nelson and Winter 1977), and specifically the notion of design evolution driven by agents who "see and seek value" (Baldwin and Clark 2000, p. 93). Architectural design decisions are part of a closed-loop feedback system, in which the selection of a design by the

market becomes an input into subsequent design decisions. This feedback may be negative for some participants, as when a market tips toward a dominant design and induces a shakeout among firms that bet on the wrong technology. Equally often in network-based industries (as with the adoption of computing platforms), critical design decisions induce positive feedback by opening up new parts of a design space for exploration and development.

5.2 Toward a Socio-technical Theory of Architectural Strategy

The framework summarized in Figure 1 offers a way to organize existing research on the elements of architectural strategy, and suggests new, unexplored relationships in designing system architectures that span the technological and organizational domains. It provides a broader perspective on the design of socio-technical systems than previously available to researchers focusing on product or organization design separately.

Prior studies have considered the four quadrants of the framework individually: technological components, technological interfaces, organizational components, and organizational interfaces. A considerable body of work has also examined two pairs of linked quadrants — the interdependence of product modules and interfaces (e.g., Ulrich 1995, Schilling 2000) and the hypothesized mirroring of technological and organizational modularity (Ferguson and Morris 1993, Sanchez and Mahoney 1996). While the problem of decomposing a system into modular components has been well studied in both domains, the framework highlights the largely unrealized potential of research on the relationship between technological and organizational interfaces. Researchers could consider how digital interface designs impact the structure and dynamics of inter-organizational coordination, and vice versa. For example, Dell Computer used IT first to support a direct distribution channel, then a build-to-order supply chain, and finally direct consumer ordering via an e-commerce system (Kraemer et al., 2000).

The case studies show how these relationships can be explored in a richer and more systematic way by considering not only the linkages between technological and organizational architecture, but also the interface and component perspectives on each. This approach provides a more nuanced view of the alignment — or tension — between design decisions, which encompasses existing research on structural mirroring between products and organizations (Quadrants I and III), but also linkages between interfaces (II and IV), hierarchical relationships within architectures (I and II, III and IV), and diagonal relationships (I and IV, II and III). Research involving more detailed study of a larger sample could inform managerial practice on the challenges of aligning strategies across quadrants. Similarly, the taxonomy of sequential design choices in Table 2 suggests processes by which firms elaborate their architectural strategies, but additional research could examine how firms actually do (and should) make such sequential decisions.

Other opportunities exist to explore these effects in the context of multi-business enterprises. Research on synergies obtained through resource reuse in diversified firms has discussed product knowledge (e.g., Tanriverdi and Venkatraman 2005) but not design process knowledge. The ability to coordinate product architecture with organizational and even industry architecture may be an important capability for firms producing digital innovations, but not one that has previously been studied. Another potentially valuable capability is that of coordinating business-unit level product and service architecture with corporate level acquisitions, spinoffs, and licensing decisions.

5.3 Improving the Effectiveness of Digital Innovation

The question of why some firms are more successful at innovation than others has been a core concern for both explanatory research and normative efforts to improve practice (Foster

1986, Christensen 1997, Liefer et al. 2000). The problem of effective innovation is magnified in industries undergoing digital convergence, where rapid change in the enabling technologies means that firms often face a narrow window of opportunity between the time an innovation first becomes feasible and its inevitable commoditization (Fine 1995).

Prior research has shown that a firm's chances of success will be improved through better understanding of markets and technology, better technical design skills, and better product implementation. For system products that integrate externally sourced components or depend on ecosystems of third-party complements, such as computer platforms, firms also require skills at developing and managing these external relationships. Such innovation efforts are becoming increasingly dispersed across countries and companies, enabled by digital collaboration technologies (Eppinger and Chitkara 2006).

By building upon and integrating prior research, this paper offers two new insights into improving the effectiveness of innovation around digital platforms. First, it is important to recognize the socio-technical nature of system architectures, and to consider the concerns of the four architectural perspectives both separately and together. Prior research suggests that this may be difficult, particularly for more established firms, because the exiting cognitive frames of decision-makers influence both their strategic choices and the success of those choices, in part by constraining the search space in which they seek a solution (Prahalad and Bettis 1986, Barr et al. 1992). For example, the perceptions of Polaroid managers based on 30 years of success in analog imaging limited the innovation strategies they considered for digital imaging (Tripsas and Gavetti 2000). Similarly, managers who focus on a subset of the design decisions involved in a given architectural innovation (e.g., licensing, standards strategy, or product development) may be blindsided by tensions between these decisions and others.

Second, cognition is not enough. To lead to more effective innovation, our case examples suggest that uniting the architectural perspectives must be accompanied by an architectural strategy that spans multiple dimensions in a coordinated way. Even if firms succeed at executing the various parts of such a strategy, it may be a daunting challenge to align the interests and behavior of the various stakeholders. This applies equally to firms that are fully integrated and those that practice open innovation (Gerstner 2002, Maula et al. 2006), and is a particular concern for large organizations that produce complex digital systems, such as IBM, Sun, and Nokia. Moreover, these challenges are heightened when coordinating among employees with diverse professional training, norms, motivations and vocabulary, who both by their background and responsibilities may have difficulty finding common ground. For example, if a product development group is organizationally distant from the business development group responsible for acquisitions, then the make-versus-buy decision for new product components may be slowed and possibly biased by high coordination costs. In some instances, the best way to overcome these costs may be through spinoff companies, as in the case of Xerox PARC (Chesbrough and Rosenbloom 2002).

5.4 Future Research

Naturally, such challenges are specific to the technological, organizational and industry context in which they arise, which increases the importance of moving past single cases to study larger, more generalizable samples. While studying complex interdependencies among technology and organizations is a difficult task, digital representations of both software architectures and organizational relationships are becoming increasingly available and subject to empirical analysis (e.g., MacCormack et al. 2006).

Possible topics for more extensive studies include:

- The prevalence of different mechanisms in each of the four quadrants, such as the use of licensing vs. other kinds of contracts as an example of inter-organizational interfaces.
- The extent to which firms in platform contests have stronger technological or organizational linkages between components (**I + III**) or interfaces (**II + IV**).
- For strongly correlated linkages, identifying the most common temporal sequence or direction of causality (e.g., **II → IV** vs. **IV → II**).
- Whether certain architectural strategy combinations (**II** + **IV**) or sequences (**II → IV**) help predict the success or failure of a firm's platform strategy.

We recognize that in order for studies like these to be done, additional research may be necessary to define and operationalize the constructs presented in this paper. We intend to further develop the framework to facilitate these efforts.

References

Alexander, C. 1964. *Notes on the Synthesis of Form*. Harvard University Press, Cambridge, MA.

Anderson, P., M. L. Tushman. 1990. Technological discontinuities and dominant designs: A cyclical model of technological change. *Admin. Sci. Quart.* **35**(4) 604–633.

Baldwin, C. Y. 2008. Where do transactions come from? Modularity, transactions, and the boundaries of firms. *Indust. Corporate Change* **17**(1) 155–195.

Baldwin, C. Y., K. B. Clark. 1997. Managing in an age of modularity. *Harvard Bus. Rev.* (Sep–Oct) 84–93.

Baldwin, C. Y., K. B. Clark. 2000. *Design Rules, Volume 1: The Power of Modularity*. MIT Press, Cambridge, MA.

Barr, P. S., J. L. Stimpert, A. S. Huff. 1992. Cognitive change, strategic action, and organizational renewal. *Strategic Management J.* **13**(Summer) 15–36.

Bogner, W. C., P. S. Barr. 2000. Making sense in hypercompetitive environments: A cognitive explanation for the persistence of high velocity competition. *Organ. Sci.* **11**(2) 212–226.

Boland, R. J., Jr., F. Collopy. 2004. Design matters for management. R. J. Boland, Jr., F. Collopy, eds. *Managing as Designing*. Stanford Business Books, Stanford, CA, 3–18.

Boudreau, K. 2006. How open should an open system be? Empirical essays on mobile computing. Unpublished doctoral dissertation, MIT.

Brandenburger, A. M., B. J. Nalebuff. 1996. *Co-opetition*. Currency Doubleday, New York.

Bresnahan, T. F., S. Greenstein. 1999. Technological competition and the structure of the computer industry. *J. Indust. Econom.* **47**(1) 1–40.

Brusoni, S., A. Prencipe. 2006. Making design rules: A multidomain perspective. *Organ. Sci.* **17**(2) 179–189.

Brusoni, S., A. Prencipe, K. Pavitt. 2001. Knowledge specialization, organizational coupling, and the boundaries of the firm: Why do firms know more than they make? *Admin. Sci. Quar.* **46**(4) 597–621.

Carlile, P. R. 2002. A pragmatic view of knowledge and boundaries: Boundary objects in new product development. *Organ. Sci.* **13**(4) 442–455.

Chesbrough, H., R. S. Rosenbloom. 2002. The role of the business model in capturing value from innovation: Evidence from Xerox corporation's technology spin-off companies. *Indust. Corporate Change* **11**(3) 529–555.

Christensen, C. M. 1997. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press, Boston.

Christensen, C. M., M. Verlinden, G. Westerman. 2002. Disruption, disintegration and the dissipation of differentiability. *Indust. Corporate Change* **11**(5) 955–993.

Clark, K. B. 1985. The interaction of design hierarchies and market concepts in technological evolution. *Res. Policy* **14**(5) 235–251.

Colfer, L. 2007. The mirroring hypothesis: Theory and evidence on the correspondence between the structure of products and organizations. Working paper, Harvard Business School, Boston.

David, P. A., S. Greenstein. 1990. The economics of compatibility standards: An introduction to recent research. *Econom. Innovation New Tech.* **1** 3–41.

Dunbar, R. L. M., W. H. Starbuck. 2006. Learning to design organizations and learning from designing them. *Organ. Sci.* **17**(2) 171–178.

Egyedi, T. M. 2001. Why Java™ was – not – standardized twice. *Comput. Standards Interfaces* **23**(4) 253–265.

Eppinger, S. D., A. R. Chitkara. 2006. The new practice of global product development. *Sloan Management Rev.* **47**(4) 22–30.

Ethiraj, S. K., D. Levinthal. 2004. Bounded rationality and the search for organizational architecture: An evolutionary perspective on the design of organizations and their evolvability. *Admin. Sci. Quart.* **49**(3) 404–437.

Ethiraj, S. K., D. Levinthal, R. Roy. 2008. The dual role of modularity: Innovation and imitation. *Management Sci.* **54**(5) 939–955.

Evans, D. S., A. Hagiu, R. Schmalensee. 2006. *Invisible Engines: How Software Platforms Drive Innovation and Transform Industries.* MIT Press, Cambridge, MA.

Farrell, J., G. Saloner. 1992. Converters, compatibility, and the control of interfaces. *J. Indust. Econ.* **40**(1) 9–35.

Fine, C. H. 1995. *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*. Perseus Books, New York.

Fixson, S., J.-K. Park. 2008. The power of integrality: Linkages between product architecture, innovation, and industry structure. *Res. Policy* **37** 1296–1316.

Foster, R. N. 1986. *Innovation: The Attacker's Advantage*. Summit Books, New York.

Galbraith, J. 1973. *Organizational Design*. Addison-Wesley, Reading, MA.

Galunic, D. C., K. M. Eisenhardt. 2001. Architectural innovation and modular corporate forms. *Acad. Management J.* **44**(6) 1229–1249.

Garud, R., A. Kumaraswamy. 1995. Technological and organizational designs for realizing economies of substitution. *Strategic Management J.* **16**(Summer) 93–109.

Garud, R., S. Jain, A. Kumaraswamy. 2002. Institutional entrepreneurship in the sponsorship of common technological standards: The case of Sun Microsystems and Java. *Acad. Management J.* **45**(1) 196–214.

Gawer, A., ed. 2009. *Platforms, Markets and Innovation.* Edward Elgar, Cheltenham, UK.

Gawer, A., R. Henderson. 2007. Platform owner entry and innovation in complementary markets: evidence from Intel. *J. Econom. Management Strategy* **16**(1) 1–34.

Gawer, A., M. A. Cusumano. 2002. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press, Boston.

Gerstner, L. V., Jr. 2002. *Who Says Elephants Can't Dance? Inside IBM's Historic Turnaround.* HarperBusiness, New York.

Haberstroh, C. J. 1965. Organization design and systems analysis. J. G. March, ed. *Handbook of Organizations*. Rand McNally, Chicago, 1171–1211.

Henderson, R. M., K. B. Clark. 1990. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Admin. Sci. Quart.* **35**(1) 9–20.

Hevner, A. R., S. T. March, J. Park, S. Ram. 2004. Design science in information systems research. *MIS Quart.* **28**(1) 75–105.

Iansiti, M. and R. Levien. 2004. *The Keystone Advantage: What The New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability.* Harvard Business School Press, Boston.

Jacobides, M. G., T. Knudsen, M. Augier. 2006. Benefiting from innovation: Value creation, value appropriation and the role of industry architectures. *Res. Policy* **35** 1200–1221.

Kephart, J. O., D. M. Chess. 2003. The vision of autonomic computing. *Computer* **36**(1) 41–50.

Kraemer, K.L, J. Dedrick, S. Yamashiro. 2000. Refining and Extending the Business Model with Information Technology: Dell Computer Corporation. *The Information Society* 16 (1): 5-21.

Langlois, R. N., P. L. Robertson. 1992. Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries. *Res. Policy* **21** 297–313.

Liefer, R., C. M. McDermott, G. C. O'Connor, L. S. Peters, M. Rice, R. W. Veryzer. 2000. *Radical Innovation: How Mature Companies Can Outsmart Upstarts*. Harvard Business School Press, Boston.

Lyytinen, K., Y. Yoo. 2002. Research commentary: The next wave of *nomadic* computing. *Inform. Systems Res.* **13**(4) 377–388.

MacCormack, A., J. Rusnak, C. Y. Baldwin. 2006. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Sci.* **52**(7) 1015–1030.

MacCormack, A., J. Rusnak, C. Y. Baldwin. 2008. Exploring the duality between product and organizational architectures: A test of the mirroring hypothesis. Harvard Business School Working Paper 08-039.

Maier, M. W., D. Emery, R. Hilliard. 2001. Software architecture: Introducing IEEE Standard 1471. *Computer* **34**(4) 107–109.

Marengo, L., G. Dosi, P. Legrenzi, C. Pasquali. 2000. The structure of problem-solving knowledge and the structure of organizations. *Indust. Corporate Change* **9**(4) 757–788.

Maula, M. V. J., T. Keil, J.-P. Salmenkaita. 2006. Open innovation in systemic innovation contexts. H. Chesbrough, W. Vanhaverbeke, J. West, eds. *Open Innovation: Researching a New Paradigm.* Oxford University Press, New York, 241–257.

McIlraith, S. A., T. C. Son, H. Zheng. 2001. Semantic web services. *IEEE Intelligent Systems* **16**(2) 46–53.

Morris, C. R., C. H. Ferguson. 1993. How architecture wins technology wars. *Harvard Bus. Rev.* (Mar–Apr) 86–95.

Murmann, J. P., K. Frenken. 2006. Toward a systematic framework for research on dominant designs, technological innovations, and industrial change. *Res. Policy* **35** 925–952.

Nadler, D. A., M. L. Tushman. 1997. *Competing by Design: The Power of Organizational Architecture*. Oxford University Press, New York.

Nadler, D. A., M. S. Gerstein, R. B. Shaw. 1992. *Organizational Architecture: Designs for Changing Organizations*. Jossey-Bass, San Francisco.

Nelson, R. R., S. G. Winter. 1977. In search of useful theory of innovation. *Res. Policy* **6** 36–76.

O'Mahony, S. 2003. Guarding the commons: How community managed projects protect their work. *Res. Policy* **32** 1179–1198.

Obitz, T., Mohan B. K. 2008. Enterprise architecture expands its role in strategic business transformation: Infosys enterprise architecture survey 2008/2009. http://www.infosys.com/newsletter/EA-survey/images/ea-strategic-business-transformation.pdf.

Orlikowski, W. 2000. Using technology and constituting structures: A practice lens for studying technology in organizations. *Organ. Sci.* **11**(4) 404–428.

Pimmler, T. U., S. D. Eppinger. 1994. Integration analysis of product decompositions. *Proc. 6th Internat. Conf. on Design Theory and Methodology.* Amer. Soc. Mech. Engineers, Minneapolis.

Prahalad, C. K., R. A. Bettis. 1986. The dominant logic: A new linkage between diversity and performance. *Strategic Management J.* **7**(6) 485–501.

Prencipe, A., A. Davies, M. Hobday, eds. 2003. *The Business of Systems Integration*. Oxford University Press, New York.

Rosenkopf, L., A. Metiu, V. P. George. 2001. From the bottom up? Technical committee activity and alliance formation. *Admin. Sci. Quart.* **46**(4) 748–772.

Saloner, G. 1990. Economic issues in computer interface standardization. *Econom. Innovation New Tech.* **1** 135–156.

Sanchez, R., J. T. Mahoney. 1996. Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management J.* **17**(Winter) 63–76.

Schilling, M. A. 2000. Toward a general modular systems theory and its application to interfirm product modularity. *Acad. Management Rev.* **25**(2) 312–324.

Simon, H. A. 1962. The architecture of complexity. *Proc. Amer. Philos. Soc.* **106**(6) 467–482.

Simon, H. A. 1969. *The Sciences of the Artificial.* MIT Press, Cambridge, MA, chap. 3.

Star, S. L., J. R. Griesemer. 1989. Institutional ecology, 'translations' and boundary objects: Amateurs and professionals in Berkeley's museum of vertebrate zoology, 1907–39. *Soc. Stud. Sci.* **19**(3) 387–420.

Staudenmayer, N., M. Tripsas, C. L. Tucci. 2005. Interfirm modularity and its implications for product development. *J. Product Innovation Management* **22**(4) 303–321.

Tanriverdi, H., N. Venkatraman. 2005. Knowledge relatedness and the performance of multibusiness firms. *Strategic Management J.* **26**(2) 97–119.

Thompson, J. D., ed. 1966. *Approaches to Organizational Design*. University of Pittsburgh Press, Pittsburgh.

Tripsas, M., G. Gavetti. 2000. Capabilities, cognition and inertia: Evidence from digital imaging. *Strategic Management J.* **21**(10/11) 1147–1161.

Ulrich, K. 1995. The role of product architecture in the manufacturing firm. *Res. Policy* **24** 419–440.

von Hippel, E. 1990. Task partitioning: An innovation process variable. *Res. Policy* **19** 407–418.

Weick, K. E. 1976. Educational organizations as loosely coupled systems. *Admin. Sci. Quart.* **21**(1) 1–19.

Weick, K. E. 2004. Rethinking organizational design. R. J. Boland, Jr., F. Collopy, eds. *Managing as Designing*. Stanford Business Books, Stanford, CA, 36–53.

West, J. 2003. How open is open enough? Melding proprietary and open source platform strategies. *Res. Policy* **32** 1259–1285.

West, J., J. Dedrick. 2000. Innovation and control in standards architectures: The rise and fall of Japan's PC-98. *Inform. Systems Res.* **11**(2) 197–216.

Yoo, Y., R. J. Boland, Jr., K. Lyytinen. 2006. From organization design to organization designing. *Organ. Sci.* **17**(2) 215–229.

Yoo, Y., K. Lyytinen, R. J. Boland, Jr. 2008. Innovation in the digital era: Digitization and four classes of innovation networks. Working paper.

Figures and Tables

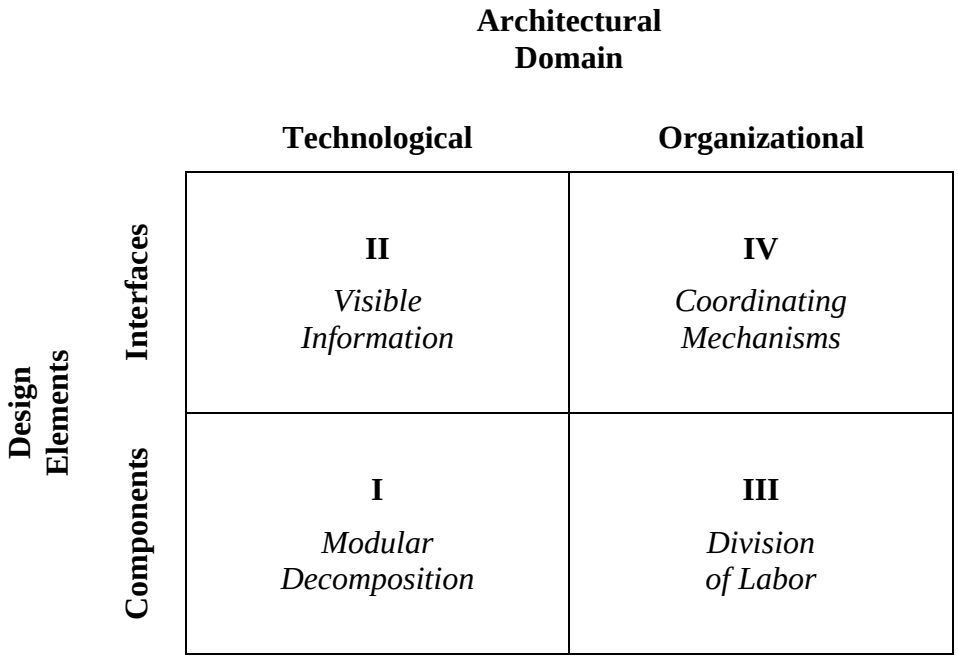

Figure 1: Matrix of architectural perspectives and key design concerns

Table 1: Technological and organizational design decisions by IBM, Sun, and Nokia

	Technological		Organizational	
	Components (Quadrant I)	***Interfaces*** (Quadrant II)	***Components*** (Quadrant III)	***Interfaces*** (Quadrant IV)
IBM PC (1981)	Mostly off the shelf, except DOS (sourced from Microsoft under contract) and BIOS (partly in-house)	Public – published in IBM PC Technical Reference Manual (including BIOS source code)	Designed, assembled by new business unit in Boca Raton; most parts from ext. suppliers, most sales through retail outlets	Intensive technical coordination with Microsoft; otherwise mostly arm's-length contracts awarded by competitive bidding
IBM PS/2 (1987)	Key innovations designed by IBM: Micro Channel bus, OS/2, Token Ring	De facto standards derived from IBM PC; proprietary interfaces for new technologies	Boca Raton unit now a full corporate division w/ substantial R&D; many parts still externally sourced	External coordination still mainly through sourcing contracts, except joint OS/2 work with Microsoft
Sun JavaSoft (1997–1999)	Sun's JVM designed to serve as a web browser component; Netscape supports fully while Microsoft integrates tightly with Windows and Internet Explorer, breaking compatibility	Sun marketing campaign urges "100% Pure Java" applications; cross-platform incompatibilities undermine promise of "Write Once, Run Anywhere"	Sun forms separate JavaSoft division, builds alliances with heavyweight partners (including IBM, Netscape, Oracle, Apple, HP, Novell)	Relationships with key licensees formalized in detailed contracts; Sun maintains exclusive control of platform evolution; no formal mechanism to mediate conflicts among Sun and licensees
Sun Software & Platforms Div. (1999–2002)	Java largely irrelevant in browsers but gains traction for server-side web applications and in mobile devices; Microsoft .NET emerges as a rival platform	Sun settles Microsoft lawsuit, abandons efforts at formal standardization; MS abandons Java in favor of its own C# language, .NET platform	Sun moves toward an integrated software organization, develops Java-based enterprise software under iPlanet brand acquired from Netscape/AOL	Java Community Process (JCP) governs platform evolution using expert groups w/ outside leads; Sun moves toward open source model with SCSL license
Symbian Ltd. (2007)	Operating system developed by Symbian, other elements by Nokia	Public APIs provided by Symbian to many handset makers; S60 APIs provided by Nokia; other APIs provided by Ericsson and DoCoMo	Symbian created to develop new operating system; Nokia has S60 and handset groups	Symbian OS and S60 licensed to third parties on per-unit royalty basis
Symbian Foundation (2009)	Nokia develops all components	Symbian and S60 interfaces combined	Nokia buys Symbian, combines Symbian and Nokia engineering teams; Nokia creates Symbian Foundation	Symbian OS being released as open source

Table 2: Sequences of strategic architectural decisions (classified by quadrant)

		Subsequent design choice			
		I. Tech. Components	*II. Tech. Interfaces*	*III. Org. Components*	*IV. Org. Interfaces*
Initial design choice	*I. Technological Components*	Create components	Creating APIs to encourage use of existing modules	Creating new division to produce a module	Create new licensing terms for own module
	II. Technological Interfaces	APIs enable production of new modules	Create, disseminate APIs	Complementors use APIs to make complements	Institutionalizing APIs, e.g., through formal standardization
	III. Organizational Components	Acquire (or spin off) the organization that creates a module	Outsiders help (or hinder) spread of APIs	Internal reorganization; acquisitions and spinoffs	Internal reorganization helps (or hinders) coordination with third parties
	IV. Organizational Interfaces	Licensing enables 3rd party module production	Create a new organization to control the creation and evolution of APIs	Licensing rules help or hinder entry	Licensing of APIs and/or implementations; joint development with suppliers