Bottlenecks, Modules and Dynamic Architectural Capabilities

HARVARD | BUSINESS | SCHOOL

Bottlenecks, Modules and Dynamic Architectural Capabilities

Carliss Y. Baldwin†

Working Paper

15-028

May 27, 2015

Third draft

† Harvard Business School
cbaldwin@hbs.edu

Bottlenecks, Modules and Dynamic Architectural Capabilities

Carliss Y. Baldwin

Abstract

How do firms create and capture value in large technical systems? In this paper, I argue that the points of both value creation and value capture are the system's bottlenecks. Bottlenecks arise first as important technical problems to be solved. Once the problem is solved, the solution in combination with organizational boundaries and property rights can be used to capture a stream of rents. The tools a firm can use to manage bottlenecks are, first, an understanding of the technical architecture of the system; and, second, an understanding of the industry architecture in which the technical system is embedded. Although these tools involve disparate bodies of knowledge, they must be used in tandem to achieve maximum effect. Dynamic architectural capabilities provide managers with the ability to see a complex technical system in an abstract way and change the system's structure to manage bottlenecks and modules in conjunction with the firm's organizational boundaries and property rights.

Keywords: architecture, architectural knowledge, dynamic capabilities, bottleneck, modularity, organization design, organization boundaries, property rights.

Bottlenecks, Modules and Dynamic Architectural Capabilities

Carliss Y. Baldwin

1 Introduction

Large technical systems made up of many interacting components are becoming more common every day. Digital information technology has made it possible to create ever larger technical systems with faster flows of material and information. At the same time, digital technology provides a common substrate for design, communication, and control. Spurred by the commonality of bits, a multitude of previously separate and small technical systems are expanding and converging towards one big system. (Yoffie, 1997; Baldwin and Clark, 2000; Weinberger, 2008; Brynjolfsson and McAfee, 2014.)

How do firms create and capture value in the increasingly large technical systems emerging today? In this paper, I argue that the points of both value creation and value capture are the system's bottlenecks. Bottlenecks arise first as important technical problems to be solved. Once the problem is solved, the solution in combination with organization boundaries and property rights can be used to capture a stream of rents.

The tools a firm can use to manage bottlenecks are, first, an understanding of the technical architecture of the system; and, second, an understanding of the industry architecture in which the technical system is embedded. In 1990, Henderson and Clark (1990) defined *architectural knowledge* as knowledge about the technical architecture of the products of a particular firm. However, today large technical systems have expanded to involve many firms, and thus managers must have a sophisticated understanding of

both technical and industry architectures. Although they are based on disparate bodies of knowledge, these architectural tools must be used in tandem to achieve maximum effect.

Architectural capabilities are an important subset of dynamic capabilities that provide managers with the ability to see a complex technical system in an abstract way and change the system's structure by rearranging its components (Teece, Pisano and Shuen, 1997; Pisano and Teece, 2007). Purposeful architectural change can then be used to create and capture value at different points in the technical system. In this paper I argue that value-enhancing architectural change arises through the effective management of bottlenecks and modules in conjunction with the firm's organizational boundaries and property rights.

The rest of the paper is organized as follows. Section 2 defines technical and industry architectures and explain how they are related. In Section 3, building on prior work, I describe two types of bottlenecks, technical and strategic, and show how bottlenecks are related to the modular structure of a technical system and to the organizational boundaries and property rights of firms in the system. Section 4 deals with the management of technical bottlenecks. Here the most basic question is whether to solve a particular technical bottleneck by focusing on only a few components or by controlling many components. Section 5 describes how different forms of modularity can protect or expose strategic bottlenecks. Section 6 describes how firms can invest to acquire architectural knowledge and dynamic architectural capabilities. Section 7 concludes.

2 Technical and Industry Architecture

To answer the question, how do firms create and capture value in large technical systems, it is first necessary to develop ways of describing such systems. A technical system is made up of conceptually separable components, each of which "does something" to contribute to the overall performance of the system (Arthur, 2009). The components of the system, taken together, are jointly complementary: the value of the system as a whole is greater than the value of the components disassembled (otherwise there would be no point in building the system).

Architecture is the term given to the abstract description of large technical systems. The concept of an "architecture" for man-made systems dates back to Herbert Simon. In his classic paper "The Architecture of Complexity," Simon proposed a theory of the structure of complex systems based on the concepts of hierarchy and modularity.[1] In brief, the architecture of a system defines its components, describes interfaces between components, and specifies ways of testing performance. All man-made systems, including all products and organizations, have architectures.[2] (Simon, 1962, 1981; Ulrich, 1995; Blaauw and Brooks, 1997; Baldwin and Clark, 2000; Whitney *et al.*, 2004; Fixson, 2005; Arthur, 2009.)

A complete architectural description of a man-made thing includes accounts of its *design structure* and its *task structure*. The design structure describes the finished artifact

[1] Simon used the term "near-decomposability" instead of modularity. Both concepts refer to an abstract pattern of linkages between components that can be arranged as a (nearly) block diagonal matrix. See Simon (1981, p. 212); Augier and Simon (2003); and Baldwin and Clark (2000, pp. 63-76).

[2] Many large technical systems are not entirely planned or designed. Such systems have architectures in the sense of "abstract descriptions," but some features of the system are not the result of planning or conscious design: instead they arise out of chance, history, mistaken beliefs, and serendipity. For example, cities, legal systems, and corporations all have both designed and undesigned features (Alexander, 1964; Arthur, 2009). In organization theory, these roughly correspond to the "formal" and "informal" aspects of the organization (Scott, 1987; Nadler and Tushman, 1997; Baker, Gibbons and Murphy, 1999; Gibbons, 2003; Gulati and Puranam, 2009; Gibbons and Henderson, 2012).

in terms of its design elements and their relationships. Design structure can be conveyed using words, pictures, blueprints, and diagrams. The task structure describes the procedures that must be carried out to bring the artifact into being.[3]

Engineers and product designers are concerned with the material structure of a product or process and the technical tasks needed to produce the product or carry out the process. Their focus is on the *technical architecture* of the system. However, to bring the components together and carry out the tasks specified by the technical architecture, a social structure must be overlaid upon the engineering task structure. Resources must be brought in, people recruited, tasks assigned, products transported and sold, and money collected. These actions take place in the social domain, mediated by organizations and markets.

As distinct from the technical architecture of a given product, Baldwin and Clark (2000) define the *contract structure* of an organization to include a description of its internal departments, compensation practices, boundaries, property rights, contracts, and transactions. Contract structures are part of the general architecture of a technical system: they are needed to assemble the resources needed to carry out the tasks of building and maintaining the system. But technical architectures and contract structures involve different types of expertise and are usually handled by different individuals. Specifically, the contract structure is generally in the hands of managers and lawyers, while the technical architecture is in the hands of product designers, developers, and engineers.

The term *industry architecture* is now used to refer to the aggregation of contract structures for firms in a particular industry (Jacobides, Knudsen and Augier, 2006; Pisano

[3] The design structure and the task structure correspond respectively to what Simon called the "state description" and the "process description" of a complex system. (Simon, 1981, p. 222).

and Teece, 2007). In what follows, I use the term "contract structure" to refer to the social architecture of a single firm and "industry architecture" to refer to the collection of contract structures in a given industry.

Technical architectures and industry architectures are related, but not in a deterministic way. Industry architectures exist to carry out the tasks specified by a given technical architecture. Those tasks in turn may be related in different ways: for example, they may be tightly or loosely coupled (Weick, 1969; Orton and Weick, 1990); or exhibit different forms of dependency (Thompson, 1967; Baldwin and Clark, 2000; Baldwin, 2008). They may be carried out by a close-knit team, by different departments within a single firm, or by different firms. Sometimes the technical recipe is sensitive to the organizational design, while at other times, a given recipe can be implemented equally well by many different forms of organization.

Although the mapping of technical architecture to industry architecture is not one-to-one, changes in the technical architecture of a system often beget changes in the industry architecture and vice versa. For example, taking advantage of a new product design (a new technical architecture) may entail setting up a new firm to make and sell the product (a change in industry architecture). Conversely, when a firm divests a division, it directly changes its industry architecture. The erstwhile division may then decide to eliminate some products and introduce others, thereby changing its technical architecture.

As an empirical matter, technical and industry architectures often change in tandem. When the technical architecture entails high levels of reciprocal task interdependency, vertically integrated firms become dominant players (Chandler, 1977;

Abernathy and Utterback, 1978; Fine, 1998; Fixson and Park, 2008). Conversely, as the technical architecture is broken apart into separate modules, firms become more specialized and the industry takes the form of a cluster or ecosystem (Langlois and Robertson, 1992; Sanchez and Mahony, 1996; Fine, 1998; Baldwin and Clark, 2000; Jacobides, 2005; Iansiti and Levien, 2004).

In the next section, I define bottlenecks, modules, and a firm's span of control in the context of technical and industry architectures.

3 Bottlenecks, Modules, and a Firm's Span of Control

In prior work, many scholars have argued that bottlenecks are key to understanding the direction and pace of technological change and to capturing value in large, complex technical systems. On the one hand, firms and individuals seeking to *create* value through technology are said to look for and resolve the technology's bottlenecks (Rosenberg 1967, 1972; Hughes[4] 1987; Langlois and Robertson, 1992; Ethiraj 2007; Arthur 2009; Adner and Kapoor 2010, 2014). On the other hand, firms wishing to *capture* value are advised to control bottlenecks, become a bottleneck, and beware of bottlenecks controlled by others (Teece,1986; Langlois, 2002, 2007; Jacobides, *et al.*, 2006; Pisano and Teece, 2007; Jacobides, MacDuffie and Tae, 2012; Jacobides and Tae, 2014; Henkel and Hoffmann, 2014).

But what is a bottleneck?

Bottlenecks Defined

In common usage, a bottleneck is a narrow place that obstructs a flow of water or traffic, for example. Thus in a road system, if all routes pass through a single stretch (say,

[4] Hughes (1983) uses the quasi-military term "reverse salient" to mean something very similar to "bottleneck."

a bridge or a mountain pass), and that part of the system is a source of congestion, then it is a bottleneck. More generally, a bottleneck is a component in a complex system whose performance significantly limits the performance of the system as a whole (see, for example, Goldratt, 1984).

Consistent with common usage, in what follows, I define a bottleneck as a critical part of a technical system that has no — or very poor — alternatives at the present time. There may be one or many bottlenecks in a given system but each has the dual properties that (1) it is necessary to the functioning of the whole; and (2) there is no good way around it. Thus to know that something is a bottleneck, the observer must see it in relation to a larger system, know what constitutes good system-level performance, and understand how the bottleneck constricts that performance.

In business, there are two types of bottlenecks, technical and strategic. With a *technical bottleneck*, the hindrance to performance derives from physical properties of the system. For example, in a railroad system, if there is no bridge over a river and goods must be taken onto barges and reloaded on the other side, then the river constitutes a technical bottleneck. It impedes the performance of the whole system and there is no good way around it.

Building a bridge can solve the problem of technical performance, but the owner of the bridge can charge a toll. The bridge plus the ability to control it then constitutes a *strategic bottleneck*. The former system of boats and barges is far less efficient, hence travellers and shippers have no good alternative except to use the bridge and pay the toll.[5]

[5] This assumes that no one builds a second bridge. However, if traffic only warrants one bridge, the owner of the first bridge can either set the toll or credibly threaten to lower the toll so as to make a second bridge unprofitable.

Technical Bottlenecks and Modules

There are three basic sub-types of technical bottlenecks in man-made systems. First, some necessary components of the minimally functional system may not exist at all. Let us call this a *sine qua non bottleneck*. Second, in systems involving flows, the capacity of the system will be constrained by the components with the least capacity relative to their frequency of use. Call this a *flow bottleneck*. Third, in systems requiring parts that match or fit together, the performance of the system as a whole will be constrained by mismatched components. Call this a *matching bottleneck*.

Sine qua non and flow bottlenecks both involve a mismatch of elements. In a *sine qua non* bottleneck, the mismatch is the non-existence of a critical solution to a subproblem. In a flow bottleneck, the mismatch is in flow capacity. Hence these types of bottlenecks can be viewed as interesting special cases of the general case of matching bottlenecks. However, the three sub-types have different implications for managerial action, thus it is useful to distinguish between them. (**Inset Box 1** gives further details and examples of the three bottleneck sub-types.)

Technical bottlenecks can be located within the system and mapped onto specific components in the system's technical architecture. An *intensive* bottleneck is caused by deficiency in one component or subsystem, whereas *extensive* bottlenecks are caused by deficiencies in several. (Because technologies are nested systems, the line between intensive and extensive bottlenecks is somewhat arbitrary. But the distinction is useful for strategy, as we shall see below.)

It is important to be clear that *bottlenecks are theoretically distinct from the modules of a technical system*. Technical bottlenecks are problems that exist whether the

designer wants them or not. (Mostly designers would prefer that bottlenecks not exist.) Technical bottlenecks are uncovered by identifying functional relationships between the characteristics of components (their capacity, size, strength, shape, etc.) and the performance of the system (good or bad).

In contrast, modules are a subsets of components that are tightly connected to each other, but only loosely connected with the rest of the system (Baldwin and Clark, 2000). Modular structure is revealed by tracking dependencies of the form "if Component A changes, Component B may have to change as well." By definition, a change within one module can be made without triggering changes in the others.

In a given system at a given time, the boundaries of modules may or may not correspond to location and extent of bottlenecks. However, modular structure is largely under the control of system designers and thus module boundaries can be drawn or redrawn to suit the designers' purposes. By definition, each component in a module is co-specialized to every other, thus in effect, components within modules are subject to very strong matching requirements. The operative question for designers is, where should these strong matching requirements be placed? As we shall see below, the firm's strategy toward bottlenecks provides a partial answer to this question.

Inset Box 1—Sub-types of Bottlenecks

Below I describe the three bottleneck sub-types in more detail and give examples.

Sine qua non bottlenecks. A complex technical system can generally be broken down into parts, each of which is necessary to the performance of the whole. For example, a digital computer at a minimum must have components that perform arithmetic computations; control the flow of instructions; store instructions and computations in memory; input instructions and data; and report results. Without all of these components, the computer cannot perform its basic function, thus each is *sine qua non* for the system. Similarly, an airplane at a minimum must have components to provide lift (the wings); thrust (the engines); a central framework (the fuselage); lateral and vertical stability (elevators, ailerons, rudder); a steering mechanism (same); and the ability to land (flaps, wheels). If one of these parts is missing, the flying machine cannot function in a useful way. Again each is *sine qua non* for the system.

Arthur (2009) describes the invention of novel technologies as a process of linking solutions "until each problem and subproblem resolves itself into one that can be physically dealth with—until the chain is fully in place" (p.110). The unsolved problems Arthur refers to are the *sine qua non* bottlenecks standing in the way of the creation of a new technical device. When the last or most difficult subproblem is solved, this is generally recognized as a breakthough, and the event becomes part of the lore of the technology. For example, the Wright brothers solved the critical subproblem of lateral control of a flying machine, and are credited with the invention of the first successful airplane. Arthur describes several other breakthroughs, including Ernst Lawrence's idea to use timed voltages to accelerate particles in a circular tube (the cyclotron) and Gary Starkweather's approach to the problems of modulation and pointing in laser printing (*ibid.,* pp 114-119).

Flow bottlenecks. Many complex systems involve flows. The flows may be water through an irrigation system, trains through a railroad, goods through a factory, electrons through a computer, instructions through a software program, messages through a communication network, transactions through a payments system, customers through a store, patients through a health care system, laws through Congress. All systems involving flow are subject to technical bottlenecks in the form of constraints caused by lack of capacity in some of the conduits of the flow.

For example, consider a rail system from point A to point E. Assume the system consists of five segments <*a, b, c, d, e*>. Segment *c* is a river for which no bridge exists: goods must be unloaded, taken on barges and reloaded on the other side. Shippers are only interested in going from A to E, thus the value of any subset of segments is zero. (Each segment is *sine qua non*, but since solutions exist for each segment, there is no *sine qua non* bottleneck.) Under these assumptions, the overall value of the system is proportional to the number of trains passing from A to E times shippers' willingness to pay for such transportation.

For purposes of illustration, let us assume that shippers value speed: the less time in transit, the higher their willingness to pay. Obviously, any technology that shortens the transit time for any segment will increase the value of the system. But on this view, the river does not stand out as a bottleneck. An hour saved in segment *a* is as good as an hour saved in segment *c*. All opportunities to reduce time-in-transit should be given equal consideration.

In contrast, the *number* of trains going from A to E in any given time period depends on the capacity of the system. Again for purposes of illustration, suppose only one train can occupy each segment at a given time. Now assume it takes one day for a train to cross segments *a, b, d,* and *e*, and one week to cross the river. The capacity of segments *a, b, d,* and *e* is 365 trains per year, but the capacity of segment *c* is only 52. Thus if two trains are dispatched from point A on successive days, the second will wait six extra days and take 13 days to cross the river. A train dispatched on day 3 will take 19 days to cross.

The capacity of the slowest segment constrains the capacity of the system as a whole. Mathematically, the capacity of the system equals the minimum capacity of its segments.[a] From this perspective, the river segment *is* a bottleneck. It is necessary to the functioning of the whole: a train cannot get from A to E without crossing it. Yet it impedes the performance of the whole. Futhermore, improving the capacity of any other segment has no effect on the capacity of the system as a whole. Only the bottleneck matters.

All systems involving flow are subject to capacity contraints. And all capacity constraints take the form of a "minimum" function in the general function determining system value.

Matching bottlenecks. Constraints on "matching" or "fit" are the third source of technical bottlenecks in man-made systems. For example, the power of the engine in an automobile must be matched by the power of its brakes. The strength of materials in a jet engine must match the force of the jets (Arthur, 2009).

Rosenberg (1969) describes a matching bottleneck caused by the introduction of high-speed steel alloys in the late 19th century:

[I]t was impossible to take advantage of higher cutting speeds with machine tools designed for the older carbon steel cutting tools because they could not withstand the stresses and strains … . As a result, the availability of high-speed steel for the cutting tool quickly generated a complete redesign in machine tool components—the structural, transmission, and control elements. (*ibid.* pp. 7-8)

In other words, once high-speed steel was introduced, the three other components of a machine tool—the frame, the transmission, and the controls—became technical bottlenecks. Until these components were redesigned to match the greater force of the high-speed cutting tool, machine tools could not take advantage of the value-creating possibilities inherent in the new steel alloy.[b]

In the case of machine tools, the inferior capabilities of weaker components made it impossible to take advantage of the capabilities of stronger components, and this mismatch constrained the performance of the system as a whole. In other cases involving matching, there is no question of "weak" or "strong," but components must be co-specialized if they are to deliver high levels of functionality. For example, in the design of a tablet computer or mobile phone handset, the technical challenge is to get components to fit within a given form factor. In the case of bicycle drivetrains (discussed below), the challenge is to have the components interact with high precision and minimal friction. These are different forms of matching bottlenecks.

---

[a] This statement holds when there is only one path through the system. More complicated functions are needed when there are multiple paths, but the minimum function is prominent in all of them. The minimum function may also constrain performance in systems that do not involve flow. For example, a chain is only as strong as its weakest link; a security system is only as good as its weakest portal.

[b] There is no generally accepted term for a component like high-speed steel that generates a new set of technical possibilities but only if the related technical bottlenecks are solved. Yet such components are common in the history of technology. We might call them generative components because they generate both opportunities and bottlenecks. In effect, generative components open up new design spaces, which can then be fruitfully explored. General purpose technologies (GPTs) are a fortiori generative components, but smaller innovations may be generative as well.

Strategic Bottlenecks and a Firm's Span of Control

Strategic bottlenecks are points of value capture and thus a source of rent in a technical system. A strategic bottleneck needs two things: (1) a unique solution to an underlying technical bottleneck; plus (2) control over access to the solution. For example, if a river is a technical bottleneck in a railroad system, a firm seeking to capture a strategic bottleneck must first build a bridge (the solution) and then prevent others from using the bridge unless they pay a toll (control).

In economics, the ability to exclude others from using a given resource is the classic definition of a property right.[6] (Alchian, 1965; Demsetz, 1967, 1988; Alchian and

[6] In law and philosophy, "rights" refer to entitlements conferred by the state and/or natural rights recognized under some ethical system. In contrast, economic usage of the term "property right" focuses on the ability to exclude and the locus of effective control. See, for example, Alchian, A. A., *Concise Encyclopedia of Economics*, http://www.econlib.org/library/Enc/PropertyRights.html: "A property right is the exclusive authority to determine how a resource is used." Whether control over the resource derives from power, community norms, or a legal system is secondary to the fact of control.

Demsetz, 1973; Grossman and Hart, 1986; Libecap, 1989, 1993; Hart and Moore, 1990; Hart 1995; Mahoney, 2004; Kim and Mahoney, 2005). Property rights in turn can be *de facto* based on power (my army controls the bridge; the chemical formula is a secret) or *de jure* based on the law (I own the bridge and police will arrest any trespassers; the chemical formula is patented). Property rights over the solutions to technical bottlenecks, whether *de facto* or *de jure*, form the basis of strategic bottlenecks. Property rights establish boundaries, hence they are part of the contract structure of firms.

Teece (1986) called the state of property rights, particularly intellectual property rights (IP), the "appropriability regime" pertaining to a resource, and noted that the regime might be weak or strong. In strong appropriability regimes, it is easy to exclude others from using a particular resource. In weak appropriability regimes, it is hard.

Under the resource-based view of the firm, a strategic bottleneck is a VRIN asset—valuable, rare, inimitable, and non-substitutable (Wernerfelt, 1984; Barney, 1991). VRIN status follows directly from the definition of a strategic bottleneck. First, it is *valuable* because it is a solution to a problem affecting the performance of the system, and the next best solution is far inferior. As the far-superior solution, the strategic bottleneck is more than simpy *rare*, it is unique. It must be *inimitable*, for if it can be copied, the copies will also solve the problem, and the bottleneck will cease to exist. By the same token, it must be *non-substitutable*, for if substitutes exist, they will also constitute solutions, and the bottleneck again will disappear.[7] The technical, legal and organizational setting surrounding the bottleneck will determine whether it is easy or

[7] Thus strategic bottlenecks have low mobility as defined by Jacobides *et. al.* (2006).

difficult to maintain the VRIN status of the technical solution and thus protect the strategic bottleneck.[8]

Property rights—the ability to determine who has access to superior solutions to technical bottlenecks—are thus critical to protecting a strategic bottleneck and claiming the associated rents. I define the *span of control* of a given firm to be the totality of its property rights over the components of a technical architecture. A firm can exercise control through a combination of physical control, secrecy, contracts, patents and copyrights. The components it controls by these means are deemed to be within its span of control.

In general, a firm's span of control coincides with its organizational boundaries. As long as it obeys the law, a firm can determine what goes on within its span of control. Its senior managers and board of directors can set policies, establish procedures, and delegate authority as they see fit (Freeland and Zuckerman, 2014). Both the technical architecture and the contract structure of the firm lie within their purview. In contrast, a firm may influence, but does not control what happens outside its span of control.

Some firms have a small or narrow span of control. Such firms perform few tasks inhouse, have few organization-specific skills or secrets, and little or no formal IP. Others have a large or wide span of control. They perform many tasks, have many organization-specific skills and secrets, and large IP portfolios.

Spans of control thus constitute a third dimension of architecture after bottlenecks and modules. While bottlenecks and modules are aspects of the technical architecture,

[8] In the resource-based view, "isolating mechanisms" prevent third parties from imitating a firm's technical and operational methods, thus maintaining the VRIN status of its resource base. Isolating mechanisms are ways of exercising control over access to technical and operational knowledge, thus are effectively ways of creating *de facto* property rights over the solutions to technical bottlenecks.

spans of control, which reflect organizational boundaries and property rights, are key parts of the industry architecture. Thus a given component might be an intensive technical bottleneck located within module A in the span of control of firm X. Firm X might then split module A into several subsidiary modules in order to accelerate the search for solutions to the technical bottleneck.[9] Alternatively, a group of components might form an extensive technical bottleneck, located in modules B, C, D and E, in the spans of control of firms W, X, and Y. Firm Z might unite modules B, C, D, E within its own span of control, dissolve the modular boundaries, solve the technical bottleneck, and control a strategic bottleneck as a result.

To create and hold a strategic bottleneck, a firm must understand the system's architecture at several levels—legal and organizational as well as technical. First, what is the solution to the technical bottleneck? Second, what property rights to solutions does the legal system grant, and how can these be secured? Third, how should an organization be formed to deliver the solution, protect it, and obtain payment for it?

The capacity to *see* bottlenecks, modules, and spans of control in a large technical system and the ability to *shift* them by changing the underlying modular structure and/or organization boundaries and property rights are an important subset of dynamic capabilities. To distinguish them from other dynamic capabilities, for example, human resource management or M&A capabilities, we may call them architectural capabilities. Combining architectural capabilities with a good strategy and resources can lead to competitive advantage and associated rents in the short run (Teece, 2014). However, bottlenecks, modules, and spans of control are not cast in stone: technical bottlenecks can

[9] The methodology and consequences of "splitting" are discussed in Baldwin and Clark (2000), Chapters 10 and 11.

be solved, strategic bottlenecks can be seized, and modules and spans of control can be redrawn. When all things are shifting, competitive advantage can only be sustained by continually investing to renew the firm's architectural knowledge and hone its dynamic architectural capabilities.

In the next sections I describe and give examples of ways in which a firm can change the modular structure of a system in conjunction with its property rights and organizational boundaries (its span of control) to gain competitive advantage. I first consider how to solve technical bottlenecks and then how to protect strategic bottlenecks.

3 Solving Technical Bottlenecks

A technical system can be envisioned as a network of components connected by dependencies or links (Steward, 1981; Eppinger, 1991; Baldwin and Clark, 2000). The dependencies and links involve transfers of material, energy or information plus payments for goods and services (Pahl and Beitz, 1995; Baldwin 2008).

Dependencies among components can be managed in two different ways. The first is for the designers to communicate in real time and work out by mutual adjustment how to handle each one. The second is for an architect to specify *ex ante* what dependencies will exist. This specification transforms a negotiated set of dependencies into a rule—a *design rule*—that is binding on all designers. In the presence of a mutually binding rule, the designers do not have to communicate with one another (Parnas, 1972; Parnas, Clements and Weiss, 1985; Baldwin and Clark, 2000). In effect negotiated dependencies require lateral coordination, while design rules provide hierarchical coordination.

Lateral dependencies are dense within modules and absent (or at least sparse) across modules. Design rules place more restrictions on the final architecture of the system than dependencies managed through real-time communication and mutual adjustment. With a rule, there is less room for give-and-take, and thus some new ideas may not be accommodated. Notwithstanding this fact, putting design rules in place does not necessarily hurt system performance. Architectural knowledge can be used to determine *which* design rules offer little or no harm to the overall system.

Corresponding to these two methods of coordination, a firm can manage technical bottlenecks in two different ways. In some cases, it will be advantageous to use design rules to split a module in order to *isolate* bottleneck components and maintain a narrow span of control in the underlying technical architecture. This move reduces the need for lateral coordination. In other cases, it will be advantageous to reduce the separation imposed by modular boundaries in order to *integrate* several components of a bottleneck within a single module. This move increases the need for lateral coordination and the firm will need to maintain a correspondingly wide span of control in the architecture.

Different types of firms will obtain different benefits from pursuing each approach (Teece, 1996). A strategy of combining modules might be feasible and attractive for a large firm but out of reach for a smaller one. Conversely, small firms, with no organizational hierarchy or vested interests, can often move swiftly to isolate technical bottlenecks and and thus claim strategic bottlenecks in promising new markets. Figuring out which approach is better requires deep knowledge and fine judgment—the bedrock of architectural capabilities.

In sections below, I discuss each of these strategies and give supporting examples.

Isolating bottlenecks within a small span of control

When a technical bottleneck involves only a few components it is often possible to change the dependencies via design rules to isolate the bottleneck components in a separate module. Isolating the bottleneck has a number of benefits, both technological and strategic. From a technological perspective, the bottleneck component can be redesigned to achieve higher performance without triggering negative interactions with components outside the bottleneck. As a result, the cost and time needed to instigate new rounds of technical improvement will decrease (Baldwin and Clark, 2000). And because it *is* a bottleneck, the performance of the whole system will improve as the performance of this part improves. The firm that owns the superior solution to the bottleneck can then charge users based on the incremental boost to *system* performance.

Isolating the bottleneck has strategic benefits, especially to (1) small firms with limited resources; and (2) second movers trying to overtake an incumbent with a more integrated architecture. Transaction costs are low at module boundaries (Baldwin, 2008) and non-bottleneck components are (by definition) substitutable. Thus the firm that owns the bottleneck solution can outsource design and production of non-bottleneck components. There will (again by definition) be several adequate alternatives for each non-bottleneck component, hence the focal firm should be able to source these components on competitive terms. The firm will operate with a "small span of control."

Two financial advantages arise from a small span of control. First, because it needs proportionately less capital to scale its operations, the firm with a small span of control will have a "sustainable growth" advantage (Higgins, 1977; Donaldson, 1984; Baldwin and Clark, 2006). This means that using only internally generated funds, the

firm can grow more rapidly than rivals with larger spans of control. Second, because its rate of return on capital is higher than its rivals', the firm can issue new equity on better terms, with less dilution to existing investors. This further increases the growth rates the firm can achieve without harm to its owners. In dynamic markets with network effects, which are subject to "tipping",[10] the ability to grow fast and profitably is a key determinant of success (Katz and Shapiro, 1985, 1992, 1994; Farrell and Saloner, 1986; Farrell and Shapiro, 1988).

In summary, the isolating strategy depends on achieving a superior, unique solution and exclusive control of an intensive technical bottleneck. The controlled part constitutes the firm's "span of control" in the overall system, hence this is a "small span of control" strategy (Baldwin and Clark, 2006). Architectural knowledge and capabilities are required to locate and solve the technical bottleneck, to modularize the technical architecture, and to set up the contract structure needed to obtain non-bottleneck components (Fine and Whitney, 1996; Mayer and Argyres, 2004; Argyres and Mayer, 2007; Blair, O'Connor, Kirchhoefer, 2011).

In the 1990s, Sun Microsystems and Dell Computer both pursued small span of control strategies with success. Highlights of these two cases are given below. Further details may be found in Baldwin and Clark (1997a, 2006).

Sun Microsystems. Sun's architectural capabilities were based on new methods measuring flows of instructions in computer hardware and software systems.[11] Using these methods, Sun's engineers identified memory management as a critical constraint

[10] Tipping is "the tendency of one system to pull away from its rivals in terms of popularity once it has gained an initial edge" (Katz and Shapiro, 1994, p.106).

[11] In their 1990 and 1994 textbooks, Hennessy and Patterson describe these new architectural methods in detail. Hennessy and Patterson's first textbook was published in 1990. Sun's founders, Andreas Bechtolscheim and Bill Joy learned them as Ph.D students in the early 1980s.

limiting the calculating capacity of engineering workstations. In effect, the principal bottleneck in the technical architecture of engineering workstations was a *flow bottleneck* from memory to the CPU. In the second generation of Sun's products, the engineers addressed this constraint with a specially designed and patented chip and associated code in the operating system (Baldwin and Clark, 1997a).

This technical strategy in turn allowed Sun to adopt a small span of control contract structure. Most of the components in Sun's systems were "cheap, off-the-shelf" parts. Sun also used Unix, a low-cost operating system, and Ethernet, a freely available set of network standards, instead of developing its own proprietary operating system or network software.

Sun's chief rival Apollo Computer adopted a very different technical strategy. Its engineers combined specific hardware designs with a proprietary operating system plus firm-specific network protocols and software. An Apollo system operated with Apollo components or not at all. All parts were co-specialized and produced by Apollo alone. Apollo had to design and manufacture all parts, thus it had relatively large span of control in the technical architecture.

Events proved Sun's strategy to be superior. By isolating the memory management bottleneck, Sun was able to build machines that were as fast as Apollo's but cost less. And because it manufactured fewer components inhouse, Sun inventories turned over more quickly, reducing its need for capital. Sun used the financial advantages associated with its small span of control to drive down product prices and to grow very fast. Apollo was unable to keep pace, and avoided bankruptcy only by being acquired by HP in 1989.

Dell Computer. Dell's architectural capabilities were rooted in the superior management of working capital. In the manufacturing of personal computers, Dell's managers saw that order-taking and assembly were a technical bottleneck, and sought to maximize throughput for these stages of the production process. Dell became formidably efficient in this narrow set of activities. It could produce similar goods at lower cost, in less time, with less working capital than its rivals. (Much of the time, Dell had negative working capital, thus the more it grew, the more cash came into the business.)

Like Sun, Dell used its financial advantage to attack its rivals by driving prices down. As it grew, moreover, Dell exercised more power over its non-bottleneck suppliers. It demanded and received generous trade credit, which gave it more cash and allowed the company to grow even faster. The other first-generation PC manufacturers could not match Dell's prices or growth and were forced to exit.

Interestingly, however, Dell never converted its original architectural capabilities into dynamic capabilities to manage architecture more broadly. Instead it defined innovation as a quest for higher return on invested capital (ROIC). When demand for PCs leveled off in 2009, and Chinese manufacturers, like Lenovo, became low-cost suppliers, Dell struggled to find new products and new distribution channels, but to no avail. In 2013, Dell became a private company through a leveraged buyout.

Integrating bottlenecks within a large span of control

Sun and Dell each succeeded by isolating a strategic bottleneck within a small span of control. They were able to employ this strategy because the underlying technical bottlenecks were intensive, that is, relatively narrow in scope. In contrast, some bottlenecks are extensive, involving a wide range of components all of which must be

redesigned to solve the bottleneck (Iansiti and Clark, 1993). Addressing extensive bottlenecks requires coordinated change in all the components in the bottleneck. One way to achieve such coordination is to break down modular boundaries and integrate the bottleneck components within a single large module.

In comparison to the isolating strategy described above, the integrating strategy requires more resources. Improving the joint performance of many interdependent components is harder than working on just a few, but the rewards in terms of better performance can be commensurately high. Also because the technical architecture is not modular, there will be few external points of attachment to the new system (Fixson and Park, 2008). Rivals must then recreate the whole system in order to compete.

Thus *as long as the performance of the integrated system is superior to that of comparable modular systems*, the integrated firm's strategic bottleneck will be strongly protected and virtually unassailable. However, when performance is equal in the eyes of customers, the modular strategy, which is more adaptable and less resource-intensive, will win, as was the case for Sun and Dell.

Firms pursuing an integrative strategy must design many components and perform many tasks within their boundaries. Necessarily, such firms will have large spans of control. When the bottleneck in question is related to the flow of materials in production, such firms will become vertically integrated. Indeed, as documented by Alfred Chandler, the "modern corporations" that arose in the 1880s succeeded by solving extensive flow bottlenecks via integrative, large-span of control strategies. More recently, the Japanese bike drivetrain manufacturer Shimano addressed a matching bottleneck related to gear shifting also by using an integrative, large-span of control strategy. Highlights of these

examples are discussed below. Further details can be found in Chandler (1977, 1986) and Fixson and Park (2008).

Chandler's "Modern Corporations." The firms Alfred Chandler labeled "modern corporations" grew up in response to opportunities made possible by high speed, mechanized production systems, rapid reliable transportation via railroads, and fast communication via telegraph (Chandler, 1986).[12] Given these new technologies, the previous systems of production and distribution were quickly shown to be shot through with bottlenecks. Chandler (1962) quotes Charles R. Flint, an organizer of US Rubber Company, on the advantages of vertical integration in this context:

> [T]he specialization of manufacture on a large scale … permits the fullest utilization of special machinery and processes, thus decreasing costs; the standard of quality is fixed and raised; those plants which are best equipped and most advantageously situated are run continuously …; there is no multiplication of the means of distribution …; terms and conditions of sales become more uniform, and credits through comparison are more safely granted; the aggregate of stocks [i.e., inventories] is greatly reduced …; greater skill in management accrues to the benefit of the whole, instead of the part; and large advantages are realized through comparative accounting and comparative administration. (pp. 34-35)

In other words, with the new machine technologies, the capacity of the manufacturing stages of production increased dramatically. But those advantages could only be realized if the plants were run continuously. Hence flow through the system had to be both high (close to capacity) and consistent.

The old methods of purchasing raw materials and distributing finished goods were capacity constrained, thus immediately became technical bottlenecks in the general system of mass production. Addressing these constraints uncovered subsidiary technical bottlenecks in inventory, transactions processing, and the management of trade credit.

[12] It seems ironic to all firms started in the late 19th and early 20th Centuries "modern," but this is the term that Chandler used.

Managers, a new professional group, were specifically trained to identify and address such bottlenecks and the entire system became more efficient as the technical bottlenecks were systematically removed.

In industries where the cost advantages of mechanized plants depended on maintaining a constant, high level of throughput, the techniques of flow management created such large cost savings that only vertically integrated enterprises remained competitive. In such industries, corporations with large spans of control administered by professional managers became the dominant form of organization. These companies are a prime example of the potential advantages of pursuing a large span of control strategy that integrates many components in a non-modular fashion within a single corporation.

Indeed any technical system whose performance rests on balanced flows will generate shifting bottlenecks that need constant oversight and coordination in real time. Today, however, the first line of monitoring and adjustment is usually done by computers such as numerical control systems, scheduling systems, and automated trading systems. Flows through computer mediated systems have increased accordingly. Transactions are also cheaper today and thus many high-flow, automated systems cross firm boundaries. Thus, in many industries, intermediate markets and firms with "vertically permeable" boundaries have replaced purely vertically integrated firms (Jacobides, 2005; Jacobides and Billinger, 2006; Luo et. al. 2012; Kapoor, 2013; Atalay, Hortascu and Syverson, 2014).

Shimano. The Japanese bike drivetrain manufacturer, Shimano, illustrates the solution of an extensive matching bottleneck via integration of component designs within a large span of control. A standard bicycle drivetrain is made up of six components. In

the early 1980s the standard drivetrain architecture was highly modular, and components made by different manufacturers could be mixed and matched at will. However, the standard drivetrain had two deficiencies with respect to shifting. First, the rider had to carefully adjust the chain after shifting (to center the chain on the sprocket). This took time (a problem for racing) and generally required taking a hand off the handlebars (a problem for mountain biking). Second, the rider had to be pedalling during the shift.

Shimano addressed these functional deficiencies by redesigning five of the six components to be co-specialized to very tight tolerances. No longer would any chain work with any derailleur, freewheel or hub: each component had to be Shimano-made. (Today, the Shimano Hyperglide™ cassette body has become the *de facto* industry standard and several manufacturers make compatible parts. However, it is claimed that suboptimal shifting results from mixing chains and hubs from different companies.[13])

The functional superiority of the new technical architecture turned the drivetrain from a technical bottleneck into a strategic bottleneck. Over the course of six years (1984-1990) following the introduction of the new drivetrain, Shimano's market share increased from approximately 20% to 50% in road bicycles and almost 80% in mountain bicycles. Most of Shimano's competitors were forced to exit. Shimano continues to dominate the industry today.

Do all extensive bottlenecks require integration?

Given these examples, it is natural to ask whether addressing extensive technical bottlenecks requires integration of components in a single module and a correspondingly large span of control. The answer is "no." As indicated, coordination in the design of

[13] http://sheldonbrown.com/k7.html#sram (viewed 8/30/14).

components can be achieved in two ways: through integration, which facilitates lateral communication, or through modularization, which achieves hierarchical coordination via design rules. Intensive bottlenecks almost always benefit from modularization, but extensive bottlenecks may go either way depending on circumstances.

The transition from traditional freight to containerized shipping is an example of an extensive bottleneck that was addressed via modularization and hierarchical coordination. To handle containers efficiently, new docking facilities had to be built at every major port. But the ports themselves operated independently and thus each one could be handled as a separate project (a module). Ships and railcars also needed to be redesigned but they too could be produced independently. All that was needed for the ships, railcars and ports to work together was a common standard for containers.

In effect, the common standard for containers served as parsimonious design rule, providing hierarchical coordination throughout the system. Once the industry arrived at a common definition of the standard container, work on other parts of the system could proceed in a modular fashion. Furthermore, because it was impossible to deny anyone access to the standard container design, it was hard to convert the separate technical bottlenecks in ports, ships and railroads into a system-wide strategic bottleneck. As a result, profits from containerization generally flowed to the module innovators and not to the original system innovators. (See Levinson, 2006, for additional details.)

In general, the less that is known about a given technical architecture and the location of its technical bottlenecks, the more advantageous is lateral communication and coordination (Monteverde and Teece, 1982; Monteverde, 1995). And, because transaction costs are high within modules, a large span of control is often the most cost-

effective way to organize the resulting non-routine problem-solving process (Nickerson and Zenger, 2004; Baldwin 2008; Colfer and Baldwin, 2010). However, lateral coordination is time-consuming and may not converge on a solution (Steward, 1981; Eppinger, 1991; Baldwin and Clark, 2000). In contrast, a modular architecture restricts alternatives but often saves time by limiting the degree to which changes in the design of one component can propagate through the system (Ethiraj and Levinthal, 2004).

It follows that integrating all technical bottlenecks within a single module is a good way to develop a truly novel technical system. However, if the system is large, it will take a very long time to complete. For example, the ATLAS particle detector at CERN, a very large and novel project, was designed in an integrated fashion but took eighteen years to go from concept to working system (Tuertscher, Garud and Kumaraswamy, 2014).

4 Protecting Strategic Bottlenecks

The value of a strategic bottleneck is proportional to the difference it makes in the value of the system relative to the next best solution. If the difference is large, then the rent stream to the bottleneck owner will be commensurately high. For instance, in the railway example above suppose building a bridge over the river cuts transit time on this segment from one week to one day. The capacity of the *system* would then increase from 52 to 365 trains per year. The owner of the bridge should be able to capture some fraction of the value thus created.[14]

[14] In general, given a significant capacity change, price changes will ripple through the system (Jacobides, Veloso and Wolter, 2014). There is no simple way to calculate precisely the value that will be captured by the bottleneck owner. The difference in system value using the old prices is an upper bound.

The firm that owns a strategic bottleneck must protect it to ensure that its value remains high. The main technical threat is the arrival of almost-as-good or better solutions to the underlying technical bottleneck. The main legal threat is loss of control of the solution. Intellectual property rights over the design of the solution are one way to defend a strategic bottleneck against both threats. Indeed, in strong appropriability regimes, a good patent or copyright may be all that is needed (Teece, 1986). But intellectual property rights, especially patents, always exist in the shadow of litigation and thus are uncertain (Hagiu and Yoffie, 2013). For example, of all patents that are litigated, approximately half are invalidated. (For this reason, Lemley and Shapiro (2005) call patents "probabilistic property rights.") What is certain is that the higher the value of the strategic bottleneck, the higher the legal costs of defending it and the greater the legal risks.

The modular structure of the system can also be used to protect a strategic bottleneck. To understand how modularity works in this context, we must first distinguish between modularity in production and modularity in use. *Modularity in production* refers to the way tasks of production are divided among separate groups. If the tasks and knowledge of different groups do not overlap and the groups do not communicate, the system is modular in production. (The groups may be business units in a single firm or different firms in a supply network.) In contrast, *modularity in use* refers to way the system is perceived by users. A system that can be configured by mixing and matching modules and adding or replacing modules is modular in use. A single, indivisible item is not modular in use. (Baldwin and Clark, 1997b; Sako and Murray, 1999; Baldwin and Clark, 2000; Baldwin and Henkel, 2014.)

In general, modularity in production protects strategic bottlenecks while modularity in use exposes them to competition. The next sections explain this apparent conundrum.

Modularity in production protects the bottleneck

One of the features of modules is that they "hide information" from other modules (Parnas, 1972; Parnas et. al., 1985; Baldwin and Clark, 2000). People working on module A don't have to know what's going on inside module B. Thus through modularity, it is possible to divide up knowledge about a bottleneck solution so that no single employee or supplier can replicate the whole. Modularity in production can thus protect a strategic bottleneck from being replicated by suppliers and employees of the focal firm (Baldwin and Henkel, 2014).

Liebeskind (1997), Henkel, Baldwin and Shih (2013) and Baldwin and Henkel (2014) show how firms have used modularity in production to protect organizational secrets. Highlights are given below: the original papers provide further examples and details.

Michelin Radial Tires. In the 1960s, Michelin used modularity in production to protect its strategic bottleneck in radial tires. According to Liebeskind (1997):

> During the 1960s, Michelin had a monopoly on knowledge relating to the production of high quality steel-belted radial tire manufacturing. In order to preserve this monopoly, manufacturing was divided into two separate processes: steel belt manufacturing, and tire production. Employees were not rotated between these manufacturing processes in a deliberate effort to restrict the number of employees that had knowledge about both processes. As a result, only a handful of very senior managers within Michelin were knowledgeable about the entire manufacturing process (p. 645).

Production and R&D in Countries with Weak IP Protection. In interviews conducted in Brazil and Mexico, Sherwood (1990) reports that, in many instances, only

the founder of a company would know the details of all steps in the production process, though key employees would know parts. In a more recent example, Schotter and Teagarden (2014) quote a former general manager of a Western-owned factory in China:

> Processes were broken down into those elements critical to the business so that no one single individual had access to all of them. As general manager, I was the only one who knew the entire manufacturing process. Extrusion dies, for example, were critical to the manufacturing process. These were kept overnight in a separate vault that could only be opened by one person, me, using a lock and key. Other staff members were never allowed to put the dies into the extrusion machine. (p. 45)

Costs of modularity in production. Using modularity in production to protect a strategic bottleneck can be costly, however. Modularizing production replaces lateral coordination with hierarchical coordination via design rules. As indicated above, this can be fatal for the production of novel systems, where unknown dependencies can result in conflicts that cause the whole system to crash. Also in systems where throughput is a key determinant of cost, excessive modularization can slow down the flow and increase inventories needed for adequate buffering (Zhou, 2014). Finally, some technologies are more amenable to modularization than others: for example, electronic chips are highly modular, whereas automobile parts must often be co-designed in order to work together in the same vehicle (Whitney, 2004; Luo et. al, 2012).

Modularity in use exposes the bottleneck

In contrast to modularity in production, modularity in use exposes a strategic bottleneck to a greater threat of imitation or substitution by third parties (Rivkin, 2000; Pil and Cohen, 2006; Ethiraj, Levinthal and Roy, 2008). To make a system modular in use, the system sponsor must offer modules as a separate products and provide ways for new modules to be attached to the system. Rivals then do not have to replicate the system

as a whole, but can concentrate on a single module. In contrast, integral, i.e., non-modular, products have no separate components and no external points of attachment. Rivals must then recreate the whole system or stay out of the game.

The case of the IBM PC illustrates the risks modularity in use poses to a strategic bottleneck. The case of the Apple Macintosh illustrates the offsetting risks of restricting users' options by making only non-modular systems.

The IBM PC and the risks of modularity in use. The original IBM PC had a highly modular technical architecture. Because IBM was a late entrant to the PC market and the PC division was resource contrained, the PC's managers adopted a small span of control strategy. They focused on controlling a small number of components, which they hoped would serve as strategic bottlenecks. They outsourced all other components of the PC system, including the central processor (from Intel) and the operating system (from Microsoft). (Ferguson and Morris, 1993.)

A key point of control for IBM was the so-called BIOS, the Basic Input Output System. A BIOS is very much like a bridge connecting peripheral devices to the central processor. Without using the BIOS, software written for the PC could not access peripheral devices, such as disk drives, printers, keyboards, and monitors. The IBM BIOS was protected by copyright, and also etched in the circuits of a read-only memory (ROM) chip. However, the chip itself was relatively small (35KB of memory).

Shortly after the IBM PC was introduced, Compaq and Phoenix Technologies both legally copied the PC BIOS using a practice known as "clean room reverse engineering."[15] With the reverse-engineered BIOS in hand, it was relatively easy for

[15] Clean room reverse engineering is a method of copying a design without infringing on copyrights or trade secrets associated with it. Typically one group of engineers will examine the behavior of an artifact

other manufacturers to create clones of the IBM PC. (Clones were substitutes that were fully compatible with IBM PCs, hence could run software and use peripheral devices developed for IBM machines.) By 1986, Compaq and numerous Taiwanese clone makers were flooding the market with IBM-compatible machines.

In this way, IBM lost control of a major strategic bottleneck in the PC's architecture. The BIOS was still an essential component, but, using reverse-engineered chips, Compaq and the clone manufacturers could make machines that worked equally well with the vast number of peripherals and software applications designed for IBM PCs. The BIOS was no longer rare, inimitable, and non-substitutable, and thus it ceased to be very valuable. Furthermore, as the clone makers grew, IBM lost the advantage of scale, and ceased being the low-cost producer of PCs. The PC division struggled to remain profitable until it was sold to Lenovo in 2004.

Apple Macintosh and the risks of integrality. Unlike IBM, Apple opted to make the Macintosh BIOS an integral part of its operating system (OS). According to one observer:

> [T]he Mac BIOS is very large and complex and is essentially part of the OS, unlike the much simpler and more easily duplicated BIOS found on PCs. The greater complexity and integration has allowed both the Mac BIOS and OS to escape any clean-room duplication efforts. This means that without Apple's blessing (in the form of licensing), no Mac clones are likely ever to exist. (Mueller, 2003, p. 28)

Apple had a correspondingly large span of control in the Macintosh's technical architecture. Users not only had to purchase their computers from Apple, but also their printers, disk drives, keyboards, and mice. Apple also wrote most of the application

---

and write up a detailed specification. Another group with no knowledge of the original design will then implement the specification. The implementation constitutes an independent invention, though its behavior mimics that of the original design.

software. Apple hardware and software were offered for sale at high markups, thus user options for the system were largely limited to high-cost items that Apple was able to supply.

In contrast, the IBM managers adopted what is now called an "open" platform strategy (Ferguson and Morris, 1993; Gawer and Cusumano, 2002; West, 2003; Boudreau, 2010; Baldwin and Woodard, 2011; Eisenmann, Parker and Van Allstyne, 2011; Gawer, 2014). They actively recruited hardware manufacturers to build complementary peripheral devices and software developers to write programs that would run on the PC. The PC's expandability via third-party hardware and software was prominently featured in its early marketing campaigns.

In this fashion, perhaps only half consciously, IBM tapped into a powerful set of "indirect network effects" (Church and Gandal, 1992; Rochet and Tirole, 2003). Indirect network effects arise when a particular institution (the platform) permits interactions between two or more types of participants, each of which has reason to value the presence of the other. In the case of computer platforms like the IBM PC, computer users have reason to value large numbers of compatible applications and peripheral devices because these increase their options and extend the useful life of their machines. At the same time, software developers and peripheral manufacturers have reason to value large numbers of users, because they face economies of scale in development and production. Thus computer users will be attracted to platforms with more peripherals and software, while peripheral manufacturers and software developers will seek out platforms with more users.

Platforms with indirect network effects often enjoy the dynamics of increasing returns to scale: as the numbers on one side grow, the numbers on the other side grow apace (Eisenmann, Parker and Van Allstyne, 2006; Evans, Hagiu and Schmalensee, 2006; Zhu and Iansiti, 2012). The result can be a winner-take-all outcome where the market "tips" and most participants converge on a single platform (Katz and Shapiro, 1994). IBM managers set loose this dynamic when they introduced the open, modular IBM PC in 1981. By 1987, IBM-compatible computers accounted for over 60% of the personal computer market; this figure rose to over 90% in the early 1990s.[16]

The upshot is that, whereas the Apple Macintosh was better protected from imitators by virtue of its non-modular BIOS and operating system, the IBM PC platform offered a more attractive value proposition to users and hardware and software developers. Users could purchase many compatible, low-cost hardware devices and software applications. Hardware manufacturers and software developers had the corresponding lure of many users. The market overwhelmingly voted for the IBM PC, and this standard dominated the personal computer industry long after IBM bowed out.

Best of both worlds? Apple's iPhone and iPad hardware devices

It is clear from this analysis that the best way to protect a strategic bottleneck is to make it modular in production but not modular in use. The Apple iPhone and iPad hardware devices are designed this way. Each device contains components made by many different manufacturers to Apple specifications, thus the designs are modular in production.[17] But each device is also sold as an single, indivisible, non-upgradable unit,

[16] http://arstechnica.com/business/2012/08/from-altair-to-ipad-35-years-of-personal-computer-market-share (viewed 10/16/14).

[17] See https://www.ifixit.com/Teardown/iPhone+5+Teardown/10525 (viewed 9/2/14) for a list of components of the iPhone 5 components and manufacturers.

thus the designs are not modular in use.[18] (In contrast, the software and music libraries are completely modular in use.)

Apple is thus engaged in a delicate balancing act. As long as the iPhone and iPad are perceived to be superior to more modular and configurable devices (such as most Android phones and other tablet and laptop computers), Apple can enjoy a strongly protected strategic bottleneck together with strong market demand. These are the "commanding heights" of a large technical system, and Apple's current profits are a testament to the value of such a position.

However, as Apple learned from the Macintosh, it must give users the options they truly value, or they will abandon the system en masse for one that is more open and flexible. A more open system in turn might unleash the power of indirect network effects, thus further undercutting Apple's strategic bottleneck. Apple thus gives users a wide range of software and content options, even as it restricts their hardware options. So far this hybrid strategy seems to be working.

5 Acquiring Architectural Knowledge and Capabilities

How does a firm acquire architectural knowledge and capabilities that allow it to solve technical bottlenecks and protect strategic bottlenecks in large and evolving technical systems? Large technical systems are generally not easy to understand (Perrow, 1984). Inconveniently, they tend to change via local adaptation and grow by accretion. Changes are often undocumented. The end result is a system no one understands, burdened with "technical debt" that stands in the way of constructive change. Such

[18] In contrast the application software is highly modular in use. See the discussion of user options below.

systems are often labeled "hairballs," or "great balls of mud," by the technicians who must coax them to perform.

Architectural knowledge of a technical system begins with seeing the system as a working whole, but also seeing that is made up of distinct components. The system as a whole has a purpose (Arthur, 2009) and its performance against its purpose can be measured in terms of effectiveness and efficiency. Each component in turn contributes in some way to the performance of the system as a whole. Observers gain knowledge about technical bottlenecks by studying each component in relation to the whole and assessing how the properties of a given component enable or impede the performance of the system.

In flow systems, technical bottlenecks give rise to delays and/or inventory buildup ahead of the bottleneck and slack capacity below it. However, given even moderately complicated flows, local adaptations and work-arounds often mask the true location of the bottlenecks (Goldratt, 1990). Generally it is necessary to track the progress of representative flow units through the system, recording the times spent in each processing stage or buffer. The observer must then aggregate over a typical mixture of paths to identify the points that place the greatest constraints on overall system capacity. Goldratt (1990) describes this process for manufacturing, retail services, and project management. Patterson and Hennessy (1994) describe how to track instructions flowing through computer systems.

Matching bottlenecks, by definition, involve several components, and thus are generally harder to identify than flow bottlenecks. Learning about the bottleneck and solving it tend to merge into the same trial-and-error process. For example, in the case of bicycle drivetrains, discussed above, Shimano engineers had to form the conjecture that

tighter coupling (a better match) between the derailleur, the sprockets, and the chain would make shifting more precise and faster. They then had to test that conjecture by redesigning and testing various configurations. Today, computer simulation and 3-D CAD modeling can shorten the experimental generate-test cycle, and thus greatly accelerate the process of finding a better match.

Solving a technical bottleneck also requires knowledge of how modular structure can be changed to isolate or integrate the components in the bottleneck. One builds knowledge of modular structure by asking the question, "If this component's design were changed, what other components might have to change as a result?" (Parnas, 1972; Baldwin and Clark, 2000). Once direct linkages of this type have been identified, they can be aggregated into a network graph and summarized in terms of a Design Structure Matrix (DSM) (Steward, 1981; Eppinger, 1991; Eppinger et al. 1994; Browning, 2001).

Unfortunately, direct linkages by themselves do not reveal the modular structure of the system. The DSM must be organized in particular ways to reveal hierarchy (chains of dependencies) and modules (cyclic groups of dependencies). Baldwin, MacCormack and Rusnak (2014) provide an algorithm for discovering and displaying the modular structure of large technical systems. In a study of over 1000 software codebases (in 17 families), they found that the great majority of these systems had a "core-periphery" structure, that is, there was one dominant cyclic group (the "Core"). However, the existence and composition of the Core and the location of other components in relation to the Core was not evident from the systems' documentation. Thus the modular structure of these software systems was hidden, even to the systems' own developers.

Finally, protecting strategic bottlenecks requires knowledge of organizational boundaries and property rights. In developed countries, property rights to physical goods—products, machinery and equipment, buildings and land—are usually fairly easy to ascertain and enforce. However, to an increasing degree, control of strategic bottlenecks rests on control of knowledge through secrecy and/or intellectual property rights. Thus to occupy a secure position with respect to a strategic bottleneck, a firm must know what parts of its knowledge are critical to maintaining exclusive control of the bottleneck solution. It must also know who might threaten the bottleneck by introducing a superior technical solution or by contesting the firm's claims to the underlying IP.

6 Conclusion

Large technical systems made up of many interacting components are becoming more common every day. Many of these systems, like tablet computers, smartphones and the Internet, are based on digital information technologies. Others, like the electrical grid, the financial payments system, and modern factories, rely on digital technologies. As indicated, digital information technology has made it possible to create ever larger technical systems and every faster flows. This paper asks the question: how do firms create and capture value in large technical systems comprising billions of components, millions of users, and thousands of firms?

To answer this question, it was first necessary to develop ways of describing such systems. The architecture of a technical system is an abstract description of what it does and how it works. The technical architecture includes both a design and a task structure and shows, from a technological perspective, how to make the system. A contract

structure is a social architecture that is overlaid on the technical architecture to assemble resources, assign tasks, transfer goods, and collect compensation. An industry architecture is the aggregation of contract structures within a given sector of the economy.

Within a large technical system, I've argued that the points of both value creation and value capture are the system's bottlenecks. In prior work, a number of scholars, including Rosenberg (1969), Teece (1986), Langlois and Robertson (1992), Langlois (2002, 2007), Baldwin and Clark (2006), Jacobides *et. al.* (2006), Ethiraj (2007), Pisano and Teece (2008), and Henkel and Hoffmann (2014), have similarly argued that bottlenecks are key drivers of technical change and sources of competitive advantage.

In light of this prior work, perhaps the most important contribution of this paper is to point out that bottlenecks come in two flavors. *Technical bottlenecks* are important technical problems to be solved. Once a solution is found, it may be combined with organization boundaries and property rights and become a *strategic bottleneck* and a source of rents.

A second contribution is to show how bottlenecks are related to modules, organizational boundaries, and property rights. Bottlenecks, both technical and strategic, exist within the modular structure of a given technical system. The system's modular structure in turn can be modified to better find and solve technical bottlenecks and protect strategic bottlenecks. Finally, a firm's span of control determines its organizational boundaries and property rights, and thus the extent of its control over various bottlenecks in the system.

Dynamic architectural capabilities give firms the ability to understand a large technical system coherently and change it in ways that are competitively advantageous.

In essence, these capabilities amount to the ability to find and solve technical bottlenecks and hold and protect strategic bottlenecks. Modular structure, organizational boundaries, and property rights are all tools the firm can use for this purpose. For example, modular boundaries can be shifted to isolate or integrate the components of technical bottlenecks. Modularity in production can be used to protect a strategic bottleneck: each agent will know part of the technical recipe, but only a few trusted agents will know the whole. In contrast, modularity in use exposes a strategic bottleneck to competition from third parties. However, modularity in use also creates user options, which may be critical to the firm's value proposition.

I close the paper by suggesting two areas of opportunity for future work.

First, this paper has only considered how a *single* firm can manage a *single* technical or strategic bottleneck. However, large technical systems have many bottlenecks at different levels of the technical design hierarchy. Also, it is now rare for large technical systems to be contained within the boundaries of a single firm. Instead, technically related activities are often distributed across many firms and involve other institutions, including standard-setting organizations and open source development communities. Further work is needed to understand how bottlenecks can be managed when knowledge and property rights are distributed among many independent agents.

A related theoretical problem is to develop tractable ways of modeling bottlenecks in large technical systems. Functions and functionality, which determine a user's inclination to buy and willingness to pay, are key to this characterization. In essence, what is needed is a mapping from system components to system functionality to system value.

The functional decomposition of systems is an important topic in many engineering disciplines, but it is difficult to translate the engineering models into concepts meaningful for economics and strategy (Suh, 1999; Hatchuel and Weil, 2009). In economics, Milgrom and Roberts (1990) developed a theory of complementarity based on the mathematical concept of "supermodularity" (Topkis, 1998).[19] Some value functions fit nicely in this framework, but others do not. Another approach is the construction of so-called NK value landscapes, but these are randomly generated structures, and have not to this point proved capable of representing real systems (Kauffman, 1993; Levinthal, 1997; Rivkin, 2000; Ethiraj and Levinthal, 2004; Ethiraj, Levinthal and Roy, 2007).

What is needed, I believe, is a representation that is powerful enough to capture the existence, location and nature of bottlenecks in real technical systems, but simple enough to support direct intuition and reasoning. Arthur's (2009) theory of technologies is one possible starting point. He conceives of technologies as hierarchies of interacting methods, each of which solves a problem and thus contributes to the value of the whole. This approach echoes earlier work on industry evolution and dominant designs by Tushman and Rosenkopf (1992), Tushman and Murmann (1998), and Murmann and Frenken (2006).

Five decades ago, the economy contained many separate and incompatible technical systems, managed independently by different firms. Today, through the evolution of digital technology, we are advancing towards a world comprising one large technical system with many interoperating and evolving parts. Architectural knowledge

[19] "Supermodularity" is a functional property of components wherein increasing contributions from one increases the positive impact of another. It is not related to modularity, which is a structural property of components.

provides a way of understanding large technical systems and dividing them up into coherent units of analysis. Dynamic architectural capabilities provide a comprehensive framework in which to manage bottlenecks, modules, organizational boundaries and property rights in pursuit of value and competitive advantage.

Acknowledgements

I thank David Teece and Sunyoung Leih for soliciting this paper and suggesting that it should focus on bottlenecks. I also thank my collaborator and co-author, Joachim Henkel: my thinking on strategic bottlenecks and modularity cannot be separated from our joint work. Likewise I thank Michael Jacobides for discussions over many years on the topic of industry architecture and value migration. Finally I thank Nuno Gil and his students for very insightful comments on an earlier draft. Errors and omissions are mine alone.

References

Abernathy, William J. and James Utterback (1978) "Patterns of Industrial Innovation," *Technology Review* 80:41-47.

Adner, R., & Kapoor, R. (2010). Value creation in innovation ecosystems: How the structure of technological interdependence affects firm performance in new technology generations. *Strategic management journal*, *31*(3), 306-333.

Adner, R., & Kapoor, R. (2014). Innovation ecosystems and the pace of substitution: Re-examining Technology S-curves. Mimeo.

Alchian, A. A. (1965). Some economics of property rights. *Economic forces at work*, 294-307.

Alchian, A. A. & Demsetz, H. (1973) "The Property Rights Paradigm." *Journal of Economic History* 33(1): 16–27.

Alexander, Christopher (1964) Notes on the Synthesis of Form, Cambridge, MA: Harvard University Press.

Argyres, N., & Mayer, K. J. (2007). Contract design as a firm capability: An integration of learning and transaction cost perspectives. *Academy of Management Review*, *32*(4), 1060-1077.

Arthur, W. Brian (2009) *The Nature of Technology: What It Is and How It Evolves*, New York: Free Press.

Atalay, E., Hortaçsu, A., & Syverson, C. (2014). Vertical integration and input flows. *The American Economic Review*, *104*(4), 1120-1148.

Augier, M., & Simon, H. A., (2003) The architecture of complexity: Background and central idea, *Managing in the Modular Age: Architectures, Networks, and Organizations*, (ed. Garud, R., Kumaraswamy, A., & Langlois, R. N.). Blackwell Publishers, 38-44.

Baker, G., Gibbons, R., & Murphy, K. J. (1999). Informal authority in organizations. *Journal of Law, Economics, and Organization*, *15*(1), 56-73.

Baldwin, Carliss Y. (2008) "Where Do Transactions Come From? Modularity, Transactions and the Boundaries of Firms," *Industrial and Corporate Change* 17(1):155-195.

Baldwin, Carliss Y. and Kim B. Clark (1997a) "Sun Wars: Competition within a Modular Cluster," in *Competing in the Age of Digital Convergence*, D. B. Yoffie, ed. Boston: Harvard Business School Press.

Baldwin, Carliss Y. and Kim B. Clark (1997b) "Managing in the Age of Modularity," *Harvard Business Review* Sept/Oct: 81-93.

Baldwin, Carliss Y. and Kim B. Clark (2000). *Design Rules, Volume 1, The Power of Modularity*, Cambridge, MA: MIT Press.

Baldwin, Carliss Y. and Kim B. Clark (2006) "Architectural Innovation and Dynamic Competition: The Smaller 'Span of control' Strategy," Harvard Business School Working Paper No. 07-014, available at http://www.people.hbs.edu/cbaldwin/ .

Baldwin, C. Y., & Henkel, J. (2014). Modularity and Intellectual Property Protection. *Strategic Management Journal* (forthcoming).

Baldwin, C. Y., MacCormack, A. D., & Rusnak, J. (2013). Hidden Structure: Using Network Methods to Map Product Architecture. *Research Policy,* 43, 1381-1397.

Baldwin, Carliss Y., and C. Jason Woodard (2011) "The Architecture of Platforms: A Unified View." In *Platforms, Markets and Innovation*, Annabelle Gawer, ed. (London: Edward Elgar)

Barney, Jay (1991) "Firm Resources and Sustained Competitive Advantage," *Journal of Management*, 17(1):99-120.

Blaauw, G. A. & Brooks, F. P. Jr. (1997). Computer architecture. Addison-Wesley, Reading MA.

Blair, M. M., O'Connor, E. O. & Kirchhoefer, G. (2011). Outsourcing, Modularity, and the Theory of the Firm. *Brigham Young University Law Review* 2011(2), 263-314. Available at: http://digitalcommons.law.byu.edu/lawreview/vol2011/iss2/1

Boudreau, K. (2010). Open platform strategies and innovation: Granting access vs. devolving control. *Management Science*, *56*(10), 1849-1872.

Browning, Tyson R. (2001) "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions in Engineering Management* 48(3):292-306.

Brynjolfsson, E., & McAfee, A. (2014). *The second machine age: work, progress, and prosperity in a time of brilliant technologies*. WW Norton & Company.

Chandler, A. D. Jr. (1986). The beginnings of the modern industrial corporation. *Proceedings of the American Philosophical Society,* 130(4), 382-389.

Chandler, Alfred D. (1962) *Strategy and Structure*, Cambridge, MA: MIT Press.

Chandler, Alfred D. (1977) *The Visible Hand: The Managerial Revolution in American Business*, Cambridge, MA: Harvard University Press.

Church, J., & Gandal, N. (1992). Network effects, software provision, and standardization. *The Journal of Industrial Economics*, 85-103.

Church, J., & Gandal, N. (1993). Complementary network externalities and technological adoption. *International Journal of Industrial Organization*, *11*(2), 239-260.

Church, J., Gandal, N., & Krause, D. (2008). Indirect network effects and adoption externalities. *Review of Network Economics*, *7*(3).

Colfer, Lyra J. and Carliss Y. Baldwin (2010) "The Mirroring Hypothesis: Theory, Evidence and Exceptions," Harvard Business School Working Paper No. 10-058, January 2010 (revised, June 2010), available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1539592 .

Demsetz, Harold (1967) "Toward a Theory of Property Rights," *American Economic Review,* 57:253-257.

Demsetz, Harold (1988) "The Theory of the Firm Revisited," *Journal of Law, Economics and Organization*, 4(1):141-161.

Donaldson, Gordon (1984) *Managing Corporate Wealth*, New York, NY: Praeger.

Eisenmann, Thomas, Geoffrey Parker and Marshall Van Alstyne (2006) "Strategies for Two-Sided Markets," *Harvard Business Review* 84(10): 92-101.

Eisenmann, Thomas, Geoffrey Parker and Marshall Van Alstyne (2011) "Opening Platforms: How, When and Why?" In *Platforms, Markets and Innovation*, Annabelle Gawer, ed., London: Edward Elgar.

Eppinger, S. D., Whitney, D. E., Smith, R. P., & Gebala, D. A. (1994). A model-based method for organizing tasks in product development. *Research in Engineering Design*, *6*(1), 1-13.

Eppinger, Steven D. (1991) "Model-based Approaches to Managing Concurrent Engineering" *Journal of Engineering Design*, 2: 283-290.

Ethiraj, Sendil and Daniel Levinthal (2004) "Modularity and Innovation in Complex Systems," *Management Science*, 50(2):159-174.

Ethiraj, Sendil K. (2007) "Allocation of Inventive Effort in Complex Product Systems," *Strategic Management Journal*, 28(6):563-584.

Ethiraj, Sendil K., Daniel Levinthal and Rishi Roy (2008) "The Dual Role of Modularity: Innovation and Imitation," Management Science 54(5): 939-955.

Evans, D. S., Hagiu, A., & Schmalensee, R. (2006). *Invisible engines: how software platforms drive innovation and transform industries*. Cambridge: MIT Press.

Farrell, Joseph and Garth Saloner (1986) "Installed Base and Compatibility: Innovation, Product Preannouncement and Predation," *American Economic Review*, 76(4): 940-955.

Farrell, Joseph and Carl Shapiro (1988) "Dynamic Competition with Switching Costs," *Rand Journal of Economics*, 19(1): 123-137.

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.

Fine, Charles H. (1998). *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*, Reading MA: Perseus Press.

Fine, Charles H. and Daniel E. Whitney (1996). "Is the Make-Buy Decision a Core Competence?" MIT Center for Technology, Policy, and Industrial Development Discussion Paper, available at http://web.mit.edu/ctpid/www/Whitney/morepapers/make_ab.html , viewed 8/31/07.

Fixson, Sebastian K. (2005) "Product Architecture and Assessment: A Tool to Link Product, Process, and Supply Chain Design Decisions," *Journal of Operations Management*, 23(2005):345-369.

Fixson, Sebastian K. and Jin-Kyu Park (2008). "The Power of Integrality: Linkages between Product Architecture, Innovation and Industry Structure," *Research Policy* 37(8):1296-1316.

Freeland, R. F., & Zuckerman, E., (2014). "The problems and promises of hierarchy: A sociological theory of the firm." Mimeo.

Gawer, A. (2014). Bridging differing perspectives on technological platforms: Toward an integrative framework. *Research Policy* (forthcoming).

Gawer, Annabelle and Michael A. Cusumano (2002) *Platform Leadership: How Intel, Microsoft and Cisco Drive Industry Innovation*, Harvard Business School Press, Boston, MA.

Gibbons, R. (2003). Team theory, garbage cans and real organizations: some history and prospects of economic research on decision-making in organizations. *Industrial and Corporate Change*, *12*(4), 753-787.

Gibbons, R., & Henderson, R. (2012). Relational contracts and organizational capabilities. *Organization Science*, *23*(5), 1350-1364.

Goldratt, E. M. (1990). *Theory of constraints*. Croton-on-Hudson, NY: North River.

Grossman, Sanford J. and Oliver D. Hart (1986) "The Costs and Benefits of Ownership: A Theory of Vertical and Lateral Integration," *Journal of Political Economy,* 94(4):691-719.

Gulati, R., & Puranam, P. (2009). Renewal through reorganization: The value of inconsistencies between formal and informal organization. *Organization Science*, *20*(2), 422-440.

Hagiu, A., & Yoffie, D. B. (2013). The new patent intermediaries: platforms, defensive aggregators, and super-aggregators. *The Journal of Economic Perspectives*, *27*(1), 45-65.

Hart, O. (1995). *Firms, contracts, and financial structure*. Oxford University Press.

Hart, Oliver and John Moore (1990) "Property Rights and the Nature of the Firm," *Journal of Political Economy,* 98(6):1119-1158.

Hatchuel, A., & Weil, B. (2009). CK design theory: an advanced formulation. *Research in engineering design*, *19*(4), 181-192.

Henderson, Rebecca M. and Kim B. Clark (1990), "Generational Innovation: The Reconfiguration of Existing Systems and the Failure of Established Firms," *Administrative Science Quarterly* 35: 9-30.

Henkel, J., Baldwin, C., & Shih, W. C. (2013). IP modularity: profiting from innovation by aligning product architecture with intellectual property. *California management review*, *55*(4), 65-82.

Henkel, J., Hoffmann, A. (2014). Value capture in hierarchically organized industries: The hierarchy strategy. Mimeo.

Hennessy, John L. and David A. Patterson (1990) *Computer Architecture: A Quantitative Approach,* San Mateo, CA: Morgan Kaufmann.

Higgins, R. C. (1977). How much growth can a firm afford?. *Financial Management*, 7-16.

Hounshell, David A. (1998) "Measuring the Return on Investment in R&D: Voices from the Past, Visions of the Future," in Assessing the Value of Research in the Chemical Sciences: Report of a Workshop, National Academy Press, Washington D.C., available at http://www.nap.edu/catalog/6200.html; pp. 6-14.

Hughes, Thomas P. (1987) "The Evolution of Large Technological Systems," in *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology,* (W.E. Bijker, T. P. Hughes, T. Pinch, eds.) Cambridge, MA: MIT Press.

Iansiti, Marco and Kim B. Clark (1993) "Integration and Dynamic Capability: Evidence from Product Development in Automobiles and Mainframe Computers," *Industrial and Corporate Change*, 3(3): 557-605.

Iansiti, Marco and Roy Levien (2004). *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, Boston: Harvard Business School Press.

Jacobides, Michael G. (2005). "Industry Change Through Vertical Dis-Integration: How and Why Markets Emerged in Mortgage Banking," *Academy of Management Journal*, 48(3):465-498.

Jacobides, M. G., & Billinger, S. (2006). Designing the boundaries of the firm: From "make, buy, or ally" to the dynamic benefits of vertical architecture. *Organization science*, *17*(2), 249-261.

Jacobides, Michael G., Thorbjorn Knudsen and Mie Augier (2006) "Benefiting from Innovation: Value Creation, Value Appropriation and the Role of Industry Architecture," *Research Policy,* 35(8):1200-1221.

Jacobides, M., MacDuffie, J. P., & Tae, C. J. (2012, May). When value sticks around: Why automobile OEMs still rule their sector. In *Industry Studies Association Conference* (pp. 29-31).

Jacobides, M. G., & Tae, C. J. (2014). Kingpins, bottlenecks and value dynamics within a sector. *Organization Science.* (Forthcoming).

Jacobides, M. G., Veloso, F., & Wolter, C. (2014). Ripples through the value chain and positional bottlenecks: Innovation, profit, and scope evolution in a competitive setting. Mimeo.

Kapoor, R. (2013). Persistence of integration in the face of specialization: How firms navigated the winds of disintegration and shaped the architecture of the semiconductor industry. *Organization Science*. 24(4), 1195-1213.

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.

Katz, M. L., & Shapiro, C. (1994). Systems competition and network effects. *The Journal of Economic Perspectives*, 93-115.

Kauffman, Stuart A. (1993) *The Origins of Order*, New York: Oxford University Press

Kim, J., & Mahoney, J. T. (2005). Property rights theory, transaction costs theory, and agency theory: an organizational economics approach to strategic management. *Managerial and Decision Economics*, *26*(4), 223-242.

Kogut, B., MacDuffie, J. P., & Ragin, C. (2004). Prototypes and strategy: Assigning causal credit using fuzzy sets. *European Management Review*, *1*(2), 114-131.

Langlois, Richard N. (2002). "Modularity in Technology and Organization," *Journal of Economic Behavior and Organization,* 49(1):19-37.

Langlois, R. N. (2007). Organizing the electronic century. University of Connecticut Working Paper. March.

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(4): 297-313.

Lemley, M. A., & Shapiro, C. (2005). Probabilistic patents. *Journal of Economic Perspectives*, 75-98.

Lemley, M. A., & Shapiro, C. (2006). Patent holdup and royalty stacking. *Texas Law Review*, *85*, 1991.

Levinson, M. (2006). *The box: how the shipping container made the world smaller and the world economy bigger.* Princeton University Press.

Levinthal, Daniel A. (1997) "Adaptation on Rugged Landscapes" *Management Science*, 43: 934-950.

Libecap, G. D. (1989). Distributional issues in contracting for property rights. *Journal of Institutional and Theoretical Economics (JITE)/Zeitschrift für die gesamte Staatswissenschaft*, 6-24.

Libecap, G. D. (1993). *Contracting for property rights*. Cambridge university press.

Liebeskind JP. (1997) "Keeping organizational secrets: Protective institutional mechanisms and their costs," Industrial and Corporate Change **6**: 623-663.

Luo, J., Baldwin, C. Y., Whitney, D. E., & Magee, C. L. (2012). The architecture of transaction networks: a comparative analysis of hierarchy in two sectors. *Industrial and Corporate Change*, 21(6), 1307-1335.

Mahoney, J. T. (2004). *Economic foundations of strategy*. Sage Publications.

Mayer, Kyle J. and Nicholas S. Argyres (2004) "Learning to Contract: Evidence from the Personal Computer Industry," *Organization Science*, 15(4):394-410.

Milgrom, Paul and John Roberts (1990) "The Economics of Manufacturing: Technology, Strategy and Organization," *American Economic Review* 80 (3): 511-28.

Monteverde, Kirk (1995) "Technical Dialog As an Incentive for Vertical Integration in the Semiconductor Industry," *Management Science*, 41(10): 1624-1638.

Monteverde, K., & Teece, D. J. (1982). Supplier switching costs and vertical integration in the automobile industry. *The Bell Journal of Economics*, 206-213.

Mueller, Scott (2003), *Upgrading and Repairing PCs*, 15th ed., Indianapolis: Que Publishing.

Murmann, Johann Peter and Koen Frenken (2006) "Toward a Systematic Framework for Research on Dominant Designs, Technological Innovations, and Industrial Change," *Research Policy* 35:925-952.

Nadler, D., & Tushman, M. (1997). *Competing by design: The power of organizational architecture*. Oxford University Press.

Nickerson, Jack A. and Todd R. Zenger (2004) "A Knowledge-based Theory of the Firm—The Problem-Solving Perspective," *Organization Science*, 15(6): 617–632

Orton, J. D., & Weick, K. E. (1990). Loosely coupled systems: A reconceptualization. *Academy of management review*, *15*(2), 203-223.

Pahl, G. and W. Beitz (1995) *Engineering Design: A Systematic Approach*, (2nd English edition: Wallace, K. et.al., trans.) London: Springer-Verlag.

Parnas, David L. (1972) "On the Criteria to Be Used in Decomposing Systems into Modules," *Communications of the ACM* 15: 1053-58; reprinted in Hoffman and Weiss (eds.) *Software Fundamentals: Collected Papers of David Parnas*, Boston MA: Addison-Wesley.

Parnas, D.L., P.C. Clements, and D.M. Weiss (1985) "The Modular Structure of Complex Systems," *IEEE Transactions on Software Engineering*, SE-11: 259-66; reprinted in Hoffman and Weiss (eds.) *Software Fundamentals: Collected Papers of David Parnas*, Boston MA: Addison-Wesley.

Patterson, David A. and John L. Hennessy (1994) *Computer Organization and Design: The Hardware/Software Interface*, San Mateo, CA: Morgan Kaufmann Publishers.

Perrow, Charles (1984) *Normal Accidents: Living with High-Risk Technologies*, Princeton, NJ: Princeton University Press.

Pil, Frits K. and Susan K. Cohen (2006) "Modularity: Implications for Imitation, Innovation, and Sustained Competitive Advantage," Academy of Management Review, 31(4):995-1011.

Pisano, Gary P. and David J. Teece (2007) "How to Capture Value from Innovation: Shaping Intellectual Property and Industry Architecture," *California Management Review*, 50(1):278-296

Rivkin, Jan W. (2000) "Imitation of Complex Strategies" *Management Science* 46:824-844.

Rochet, Jean-Charles and Jean Tirole (2003) "Platform Competition in Two-sided Markets," *Journal of the European Economic Association*, 1(4): 990-1029.

Rosenberg, N. (1969). The direction of technological change: inducement mechanisms and focusing devices. *Economic Development and Cultural Change*, 1-24.

Rosenberg, N. (1982). *Inside the black box: Technology and economics*. Cambridge University Press.

Sako, Mari and Fiona Murray (1999) "Modules in Design, Production and Use: Implications for the Global Automotive Industry," Paper presented at the Fall Meeting of the International Motor Vehicle Program (IVMP) Annual Sponsors Meeting, 5-7 October 1999, Cambridge MA.

Sanchez, Ronald A. and Joseph T. Mahoney (1996). "Modularity, Flexibility and Knowledge Management in Product and Organizational Design," *Strategic Management Journal*, 17: 63-76.

Schotter, A., & Teagarden, M. (2014). Protecting Intellectual Property in China. *Sloan Management Review*, Summer 2014, 41-50.

Scott, W. R. (1987). *Organizations*. Englewood Cliffs, NJ: Prentice hall.

Sherwood, R. M. (1990). *Intellectual property and economic development.* Boulder: Westview Press.

Simon, Herbert A. (1981). *The Sciences of the Artificial, 2nd Ed.* Cambridge, MA: MIT Press.

Steward, Donald V. (1981) "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management* EM-28(3): 71-74 (August).

Suh, Nam (1990) *The Principles of Design* Oxford UK: Oxford University Press.

Teece, David J. (1986)."Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy," *Research Policy*, 15(6): 285-305.

Teece, David J. (1996). "Firm Organization, Industrial Structure, and Technological Innovation," *Journal of Economic Behavior and Organization*, 31(2): 193-224.

Teece, D. (2014). The Foundations of Enterprise Performance: Dynamic and Ordinary Capabilities in an (Economic) Theory of Firms. *The Academy of Management Perspectives*, amp-2013.

Teece, David J., Gary Pisano and Amy Shuen (1997) "Dynamic Capabilities and Strategic Management," *Strategic Management Journal*, 18(7): 509-533.

Thompson, James D. (1967) *Organizations in Action: Social Science Bases of Administrative Theory*, New York, NY: McGraw-Hill.

Topkis, D. M. (1998). *Supermodularity and complementarity*. Princeton University Press.

Tuertscher, P., Garud, R., & Kumaraswamy, A. (2014). Justification and Interlaced Knowledge at ATLAS, CERN. *Organization Science* (forthcoming).

Tushman, Michael L. and Lori Rosenkopf (1992) "On the Organizational Determinants of Technological Change: Toward a Sociology of Technological Evolution," *Research in Organizational Behavior*, 114:311-347.

Tushman, Michael L. and Murmann, J. Peter (1998) "Dominant designs, technological cycles and organizational outcomes" in Staw, B. and Cummings, L.L. (eds.) *Research in Organizational Behavior*, JAI Press, Vol. 20.

Ulrich, Karl (1995) "The Role of Product Architecture in the Manufacturing Firm," *Research Policy,* 24:419-440.

Weick, Karl E. (1969) *The Social Psychology of Organizing* (2nd edition) New York, NY: McGraw-Hill.

Weinberger, D. (2008). *Small pieces loosely joined: A unified theory of the Web.* Basic Books.

Wernerfelt, Birger (1984) "A Resource-Based View of the Firm," *Strategic Management Journal*, 5(2):171-180.

West, Joel (2003). "How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies," *Research Policy*, 32(7): 1259-1285.

Whitney, Daniel E. (2004) "Physical Limits to Modularity," http://esd.mit.edu/symposium/pdfs/papers/whitney.pdf, viewed July 21, 2005

Whitney, Daniel E., Edward Crawley, Olivier de Weck, Steven Eppinger, Christopher Magee, Joel Moses, Warren Seering, Joel Schindall, David Wallace (2004) "The

Influence of Architecture in Engineering Systems," Engineering Systems Monograph, MIT, Cambridge, MA (March); available at http://esd.mit.edu/symposium/monograph/ (viewed 7/29/07).

Yoffie, D. B. (Ed.). (1997). *Competing in the age of digital convergence*. Harvard Business Press

Zhou, Y. M., & Wan, X., (2012). "The Bottleneck: Product Variety and Coordination Failures at A Major Soft Drink Bottling Company." Mimeo.

Zhu, F., & Iansiti, M. (2012). Entry into platform-based markets. *Strategic Management Journal*, *33*(1), 88-106.