



**Modularity for Value
Appropriation:
Drawing the Boundaries of
Intellectual Property**

**Joachim Henkel
Carliss Baldwin**

Working Paper

09-097

First revision

Copyright © 2009 by Joachim Henkel, Carliss Baldwin

Working papers are in draft form. This working paper is distributed for purposes of comment and discussion only. It may not be reproduced without permission of the copyright holder. Copies of working papers are available from the author.

09-097

Modularity for Value Appropriation: Drawing the Boundaries of Intellectual Property

Joachim Henkel*
Carliss Y. Baldwin†

March 2009

First revision

* Technical University Munich
henkel@wi.tum.de

† Harvard Business School
cbaldwin@hbs.edu

Copyright © 2009, Joachim Henkel and Carliss Y. Baldwin

Working papers are in draft form. This working paper is distributed for purposes of comment and discussion only. It may not be reproduced without permission of the copyright holder. Copies of working papers are available from the author.

Modularity for Value Appropriation – Drawing the Boundaries of Intellectual Property

March 25, 2009

Joachim Henkel¹, Carliss Y. Baldwin²

Abstract

The existing theory of modularity explains how modular designs create value. We extend this theory to address value appropriation. A product or process design that is modular with respect to intellectual property (IP) allows firms to better capture value in situations where knowledge and value creation are distributed across many actors. We use case studies to develop an inductive theory of “IP modularity,” from which we derive testable propositions and managerial implications.

Keywords: Modularity, value appropriation, intellectual property, open innovation, design

¹Munich University of Technology, Arcisstr. 21, 80333 Munich, Germany, henkel@wi.tum.de.

²Harvard Business School, Soldier’s Field, Boston, MA 02163, USA, cbaldwin@hbs.edu.

We are grateful to many people who shared their insights or commented on earlier drafts of the paper. Special thanks go to Sung Joo Bae, Marc Gruber, Kathrin Henkel, Mako Hill, Eric von Hippel, Michael Jacobides, Andrew King, Karim Lakhani, Deishin Lee, Alan MacCormack, Matt Marx, Phanish Puranam, Manuel Sojer, Victoria Stodden, Mary Tripsas, David Upton, and three anonymous reviewers for the Academy of Management Meeting. Thanks also to participants in seminars and workshops at Bocconi University, Harvard Business School, Imperial College Business School, London Business School, MIT, and Ross School of Business. Errors and oversights are ours alone.

INTRODUCTION

Going back to Simon's (1962) notion of "nearly decomposable systems," the concept of modularity has received broad attention in the field of management (e.g., Baldwin & Clark, 1997, 2000; Langlois, 2002; Sanchez & Mahoney, 1996; Schilling, 2000). Important research streams have studied how the modular structures of products and of organizations interact, both within firms (Brusoni & Prencipe, 2006; Hoetker, 2006; Sanchez & Mahoney, 1996) and industry-wide (Langlois, 2003; Schilling & Steensma, 2001; Sturgeon, 2002; Takeishi, 2002). Various benefits of modularity have been analyzed, including the division of labor in both innovation and production processes (Chesbrough & Kusunoki, 2001; Staudenmayer et al., 2005; von Hippel, 1990; Sako, 2003); the ability to recombine modules in new ways (Garud & Kumaraswamy, 1995; Sanchez, 1995); and technical advantages in terms of improved performance, variety, and evolvability (e.g., Baldwin & Clark, 2000; Ulrich, 1995).

Broad as it is, we argue that the existing analysis of modularity from a management perspective is incomplete, and in a fundamental way. With few exceptions, the prior literature on modularity has focused on *value creation*. In contrast, the impact of modularity on *value appropriation* has largely been ignored. We address this gap.

The link between modularity and value appropriation is especially tight when intellectual property (IP) is at stake. As distinct from many tangible products, which are sold once and for all with the purchaser obtaining all rights of use, intellectual property is often not transferred in an unconditional way. For example, a buyer of a machine is typically free to use the machine itself in any way possible, but not the knowledge embodied in it: the seller may restrict her rights to use this knowledge by patents and licensing provisions, and may deny access to this knowledge by secrecy.

More generally, a firm may keep its intellectual property inhouse, or may license it. IP licenses in turn are complex and come in many varieties.

This variegated aspect of IP creates both pitfalls and opportunities, which “IP-oriented” modularity can address. Consider the following example. In 1998, Valve Software released the game “Half-Life,” having designed the code base in such a way that the core engine and the complementing code were separate modules (Jeppesen, 2004). Importantly in our context, *these modules were licensed under different terms*, in other words, were given different “IP status.” Valve put the engine under a proprietary license and kept its source code secret, while it disclosed the complementing source code and granted users a broad license to modify and share it. Valve thus waived many of the IP rights it owned to the complementing code, and by doing so changed this code’s IP status. Within eight months of release, users had used the complementary code to build a modified game, “Counter-Strike,” which became far more popular than the original game. Still, in order to run the free modification, Counter-Strike, players had to license and use the Half-Life core engine.

Through its IP-oriented modularization, Valve achieved a remarkable commercial success. In the counterfactual case of a non-modular code base, the firm would have had to disclose no code at all or the entire code base. In the first case, value co-creation by users would have been impossible; in the second, user involvement would have been easy, but *capturing* value would have been difficult. Thus an IP-oriented modularization allowed Valve to *reconcile an intrinsic conflict between value co-creation and value capture*.

The literature on the appropriation of innovation rents provides another way to look at the relationship between modularity and intellectual property. Mechanisms that enable value appropriation, often referred to as “isolating” or “appropriability” mechanisms, include patents, copyright, and secrecy (Cohen et al., 2000; Levin et al., 1987; Rumelt, 1984; Teece, 1986;). The

diversity of such mechanisms suggests that for complex products and processes, the optimal mechanism—hence the optimal IP status—may be different for different parts of the system. For example, recent studies have shown that in some instances (as in the Counter-Strike example above) innovation and value creation are best served by providing open access to some parts of the system (e.g., Foray, 2004: 165; Harhoff et al., 2003; Pénin, 2007; von Hippel & von Krogh, 2003). But when innovations are undertaken by firms that must recoup their investments, openness will typically be practiced selectively (Henkel, 2006). Thus, a linkage between modularity and value appropriation emerges naturally from the need to choose different IP treatments for different parts of a complex system.

In this paper we investigate the concept of modularity with respect to intellectual property, or “IP modularity” for short. For a complex system, modularity is the property of being divided up into discrete components—called modules—that have tight linkages within, but loose linkages to other parts of the system (Baldwin & Clark, 2000; Schilling, 2000). The modules in turn must work together in order for the system as a whole to function and be valued.

The *knowledge* needed to make a complex product or implement a complex process forms an *overlay* on the modular structure of the product or process. That knowledge and the corresponding IP, must in turn be managed in accordance with the firm’s strategy for capturing value. Specifically, in complex systems, it is often desirable to treat different “chunks” of knowledge differently from an IP perspective, adopting what we call a “mixed-IP strategy.” A mixed-IP strategy involves creating “IP modules” each of which contains only “compatible” intellectual property, and designing the modules of the underlying product or process to correspond with the IP modules. (Below we define what it means for IP to be “compatible”: the basic idea is that there should be no conflicts of intellectual property rights within a module.)

In what follows, we first lay out a theoretical framework and then use case vignettes to develop a set of inductive propositions regarding IP modularity and mixed-IP strategies. These propositions in turn can serve as the basis for testable hypotheses. Most fundamentally, we propose that IP modularity is more advantageous when the potential for value co-creation in the surrounding ecosystem is high and when the co-creators of value are widely distributed and hard to identify. Additional propositions address the effects of complexity, customization, outsourcing, hold-up, and uncertainty on IP modularity.

IP modularity matters at all levels of a firm. It matters at the strategic level, because it sets the basic terms of collaboration between the focal firm and its suppliers, customers and complementors. As the example of Counter Strike shows, IP modularity may even form the basis of a firm's business model. At the level of product design, product architects must take considerations of IP modularity into account when designing their artifacts. And at the operational level, organizations may have to adapt their day-to-day routines to accommodate selectively open innovation processes enabled by IP modularity and mandated by a mixed-IP strategy (Alexy & Henkel, 2007).

Also, we believe that IP modularity is increasingly important to managers, for two reasons. First, the management of formal IP in the form of patents and copyrights is an increasingly important aspect of the management of innovation. But at the same time, there is movement in the opposite direction: open and semi-open innovation processes are steadily gaining ground in a number of industries (e.g., Chesbrough, 2003; Henkel, 2006; West, 2003). IP modularity helps to reconcile both trends.

The rest of the paper is organized as follows. In the following section, we review the literatures on modularity and value appropriation, and explain how this paper relates to prior work. Next, we lay the theoretical foundation of our analysis, formally defining the concepts of "IP

status,” “IP incompatibility,” and “IP modularity.” We also categorize the different types of IP—“outgoing” and “incoming” IP—which provide opportunities or threats that IP modularity can address. We then present ten propositions with supporting case vignettes; these can serve as the basis for testable hypotheses. The final section provides a discussion and conclusions.

LITERATURE REVIEW

In this section we review three bodies of literature which are fundamentally related to IP modularity. We first review the management literature on modularity and value creation. Next we turn to the strategy literature on value appropriation, looking at both the resource-based view (RBV) of the firm (Barney, 1991; Rumelt, 1984; Wernerfelt, 1984), and the literature on “profiting from innovation” (Chesbrough & Teece, 1996; Teece, 1986, 2000). By and large, in the strategy literature, the structures of products and processes are assumed to be fixed, thus modularization and a related partitioning of intellectual property rights are not envisioned as strategic possibilities.

Modularity and Value Creation

Simon (1962: 477) was the first to note that modularity reduces the cost of managing the cognitive burden of complexity. In addition, modularity allows tasks to be partitioned (Gomes & Joglekar, 2008; von Hippel, 1990) and worked on in parallel (Baldwin & Clark, 2000). With task partitioning, a productive division of labor may arise between individuals within one firm (Sanderson & Uzumeri, 1995) or across firm boundaries (Baldwin & Clark, 2000; Langlois & Robertson, 1992; Sanchez, 1995; Staudenmayer et al., 2005). Sanchez and Mahoney (1996) more broadly made the point that modularity in the design of products and processes permits tasks within a complex production process to be divided up among several firms. The existence of task modules

with well-defined interfaces reduces transaction costs, thus supports outsourcing (Baldwin, 2008; Fine, 1998; Hoetker, 2006; Jacobides, 2005; Sturgeon, 2002).

Modularity also makes it possible to achieve high product variety at relatively low cost (Baldwin & Woodard, 2009; Matutes & Regibeau, 1988; Sanderson & Uzumeri, 1995; Wheelwright & Clark, 1992). Also, users may be able to upgrade their products selectively by replacing components with better ones as they become available (Garud & Kumaraswamy, 1995; Baldwin & Clark, 2000; Schilling, 2000). Generally, modular systems are celebrated for their flexibility, adaptability, and evolvability (e.g., Baldwin & Clark, 1997, 2000; Baldwin & Woodard, 2009; Garud & Kumaraswamy, 1995; Langlois & Robertson, 1992). Changes within modules and recombinations of modules are easy given a modular architecture. (In contrast, a change of the architecture itself is difficult, indeed virtually impossible when many actors rely on it.)

In a modular system, innovators do not have to be lodged within the firm that created the modular architecture. Innovations can be introduced by firms in related markets, entrepreneurial startups, or even users (Baldwin & Clark, 2000; Franke & von Hippel, 2003; Jeppesen, 2004; von Hippel, 2001; von Hippel & Finkelstein, 1979). However, the fact that modularity enables innovation “at a distance” from the focal firm creates implicit tension between value creation and value appropriation. The architect of a modular system risks losing the lion’s share of value to complementary innovators. For example, the modular architecture of IBM’s mainframe computer System/360 allowed the makers of plug-compatible peripherals, such as disk drives, to enter the market (Baldwin & Clark, 2000; Baldwin, 2008). The IBM PC had a highly modular architecture, but in the end most of the value was captured by Intel and Microsoft, not IBM (Ferguson & Morris, 1993). Yet MacCormack & Iansiti (2009) also point out that creating a modular architecture, with clear separations between interfaces and code, can facilitate value appropriation by allowing the firm to make interfaces public while hiding the internal code.

The above is a sampling of a very large management literature on modularity. Almost without exception, this literature focuses on how modular architectures *create value*. When value *appropriation* is considered, it is in the context of stories, such as System/360 or the IBM PC. Thus far, there has been no systematic analysis of how firms can use modularity to appropriate value.

Value Appropriation

Two strands in the modern strategy literature address value appropriation directly. First is the resource-based view of the firm (RBV), whose modern formulations are due to Rumelt (1984), Wernerfelt (1984), Barney (1991) and others. Second is the literature on how firms profit from innovation, which originated with Teece (1986).

The resource-based view holds that competitive advantage derives from a firm's ownership or control of "assets, capabilities, organizational processes, information, or knowledge" (Daft, 1983, quoted by Barney, 1991) that are valuable, rare, inimitable and non-substitutable (Barney, 1991). The ensuing rents must be protected by "isolating mechanisms" (Rumelt, 1984) or "resource position barriers" (Wernerfelt, 1984). When the resource is knowledge, state-sanctioned property rights (patents and copyright) and/or secrecy serve as isolating mechanisms or position barriers that enhance the focal firm's bargaining power, and allow it to appropriate the value of its knowledge resources (Lavie, 2007; Reitzig and Puranam, 2008).

The traditional RBV does not envision a world in which firms are "interconnected" and might gain by sharing or giving away knowledge to others (Lavie, 2006). Yet the growing importance of alliance networks (Gulati, 1998, Lavie, 2006) and "ecosystems" (Iansiti and Levien, 2004) indicates that knowledge sharing can be advantageous both for individual firms and groups of firms. But at the same time, a firm with no protected resources can only earn competitive returns, not superior returns.

Such cases call for an intermediate strategy in which some knowledge is shared while other knowledge remains protected. Modularity theory explains how to partition complex products and processes (Baldwin and Clark, 2000), making it practicable to partition knowledge into discrete “chunks,” which in turn receive different intellectual property treatments. On this view, module boundaries serve as “partitioning mechanisms,” dividing the firm’s knowledge into discrete subsets. Some knowledge has relational value (Dyer and Singh, 1998; Lavie, 2006) hence should be shared. Other knowledge has positional value (Wernerfelt, 1984) hence should be protected via intellectual property rights and/or secrecy.

A second strand in the strategy literature that deals with value appropriation asks the question how firms profit—or fail to profit—from innovation (Teece, 1986). Teece and others make a distinction between “autonomous” innovations, which can be introduced without major modifications to other parts of the system, and “systemic” innovations, which require changes to many other parts (Chesbrough & Teece, 1996; Langlois & Garzarelli, 2008; Teece, 1988, 2000). In effect, autonomous innovations arise *within the modules* of a modular system, whereas systemic innovations arise in integral (i.e., non-modular) systems. According to Teece (1988, 2000), the value of autonomous innovations can be captured by small firms, while systemic innovations are best pursued by large companies with complementary assets and capabilities and financial resources.

In this literature, “autonomous” and “systemic” are taken to be essential, unchangeable properties of an innovation. In contrast, the literature on modularity offers many examples where it is possible to change the structure of the system to make it more modular (subject to autonomous innovation) or integral (subject to systemic innovation). However, modularizations may be problematic for value appropriation. If the focal firm modularizes a product or process, then, according to Teece, it will be possible for smaller firms to enter, innovate on modules, and compete

away rents. (This is essentially what happened with IBM's System/360.) Why would a firm want to open itself up to this type of competition?

Taken as a whole, the prior literature on modularity and value appropriation seems to imply that modularizations are problematic for value appropriation. A modular architecture by definition permits innovation at the module level of the system, which in turn encourages new players to enter the product market. In the context of the RBV, modularity can be seen as weakening the focal firm's "isolating mechanisms" or "resource position barriers." In the Teeceian context, modularity changes the nature of innovations from "systemic" to "autonomous" and thus allows smaller firms to compete for innovation rents.

We agree that modularization can be a dangerous strategic move if the implications of the new architecture are not thought through in advance. For this reason, modularity may intentionally be avoided. However, we believe that implications of the prior literature are too pessimistic. As the case of Counterstrike shows, when combined with "IP modularity," product or process modularity may allow the focal firm to appropriate more value than it could obtain with an integral (i.e., non-modular) architecture.

Of course a firm's control over its own architecture is never perfect: there are always constraints and tradeoffs to be considered. Indeed it is precisely the fact that there are tradeoffs that makes the concept of IP modularity, introduced in the next section, economically and managerially significant.

INTRODUCING IP MODULARITY

We begin this section by introducing the notion of "antagonistic coupling" or "incompatibility," which is central to the concept of IP modularity. We then formalize "IP modularity," and finally present a taxonomy of types of IP modularization.

Antagonistic Coupling: Incompatibility

Just as it has focused on value creation, the prior literature on modularity has focused almost exclusively on linkages—often called “couplings”—that encourage designers to place elements *closer together*. Typically, if design decisions with respect to element X depend on decisions with respect to Y or vice versa, it is recommended to place X and Y in the same module.

Conceptually, however, it is possible for X and Y to repel one another, in which case it is good practice to place the two elements in *separate* modules. This is what often happens with IP. Consider the following example. A software developer obtains code from two sources: from a commercial supplier, under a typical proprietary license; and from an open source repository, licensed under the General Public License (GPL). The proprietary license prohibits passing on the human-readable source code to any third party. The GPL stipulates that any user of a program derived from GPL-licensed code is entitled to obtain the source code, read it, modify it, and pass both source and binary code on to others (Free Software Foundation, 1991). Interweaving the commercially licensed code and the open source code then leads to conflicting legal requirements. While the commercial license *forbids* passing on the source code to third parties, the GPL *requires* exactly that. Thus, the two licensed-in blocks of code exhibit, from an IP perspective, a negative, antagonistic coupling, which we call “IP incompatibility.” At the very least, the two blocks must not be placed in the same module.

The above example involved IP obtained from others, which we call *incoming* IP. But there can also be incompatibility in the firm’s own *outgoing* IP. Recall the example of Counter-Strike described in the Introduction. The core engine needed strong IP protection (realized by copyright and secrecy) to minimize the risk of imitation. But the complementing code needed to be accessible to those who would be motivated to create new games. Many potential game creators did not work for Valve, nor were they interested in signing non-disclosure agreements that would allow them to

view closed source code. Thus, the two parts of the system exhibited incompatibility with respect to IP: the desired IP status for one was incompatible with the desired IP status for the other.

Accordingly, Valve placed the core engine and the complementing code in segregated modules, and established a different IP status for each part.

With these examples in mind, we can formulate the following general definition:

Antagonistic coupling, or incompatibility, arises between two elements when some type of interaction requires or strongly advises that they be treated in different ways. Placing them in separate modules will be necessary, and often also sufficient, in order to allow for such different treatment. Note that antagonistic coupling and incompatibility can also arise in contexts unrelated to IP, a point we will come back to at the end of the paper.

We are now in a position to introduce “IP modularity” formally.

Formalizing the Concept of IP Modularity

We begin this section with a historical vignette. In the eighteenth century, Saxony’s ruler Augustus the Strong obtained a monopoly on European porcelain by the simple expedient of imprisoning the inventor of the process in a fortress in Meissen (this was the first Meissen porcelain factory). But when the inventor was close to death, the ruler sought to protect his valuable intellectual property by resorting to another age-old strategy—*divide et impera*, or divide and rule. He instructed the inventor to tell his secrets to two men. One learned how to make porcelain paste; the other porcelain glaze. When the inventor died, no one person knew everything about how to make Meissen porcelain products. The rulers’s monopoly was safe, although eventually the secrets did leak out (Gleeson, 1998).

The above is a classic example of the construction of what we are calling an “IP-modular” system with a mixed-IP strategy, designed for the purpose of appropriating value (protecting the

ruler's monopoly). We will use this vignette to introduce the concept of IP modularity and to work through the elements of a mixed-IP strategy. Central points of our theoretical framework are the following. Complex systems consist of linked elements, each of which carries a certain "IP status." The IP status of two elements may be inconsistent. To accommodate such IP incompatibility, the respective elements should be placed in different modules. Doing so is possible since designers have choice in defining module boundaries within the system. The resulting "IP modularity" of the system is conducive to the focal firm's value appropriation.

We expand on this outline in the paragraphs below using the Meissen case to make our arguments more concrete (insofar as porcelain can be made concrete!).

The system as a network of linked elements. The concept of IP modularity applies to *complex products or processes*, which consist of several coherent parts, called *components*. In our Meissen example, the making of porcelain paste, the shaping and firing of porcelain objects, and the glazing of the objects are all components (or steps) of the porcelain process. Each component in turn will be made up of smaller components and *elements*—these would be the sub-processes and individual steps of the porcelain-making recipe. The components and their dependencies can be represented as a network, as illustrated in Figure 1. The linkages indicate dependencies of the form "a change in one component may require an adaptive change in some other component."

---- *Insert Figure 1 about here* ----

Complementarity between components. Essentially all components must be present for the system as a whole to have value. In particular, in a process like porcelain-making all steps must be carried out, hence, they are *strong complements* to each other, just as Valve's core engine and the user-generated code are in the case of Counter-Strike. In general, as we shall see, strategies based

on IP modularity rely on strong complementarity between the components of the system. The product or process as a whole is much more valuable than the sum of its parts.

Choice in defining module boundaries. For purposes of carrying out the work, the *different components may be separated and segregated*d, and there are *different ways* of doing so. In porcelain making, the unglazed objects could be made in one location by one set of workers, and the glazing carried out in a different location by a different set of workers. The segregated components are “modules.” Note that the process of porcelain making, in common with many other products and processes, did not *have* to be organized in a modular fashion, but it *could* be carried out that way. The degree of modularity in the process and, especially, the location of module boundaries were to a good extent under the control of the process designers. For illustration, Figure 1(a) exhibits an integral system turned into a modular system (b), during which process one dependency is removed and another added.

Knowledge and IP corresponding to each module. In porcelain making, not only were the steps separable, but the knowledge relating to the early and late stages of the process was separable as well. Knowledge of how to mix, shape and firm porcelain paste did not give one knowledge of how to mix and apply glazes, and vice versa. Therefore, knowledge could be split up in ways that corresponded to the underlying modular structure of the product or process. When, furthermore, knowledge is covered by IP—because the owner has legal or de facto means of excluding others from using it, as Saxony’s ruler had—then the relevant IP can also be split up according to the underlying modular structure.

A module’s IP status. The IP covering a particular module (or any element of the system, for that matter) defines what we call the module’s “IP status.” By this term we mean the legal rights to and de facto accessibility of knowledge about the module and its role in the surrounding system. In our porcelain example, the knowledge related to the “porcelain paste” module was awarded the

IP status “secret to the world, known only to person A,” while the “porcelain glaze” module was given the distinct status “secret to the world, known only to person B.”

Generally, some elements, and often entire components, will be based on knowledge that the focal firm owns outright. For these elements and components, the firm must decide which legal rights and what types of access to knowledge to grant others. The firm may decide to keep its knowledge proprietary, using patents, copyrights, or secrecy. It may choose to license its proprietary IP to third parties or keep it inhouse. In other cases, the firm may make its IP available under a range of “open source” licenses, or may release it into the public domain.

In addition to using knowledge that it owns, the firm may also obtain relevant knowledge from others. In such cases, an element’s or module’s IP status is externally given. Such IP may be acquired under a commercial license, obtained under an open source licence, or taken from the public domain.

IP incompatibility. As explained above, the IP rights or de facto ability to use different chunks of knowledge may be inconsistent. Such inconsistency may arise, for incoming IP, because of legal or contractual conflicts (e.g., two licenses specify different terms of use), or, for outgoing IP, as a matter of strategy (e.g., the owner of the IP wants to share one part while keeping the other part proprietary).

When IP rights are inconsistent, we say the IP is “incompatible.” In the Meissen case, the ruler introduced IP incompatibility as a matter of strategy: the IP status “known only to person A” and the IP status “known only to person B” are obviously incompatible. To accommodate this incompatibility, “IP modularity” was introduced in Meissen.

IP modularity. A particular module of a larger system is “IP-modular” if all of its elements have the same or compatible IP status. The module in question is then an “IP module.” If all of the modules of a system are IP modules, we say that the system as a whole is “IP-modular.”

Importantly, this definition implies that a system may use different, incompatible forms of IP and still be IP-modular, as long as the incompatible “chunks” of IP are associated with different modules of the overall system. For example, turning back to Figure 1, let A and B refer to incompatible forms of IP. The system shown in Figure 1(a) then is not IP-modular, while the one shown in Figure 1(b) is.

Note that a part of the system may be a product or process module without being an IP module: IP modularity is thus a stronger condition than “simple” product or process modularity. However, architects exercise control over the module boundaries and interfaces of the system, and (if the firm owns the IP in question) may also exercise control over the IP status accorded to different chunks of knowledge. Thus it is often possible to design or redesign a system to be “IP-modular,” if that is viewed as a desirable goal. Such redesign is illustrated in Figure 1. Starting from an integral system (a), IP modularity is established by introducing new module boundaries (b) (plus, removing one dependency and adding another). In Figure 2, the original system (a) is modular, but not IP-modular. Shifting the boundary creates a system (b) which is also IP-modular. Note that, in order to shift a module boundary, a designer may have to accept a relatively “thick” interface, illustrated in (b) by two linkages crossing the module boundaries.

---- *Insert Figure 2 about here* ----

A modular system is by definition IP-modular when all elements of the system have identical IP status. However, from our perspective, this is a trivial, not-very-interesting case. The concept is powerful when different elements of the system have *different* IP status. In this case, the IP modularity *of the system* becomes a crucial question for both system designers and strategic managers at the firm.

Trade-offs and costs. Even though the modularity of a product or process and its IP modularity are largely under the control of product or process architects, *neither type of modularity comes for free*. Strict product or process modularity with well-defined boundaries and interfaces is costly to achieve and maintain (Baldwin & Clark, 2000). It was possible to organize the porcelain factory as two non-overlapping task modules with a formal procedural interface, but it would have been easier to organize the work with overlapping tasks and informal handoffs. Similarly, segregating different chunks of knowledge and intellectual property is more expensive than leaving the knowledge and intellectual property unseparated. In the case of porcelain making, to maintain IP modularity, the ruler had to use soldiers to keep workers in separate zones within the fortress (Gleeson, 1998). In general, the advantages of modularity must always be weighed against the costs of achieving a particular modular structure and maintaining the module boundaries.

From a pure value creation standpoint, there will at any time be an optimal modular structure. However, the optimal modular structure for *capturing* value is not necessarily the same as the optimal structure for *creating* value. This tension between value creation and value capture and the need to trade-off benefits and costs make the strategic, IP-related use of modularity a significant issue for senior managers in industries with complex products and distributed IP.

Taxonomy: Types of IP modularization

Our analysis makes use of a 2x2 categorization scheme. First, the focal firm must distinguish between *outgoing* and *incoming* IP. Then, for each “chunk” of knowledge associated with the system, the firm must determine whether its IP status is *known* or *unknown*.

With *outgoing* IP, the focal firm owns the IP under consideration, and is free to award *to each element* the IP status it finds most advantageous. Thus it can grant to other agents bundles of IP rights and levels of de facto access, which may differ across both elements and agents. The focal

firm's ability to differentiate its IP is limited only by the cost of writing and enforcing different contracts and administering different levels of de facto access. With *incoming* IP, the roles are reversed: the owner of the IP has determined the IP status of the elements under consideration, and the focal firm must accept the bundle of rights and possibilities it has been granted.

The status of outgoing IP is *known* if, at the point in time when the product's architecture is chosen, the focal firm knows what combination of modular structure and IP configuration will be most advantageous over the lifetime of the system. It is *unknown* if, when the architecture is chosen, the optimal modular structure and IP status of the different components are not known or may change because of shifting demand or competitive threats. When the most desirable outgoing IP configuration is unknown or uncertain, an IP-oriented modularization of the system creates options that can be exercised at a later date.

The status of externally-sourced, incoming IP is *known* when the focal firm knows exactly the terms and conditions it must adhere to. It is *unknown* when the focal firm does not know the terms governing use of this knowledge. For example, a new product might inadvertently infringe upon some unknown and difficult-to-find patent, or a software codebase may contain copyrighted code that a programmer "borrowed" and passed off as his own work. Note that the level of knowledge of a system's IP status is in fact a continuous variable. We focus on the polar cases of "known" and "unknown" IP status in order to keep our taxonomy simple.

Figure 3 illustrates our taxonomic framework, and shows keywords for each of the case vignettes to be discussed in the next section.

---- *Insert Figure 3 about here* ----

ANALYSIS AND EMPIRICAL PREDICTIONS

In this section, we present case vignettes that flesh out the concept of IP modularity, and based on these cases, derive theoretical propositions that can serve as the basis of empirical tests. Our propositions address when IP modularity is advantageous and where the module boundaries should go. In selecting cases for analysis, we had three goals: First and foremost, we sought to illustrate the dependencies between IP modularity and value appropriation. Second, we wanted to show how firms can and do trade-off other benefits to achieve IP modularity. If IP modularity helps firms to appropriate value, they should be willing to incur costs or compromise on other value-creating dimensions of modularity to achieve the desired amount of IP modularity. And third, we sought to illustrate the breadth of the phenomenon.

In what follows, we discuss outgoing IP first and incoming IP second. Not surprisingly, strategies for outgoing IP are more complex and nuanced, thus seven of our ten propositions deal with outgoing IP and three with incoming IP.

Outgoing IP

The potential for value co-creation. In 2005, Israel-based M-Systems Ltd. faced a quandary with respect to its embedded flash memory products, widely used in mobile handsets. With the open source operating system Linux becoming popular for mobile devices, buyers of flash memory increasingly demanded that the memory's *driver* (the software that allows it to operate within the mobile device) be made publicly available as open source software (Kaplan, 2006). However, M-Systems wanted to protect the sensitive IP contained in the so-called *flash management software*, which in the first generation of the product was bundled with the driver.

As a solution, M-Systems introduced an IP-modular architecture. They redesigned their flash memory system, separating the flash management software from the driver software. They placed

the flash management code on the physical memory device itself, thus keeping it protected via copyright and secrecy. As Francois Kaplan, Associate Vice President of Product Marketing at M-Systems, explained: “*With sensitive IP now embedded in the storage device, flash vendors no longer need to worry about exposing competitive secrets*” (Kaplan, 2006). The remaining “thin” driver could now be published as open source software. As illustrated in Figure 1, an integral system affected by IP incompatibility (a) had been turned into an IP-modular system characterized by a mixed-IP strategy (b).

Importantly for our argument, M-Systems’s redesign of its flash memory system was costly in several ways. The company had to rewrite its software, redesign the physical memory device, and implement the new product design in their production facility. The fact that the firm was willing to incur these costs demonstrates the value of the resulting IP-modular architecture for M-Systems.

As in the case of Counter-Strike, IP modularity made it possible for M-Systems to reconcile value co-creation in other parts of the surrounding “ecosystem” (Iansiti & Levien, 2004) with value capture by the focal firm. Whenever the potential for such value co-creation exists, splitting a product into relatively “open” modules that allow for value co-creation and other “proprietary” or “closed” modules that enable value capture will make sense. This leads to our first general proposition:¹

Proposition 1: [Value Co-creation] The greater the potential for value co-creation in the surrounding ecosystem, the more advantageous is outgoing IP modularity. Module boundaries should go between the open and proprietary IP.

¹ We state some sub-propositions using the normative verb “should.” Every sub-propositions using the verb “should” can be reframed as a positive statement of the form, “The optimal location of the module boundaries is ...”

The benefits of IP modularity also depend on the extent to which value co-creation is distributed. Valve Software did not in the least know who of the tens of thousands of buyers of Half-Life would work on the game's code, and developers like to tinker with code rather than sign legal papers. Hence, requiring contracts and non-disclosure agreements (NDAs) with potential downstream developers was not a viable option for Valve. In contrast, if downstream value co-creation were going to be carried out by a single buyer, contractual IP protection (which, we note, is still an instance of selective openness) becomes much more feasible.

The case of M-Systems lies between these two extremes. Buyers of M-System's flash memory were handset makers, who were many fewer in number than the game players of Half-Life, and were known to M-Systems. Indeed in its first generation product, M-Systems relied on NDAs and licenses to protect its sensitive IP. However, its IP-oriented modularization meant that it no longer had to negotiate and enforce such contracts. Also, it enabled an open and distributed innovation process among its buyers which potentially improved the released driver (Käs, 2008). The following proposition summarizes this reasoning:

Proposition 2: [Distributed Co-creators] The more distributed, numerous, and anonymous the co-creators of value, the more advantageous is outgoing IP modularity.

Having established the general linkage between IP modularity and value co-creation, we now turn to propositions that depend on the relative position of value co-creators in the value chain.

Downstream value co-creation. Assume the focal product is sold to a systems integrator and becomes part of a larger system downstream. When this downstream system is complex, the system integrator typically assembles numerous pre-existing components. To optimize system-level performance, system integrators must often adapt the behavior of individual components. Such

adaptations can only take place if the system integrator knows and understands the components in some detail—thus systems integrators must “know more than they do” (Brusoni et al., 2001).

However, full knowledge by the system integrator of the design of a component may compromise the supplier’s intellectual property. Again, IP modularization provides a way out of this quandary. Quite often it is possible to segment the component’s design into information that is helpful to the integrator and information that pertains only to the internal workings of the component (Parnas, 1972; Brooks, 1975). From an IP perspective, the former can be treated as relatively “open” IP and the latter as more proprietary or “closed” IP. Again, the case of M-Systems provides a prime example. This leads to our third proposition:

Proposition 3: [Complexity] The more complex the downstream system, the more advantageous is an outgoing IP modularization. The module boundaries should separate the IP that helps to integrate the focal component into the system from that which contributes to the component’s internal performance.

IP modularity is also important when downstream user needs are heterogeneous and unpredictable, i.e., when users demand variety. One way to accommodate customers’ demand for variety is for the focal firm to provide a list of features or options that customers may select. However, this approach is often very costly, and may still not cover the full range of customer needs or desires.

Another way to address heterogeneous and unpredictable user needs is to split the product, and the related outgoing IP, into two sets of modules. Modules in the first set are unmodifiable and their IP is protected, while those in the second set are supplied in a modifiable fashion and under a more open IP status. In a similar way as with toolkits for user innovation (von Hippel, 2001), users’ modifications may then range from minor changes to entirely new modules. This example gives rise to our fourth proposition:

Proposition 4: [Customization] The greater and more varied the need for downstream adaptations, the more advantageous is outgoing IP modularization. The module boundaries should separate the IP that serves as the basis for modification from the IP supporting the proprietary “core” modules.

It is worth noting the link between Proposition 4 and “platform strategies” (e.g., Gawer & Cusumano, 2002). In a “platform architecture,” certain components (the platform) remain fixed, while others (the complements) are permitted to vary (Baldwin & Woodard, 2009). This strategy then permits one to give control over the design of complements to users or third parties. A platform strategy is a specific instance of IP modularization in the sense that, as MacCormack & Iansiti (2009) describe, the platform owner will often award a proprietary IP status to the core of the platform, and a more open IP status to the interface that allows others to link complementary components to it.

Upstream value co-creation. Sometimes value co-creation takes place upstream of the focal firm, i.e. by suppliers. In order to enable upstream value co-creation, the focal firm typically needs to provide outgoing IP such as blueprints or parts specifications to its suppliers. It can protect that knowledge via NDAs and licenses, but it perforce gives up the protection of secrecy. IP modularity, by partitioning knowledge and IP, reintroduces secrecy as a protective measure for some parts of the focal firm’s overall system, strengthening its ability to appropriate value in the supply chain.

There are two situations in which IP modularity vis a vis suppliers deserves special attention from a strategic perspective. First, when a given supplier also sells to the focal firm’s competitors, leakage of IP may result. As an example, consider the following case from the automotive industry. A supplier of braking systems sells its product to several auto manufacturers. One of its customers developed a stability control system which relied heavily on features of the braking system, in

particular on its ABS (anti-lock braking system). From a strictly technical perspective, it would have been optimal to integrate the braking and stability systems into one module.

Suppose the auto maker had gone this route and sought to co-develop an integrated braking-and-stability system in conjunction with its supplier. The supplier might then have sold two braking systems—one to the auto maker who provided the IP and one to its other customers. Or it might have sold the braking-and-stability system to all its customers. The second alternative was unacceptable to the auto maker, which considered its proprietary stability system to be an important point of product differentiation. The first would have required the brake supplier to maintain two distinct products, forgoing economies of scale, and would still have posed the risk of inadvertent leakage of critical IP.

The automaker instead developed the stability system as a separate module, which was added to the braking system during assembly. In other words, its desire to protect proprietary IP in the stability system caused the automaker to adopt a more modular—and IP-modular—design for its full braking system than it might have chosen had it been dealing with an inhouse division or a captive supplier. In the words of an interviewee from the auto company: “*Sometimes we need to re-segment both hardware and software modules, or the modularity of the system, based more on the commercial needs of, say, protecting an in-house algorithm, than on just the most efficient design.*” These considerations lead to our fifth proposition (which, we note, is the mirror image of Proposition 4):

Proposition 5: [Leakage of IP through a Common Supplier] IP modularity is more advantageous when the focal firm shares a common supplier with competitors. Module boundaries should separate the knowledge suppliers need to build components that work in the customer’s system from the knowledge contributing to features that differentiate the customer’s product from the competition.

We now turn to the second important instance of IP modularity vis a vis suppliers, and the related strategy of *divide et impera* (divide and rule). When a firm outsources a large portion of its production to external suppliers, it must be concerned that even as it gives suppliers the design knowledge they need to make products for its system, it is also giving them the knowledge they need to compete in its own markets. However, each supplier only needs access to knowledge that is relevant to the components it builds. Thus, if the focal firm takes care to outsource to different suppliers, which are kept from exchanging information with each other, critical IP can be protected by the fact that each supplier knows only part of the puzzle, but none can reconstruct the whole. IP modularization in this case requires breaking up the product design or production process into different task modules (e.g, making porcelain paste, making porcelain glaze), and then allocating those task modules to different suppliers.

In addition to the porcelain example discussed above, a modern case in point is that of SMaL, a maker of the core components for very small, low-power cameras.² SMaL sold a “camera kit” consisting of five components, two of which were the control chip and the imager chip. It would have been common practice to integrate these chips into one. However, SMaL decided against an integral design. IP considerations played a role in this decision. Separating the chips allowed SMaL to encase the control chip, which helped to conceal SMaL’s cost structure from buyers and to some extent also made reverse engineering more difficult. In addition, even though SMaL eventually decided to source both chips from one supplier (whom they judged to be very trustworthy), SMaL looked into having the two chips manufactured by different suppliers, to prevent one supplier knowing the design of these two key parts of the system.

² Sources: Interview with Maurizio Arienzo, former CEO of SMaL (9/10/2008), and Christensen & Anthony (2003).

IP-oriented modularizations based on the logic of “divide and rule” can strengthen intellectual property protection in otherwise weak appropriability regimes. This leads to our sixth proposition:

Proposition 6: [Divide and Rule] When the focal firm intends to outsource development or production but its IP is not strongly appropriable, IP modularity is advantageous. The module boundaries should divide the total knowledge needed to make the product into separate task modules, which can then be outsourced to different suppliers.

Note that—in line with our general argument about complementarity between IP modules—for the “divide and rule” strategy to work, it is essential that the IP allocated to the individual task modules have little or no value independent of the focal firm’s overall system.

Zhao (2006) provides a very clear example of a “divide and rule” strategy based on complementarity between “open” and “proprietary” modules, in the context of multinational companies (MNCs) managing R&D across international boundaries. The knowledge created through R&D cannot be protected effectively in countries with weak intellectual property rights (IPR). In our terminology, R&D projects in weak IPR countries are (process) modules with an unavoidably “open” IP status. According to Zhao, multinationals address this problem by assigning to weak IPR countries projects whose results are strongly complementary with those of R&D projects conducted in the U.S. In effect, the MNCs *divide* and *allocate* R&D projects so that the knowledge obtained in weak IPR countries has relatively little stand-alone value. This strategy in turn allows the MNCs to appropriate a greater portion of the aggregate value of their R&D investments. In support of this argument, Zhao presents evidence from patent citations that IP created in weak IPR countries has more value inside the focal firm than outside of it.

The effect of uncertainty. By creating IP-modular designs, firms also gain the ability to change their IP strategy in specific parts of the system in response to new value-capture opportunities. Thus, IP modularity creates option value.

Consider the example of the operating system Darwin, originated from Apple Inc.³ Originally, Apple kept the code that later became Darwin proprietary, thus from an IP perspective there was no reason not to adopt an integral design. However, in 2000 Apple made large parts of Darwin's source code—but not all of it—publicly available under an open source license, to take advantage of distributed value creation. According to our Propositions 1 and 2, IP modularity is desirable in such circumstances, but it is typically difficult and expensive to implement ex post. To the extent that Apple anticipated this move when designing the system, they might have built the option of selectively changing the IP status of certain parts into the program's architecture. That is, they may have performed an “anticipatory” IP modularization by creating a design which featured a high, seemingly excessive degree of modularity.

Generally, in settings characterized by high levels of technological or market uncertainty, changes in the external environment can arise after the fact which make a mixed-IP strategy desirable. For example, customers may require openness of some of the product's IP as a condition of purchase (e.g., Kas, 2008: 140), or outsourcing of production may become attractive for cost reasons. An “overly modular” initial design allows the focal firm to respond to such changes. In summary, we have:

³ Sources: [http://en.wikipedia.org/wiki/Darwin_\(operating_system\)](http://en.wikipedia.org/wiki/Darwin_(operating_system)) (accessed 03/25/2009). Note that Darwin was, even before its release under an open source license, largely based on licensed-in open source code. However, the respective licenses (mainly the BSD license) allowed keeping derived work proprietary, which is what Apple initially did.

Generally, such an “overly modular” initial design allows firms to respond to subsequent environmental changes that make a mixed-IP strategy desirable. Other examples of such changes are that customers may make openness of some of the product’s IP a purchasing criterion (e.g., Käs 2008: 140), or that outsourcing of production may become attractive for cost reasons. In summary, we have:

Proposition 7: [Uncertainty and Outgoing IP] An “overly modular” product or process architecture creates options to capture value in the future by selectively adapting the IP status of individual modules. Such options are more valuable in the presence of high market or technological uncertainty.

Incoming IP

We now turn to situations in which the focal firm does not control the IP status of some elements of its system. However, it can control its system’s modular architecture (subject to physical and cost constraints) and use modularity to mitigate various IP-related concerns.

Holdup risk. LaMantia et al. (2008) describe a software company whose entire product family depended on a central platform component. This platform contained both the company’s own code as well as code licensed-in from another software vendor, hence this component was not IP-modular. Furthermore, the platform was designed in an integral fashion, with the licensed-in code spread throughout the codebase. It would have been very difficult to separate the licensed-in code from the company’s own code on short notice. When the license came up for renewal and renegotiation, this fact would have exposed the firm to “holdup” on the part of the licensor (Williamson, 1979).

Anticipating this threat, the focal firm re-modularized the codebase, placing a newly designed platform and the licensed-in code in separate modules. Although the system required both

modules, the platform module no longer contained and did not depend on any licensed-in code. As a result, the focal firm's switching cost was greatly reduced, and risk of holdup was largely eliminated. This case vignette motivates our eighth proposition:

Proposition 8: [Holdup Risk] IP modularity is advantageous when the focal firm faces the risk of holdup from suppliers of incoming IP. The module boundaries should go between the firm's own IP and the incoming IP.

Proposition 8 can be reinterpreted in terms of Williamson's concept of asset specificity (Williamson, 1979). When a system has an integral (i.e., non-modular) design, its subsidiary components depend on each other in myriad ways. In this case, the firm's own assets (its code) became intertwined, hence highly specific, to assets it did not own (the licensed-in code). Asset specificity increases the threat of holdup, making armslength transactions more costly relative to transfers within a firm. As a result, transaction cost economics recommends that highly specific assets should be owned by a single firm. In LaMantia et. al.'s (2008) example, the focal firm took a different route: it reduced asset specificity by making the underlying system IP-modular.

IP Incompatibility. In the theoretical exposition in the previous section, we described how inconsistent legal or contractual restrictions can create incompatibility between different "chunks" of incoming IP. This problem arose at Sun Microsystems when it moved to license key implementations of its Java programming language as open source software. The technology magazine eWeek quotes Sun General Counsel Mike Dillon, saying the transition was tedious and legalistic:⁴ *"Java Standard Edition contains about 6 million lines of code. [...] Our legal team [of 190 lawyers] had to go over it, line by line, and look for all copyright marks and third-party*

⁴ See <http://www.eweek.com/c/a/Application-Development/Sun-Pours-Out-Java-Cup/> (accessed 01/06/2009).

involvements. Where Sun didn't have the correct licenses, we had to contact the owners, one by one, and determine the rights.”

In Sun's case, the problem was caused not by the opportunism of a single supplier, but by the sheer multitude and variety of upstream IP sources. Such multiplicity of sources is now a common occurrence in large software-intensive systems. It may be compounded by incompatibilities between different chunks of incoming IP and between incoming and own IP. In particular, the entanglement of incoming commercial IP with the firm's own IP creates a headache when, for strategic reasons, the focal firm wants to release part or all of its system as open source software to instigate external value creation.

We conclude that, when a firm has many, diverse sources of incoming IP, it should reduce the impact of IP incompatibilities by designing its whole system—whether it be a codebase, a product, or a process—in an IP-modular fashion. With software, doing so may even be a contractual obligation since, as discussed above, open source and commercial licenses often impose contradictory requirements on the licensee. This leads to

Proposition 9: [IP Incompatibility] IP modularity is advantageous when the focal firm obtains incoming IP on different terms from diverse sources. Module boundaries should be placed to ensure IP-compatibility within modules and to minimize the costs of renegotiation. In particular, it should be possible to change the IP status of a module without renegotiating numerous incoming IP licenses.

Incoming IP Under Uncertainty. We have already seen (Proposition 8) that IP modularity is a way to counter a known threat of holdup. Today, however, firms are increasingly vulnerable to IP-related threats that cannot be predicted when the product or process architecture is chosen. For a complex new product or process, in fields like electronics and software, it is often impossible to identify with certainty all patents that the product might infringe. As a result, innovators may

unexpectedly be faced with infringement allegations, and may be liable to pay damages and excessive licensing fees.

Patent “trolls” or “sharks” are firms that exploit such inadvertent infringement systematically. They often acquire obscure patents and then sue firms that unwittingly use the IP covered by the patent (Golden, 2007; Lemley & Shapiro, 2007; Reitzig et al., 2007; Henkel & Reitzig, 2007). A key aspect of the trolls’ strategy is surprise: ideally they want to wait until the IP-related product or process is successful before filing suit. In this way, they can maximize the value of any future damages or settlement.

One way to counter a patent infringement suit is to “design around” the patent by replacing the allegedly infringing component with a non-infringing substitute. Design-arounds in turn are less costly when the underlying product or process is modular. As long as the infringement is confined to one (or a few) modules, those modules can be redesigned or even removed without compromising the functionality of the rest of the system. In contrast, if the underlying product or process is integral, redesigning one component will trigger changes that propagate throughout the whole system (e.g., MacCormack et al. 2006: 1027). Thus a modular product or process architecture can reduce the cost of an unanticipated holdup based on IP claims to some part of the system (Henkel & Reitzig, 2008).

Consider the example of the MPEG-4 compression standard for audio and video data.⁵ MPEG-4 is not a single technology, but rather a collection of methods and so-called “parts.” For each application, MPEG-4 is implemented by generating a specific “profile” that assembles the parts (“modules” in our context) required for it. Since MPEG-4 is covered by a variety of patents

⁵ Sources: Interview with Klaus Diepold, member of the MPEG-4 standardization committee (07/24/2007); Henkel & Reitzig (2008); and <http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm> (accessed 01/07/2009).

under fragmented ownership, any firm implementing MPEG-4 in its products deals with incoming IP. Worse, however, is the uncertainty arising from unidentified patents pertaining to the technologies embodied in the standard. After wide adoption, a hitherto unknown patent owner might use its IP rights to extract high licensing fees and royalties. Indeed this is what happened in the case of the related compression standard JPEG. Forgent Networks collected, in less than three years, more than US\$ 100 million in licensing fees after suing users of the standard for inadvertent infringement of one of its patents (Reitzig et al., 2007, footnote 6).

IP modularity mitigates this problem. The modular structure of MPEG-4 allows firms, in defining a “profile,” to leave out parts with a high perceived IP uncertainty. Furthermore, when elements under uncertain IP status are encapsulated into a separate module of the product, it will be easier to replace or even drop them if it turns out that they infringe on so far unidentified patents. This leads to our last proposition:

Proposition 10: [Inadvertent Infringement] A modular product or process architecture is a partial defense against unanticipated incoming IP claims, because it gives the firm options to “design-around” the claims without redesigning the whole system.

DISCUSSION AND CONCLUSIONS

In concluding, we want to point out generalizations of the concept of “IP modularity,” discuss a number of caveats, and summarize.

For clarity in exposition, we have focused in our analysis of IP modularity on the products firms sell and the processes they use. However, the concept extends to organizations. “Chinese Walls” (McVea, 1993) are an illustrative example. This term refers to virtual barriers between different organizational units which prevent information exchange between these units—the financing and the share dealing department of a merchant bank, for example. These units, in our

notion, then constitute “IP modules” within the organization, and the Chinese Walls between them are module boundaries. Similarly, Liebeskind (1997) proposes “structural isolation” of organizational units that deal with sensitive knowledge as a mechanism to keep that knowledge secret. These units thus constitute organizational IP modules under a strict proprietary IP status.

A second generalization concerns the rights and obligations under consideration. While we focused on IP as an important source of such rights and obligations, the latter may also have other legal or contractual origins. For example, the need to clearly attribute liability in case of product failures may, in a similar way to IP, give rise to a modular architecture. In safety-critical products, modules may be segregated due to differing safety certification levels. And classified data may need to be stored and processed separately from unclassified data, again suggesting a modular architecture (Ames, 2003).

Also the concept of “IP incompatibility,” discussed in the theory section and in Proposition 9, may be generalized. It is applicable not only to the IP-dimension of product modularity, but also to other dimensions of coupling between the elements of a complex artifact. It thus constitutes an extension of the theory of modularity in general. As an example of “function-based” antagonistic coupling in software, consider two elements competing for a common resource (e.g., memory). Putting them into the same component may lead to conflicting access attempts to that same resource. In contrast, locating them in different components avoids these conflicts by making use of component-level resource management. A case of “organization-based” antagonistic coupling is given, e.g., when different compensation levels between groups of employees make it preferable to keep them in separate organizational units.

Finally, mixed-IP strategies are not restricted to modules alone. The interfaces between the modules and the overall architecture of the system may also have distinct IP status. For example, modules may be kept proprietary, while interfaces and architecture are openly disclosed.

We now briefly address approaches to testing our propositions empirically. First, one has to choose an industry characterized by complex products and processes, such as software or electronics. Then, one needs to devise a way to measure the degree of modularity of artifacts, and to quantify the extent to which individual modules carry diverse IP status. Depending on the proposition under consideration, one must finally operationalize the respective independent variable—among others, the potential for value co-creation, the degree of distributedness of value co-creators, and the level of complexity of downstream systems.

While the above can in principle be achieved for any industry, the software industry appears particularly suitable. Unlike processes or hardware products, software can be analyzed in an automated fashion with respect to its modularity (e.g., MacCormack, et al. 2006) and even, to some extent, with respect to the modules' IP status (Black Duck, 2008). One may then either assume that actors behave rationally, and take the occurrence of IP modularity as an indication of its being advantageous. Alternatively, one may relate the degree of IP modularity to some measure of success in value appropriation, although this approach is typically challenging due to confounding effects.

Before closing the paper, we must emphasize a few caveats and limitations. First and foremost, modularity almost always comes at a cost. Thus IP modularity, which we have said is stricter than simple modularity, is likely to increase the cost of design, require legal clarifications, and may imply a loss of performance. IP strategists must evaluate these costs in relation to the perceived benefits of IP modularity.

Second, in some cases the best form of IP protection is to adopt an integral (non-modular) system architecture, because a modular architecture provides more opportunity for imitation and entry (Baldwin & Clark, 2000; Ethiraj et al., 2008; Fixson & Park, 2008). IP modularity becomes a strategic issue when the firm believes there are significant advantages (for example, value co-creation) to adopting a modular system design.

Third, in some circumstances, even with a modular product or process architecture, IP modularity may not be necessary. For example, if a company's strategy for appropriating value rests heavily on ownership of complementary assets (Teece, 1986), that company may not need to keep and protect a "proprietary" IP module. Similarly, trust between organizations reduces the risk of opportunistic behavior, and thus may reduce the need to pursue IP modularity or define a mixed-IP strategy.

Finally, since the concept of IP modularity applies naturally to products and processes alike, it is appropriate to address both in parallel. Doing so requires a certain level of abstraction, which is beneficial in bringing out the fundamental mechanism and the breadth of the concept. However, this abstraction necessarily limits the depth in which we can discuss IP modularity for each application, products and processes. Similarly, the industries and technologies we cover in our case vignettes range from software to automobiles to porcelain. Again, such variety is desirable for our ends, but unavoidably limits the level of detail of our discussion. Future research should address these issues, and should in particular provide industry-specific studies.

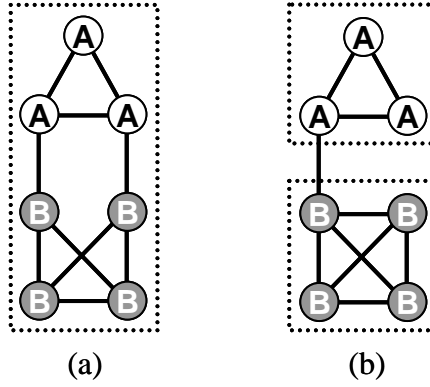
In conclusion, in this paper we have explored the concept of IP modularity. Fundamentally IP modularity eliminates incompatibilities between IP rights in a given module, while permitting incompatibilities within the overall system. This in turn permits a firm to pursue a mixed-IP strategy. In this fashion, IP modularity overcomes intrinsic conflicts between distributed value creation on the one hand and value appropriation on the other hand.

The details of IP modularization must be determined by engineers and legal experts working together. But beyond the technical and legal concerns, IP modularity affects a firm's strategies for value appropriation in increasingly complex and fragmented technological spaces. It is an issue that deserves the attention of general managers and management scholars generally.

FIGURES

FIGURE 1

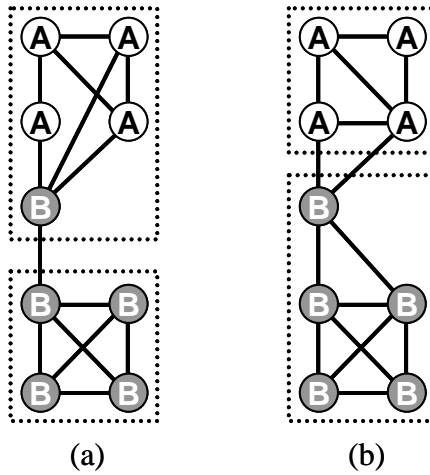
Systems with integral (a) and IP-modular (b) structure.



Note: Elements A, B are under distinct IP status.

FIGURE 2

Systems with modular (a) and IP-modular (b) structure.



Note: Elements A, B are under distinct IP status.

FIGURE 3

Taxonomy of types of IP modularization.

		<i>Knowledge about IP status</i>	
		known	unknown
<i>Ownership of IP</i>	outgoing IP: IP status of elements can be specified	Counter-Strike M-Systems stability control system	Apple's Darwin operating system
	incoming IP: IP status of elements externally given	software platform (La Mantia et al., 2008) Java as OSS	patent trolls

REFERENCES

- Alexy, O., & Henkel, J. 2007. *Promoting the penguin: Who is advocating open source software in commercial settings?* Working paper, Technische Universität München.
http://papers.ssrn.com/sol3/papers.cfm?abstract_id=988363.
- Ames, B. 2003. Real-time software goes modular. *Military & Aerospace Electronics*, September.
- Baldwin, C. Y. 2008. Where do transactions come from? Modularity, transactions, and the boundaries of firms. *Industrial and Corporate Change*, 17(1): 155-195.
- Baldwin, C. Y., & Clark, K. B. 1997. Managing in an age of modularity. *Harvard Business Review*, September-October: 84-93.
- Baldwin, C. Y., & Clark, K. B. 2000. *Design Rules, Volume 1, The Power of Modularity*. Cambridge, MA: MIT Press.
- Baldwin, C. Y., & Clark, K. B. 2006. The architecture of participation: Does code architecture mitigate free riding in the open source development model? *Management Science*, 52(7): 1116–1127.
- Baldwin, C. Y., & Woodard, C. J. 2009. The architecture of platforms: A unified view. In A. Gawer (Ed.), *Platforms, Markets, and Innovation*. Edward Elgar, forthcoming.
- Barney, J. B. 1991. Firm resources and sustained competitive advantage. *Journal of Management*, 17(1): 99–120.
- Black Duck 2008 *Software Compliance Management*. Company white paper.
<http://ducks.blackducksoftware.com/~whitepapers/WP-SP-0805-UL-AA-SCM-Web.pdf>
- Blyler, M., & Coff, R. W. 2003. Dynamic capabilities, social capital, and rent appropriation: ties that split pies. *Strategic Management Journal*, 24(7): 677-686.
- Bowman, C., & Ambrosini, V. 2000. Value creation versus value capture: Towards a coherent definition of value in strategy. *British Journal of Management*, 11(1): 1-15.

- Brooks, F. P. 1975. *The Mythical Man Month: Essays on Software Engineering*. Addison-Wesley.
- Brusoni, S., & Prencipe, A. 2006. Making design rules: A multidomain perspective. *Organization Science*, 17(2): 179–189.
- Brusoni, S., & Prencipe, A., Pavitt, K. 2001. Knowledge specialization, organizational coupling, and the boundaries of the firm: Why do firms know more than they make? *Administrative Science Quarterly*, 46: 597–621.
- Chesbrough, H. W. 2003. *Open Innovation. The New Imperative for Creating and Profiting from Technology*. Boston: Harvard Business School Press.
- Chesbrough, H. W., & Kusunoki, K. 2001. The modularity trap: Innovation, technology phase-shifts, and the resulting limits of virtual organizations. In I. Nonaka, D. Teece (Eds.), *Managing Industrial Knowledge*: 202–230. London, UK: Sage Press.
- Chesbrough, H. W., & Teece, D. J. 1996. When is virtual virtuous? Organizing for innovation. *Harvard Business Review*, 74 (January/February): 65–73.
- Christensen, C. M., Anthony, S. D. 2003. Making SMaL Big: SMaL Camera Technologies. *Harvard Business School Case* N-603-116.
- Coff, R. W. 1999. When competitive advantage doesn't lead to performance: the resource-based view and stakeholder bargaining power. *Organization Science*, 10(2): 119-133.
- Cohen, W.M., Nelson, R.R., & Walsh, J.P. 2000. Protecting their intellectual assets: Appropriability conditions and why U.S. manufacturing firms patent (or not). *NBER Working Paper* w7552.
- Daft, R. 1983. *Organization Theory and Design*. New York: West.
- Ethiraj, S. K., Levinthal, D. A., & Roy, R. R. 2008. The dual role of modularity: Innovation and imitation. *Management Science*, 54 (5): 939-955.

- Ferguson, C.H., & Morris, C.R. 1993. *Computer Wars: How the West Can Win in a Post-IBM World*. New York: Times Books.
- Fine, C. H. 1998. *Clockspeed: Winning Industry Control in the Age of Temporary Advantage*. Cambridge, MA: Da Capo Press.
- Fixson, S. K., & Park, J.-K. 2008. The power of integrality: Linkages between product architecture, innovation, and industry structure. *Research Policy*, 37(8): 1296-1316.
- Foray, D. 2004. *The Economics of Knowledge*. Cambridge, MA: MIT Press.
- Franke, N., & von Hippel, E. 2003. Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software. *Research Policy*, 32(7): 1199–1215.
- Free Software Foundation. 1991. *The GNU General Public License (GPL) – Version 2, June 1991*. <http://www.opensource.org/licenses/gpl-2.0.php>.
- Garud, R., & Kumaraswamy, A. 1995. Technological and organizational designs for realizing economies of substitutions. *Strategic Management Journal*, 16: 93-110.
- Gawer, A., & Cusumano, M. A. 2002. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Boston, MA: Harvard Business School Press.
- Gleeson, J. 1998. *The Arcanum: The Extraordinary True Story*. New York: Warner Books.
- Golden, J. M. 2007. “Patent trolls” and patent remedies. *Texas Law Review*, 85: 2111-2162.
- Gomez, P. J., & Joglekar, N. R. 2008. Linking modularity with problem solving and coordination effort. *Managerial and Decision Economics*, 29: 443-457.
- Gulati, R. 1998. Alliances and networks. *Strategic Management Journal*, 19: 293-317.
- Harhoff, D., Henkel, J., & von Hippel, E. 2003. Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations. *Research Policy*, 32(10): 1753–1769.

- Henderson, R. M., & Clark, K. B. 1990. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35(1): 9-30.
- Henkel, J. 2006. Selective revealing in open innovation processes: The case of embedded Linux. *Research Policy*, 35(7): 953-969.
- Henkel, J., & Reitzig, M. 2007. *Patent sharks and the sustainability of value destruction strategies*. Working paper, available at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=985602.
- Henkel, J., & Reitzig, M. 2008. Patent sharks. *Harvard Business Review*, June: 129-133.
- Hoetker, G. 2006. Do modular products lead to modular organizations? *Strategic Management Journal*, 27: 501-518.
- Iansiti, M., & Levien, R. 2004. *The Keystone Advantage*. Cambridge, MA: Harvard Business School Press.
- Jacobides, M. G. 2005. Industry change through vertical disintegration: How and why markets emerged in mortgage banking. *Academy of Management Journal*, 48(3): 465-498.
- Jeppesen, L. B. 2004. Profiting from innovative user communities: How firms organize the production of user modifications in the computer games industry. *Copenhagen Business School Working Paper 2004-03*. <http://ep.lib.cbs.dk/download/ISBN/8778690978.pdf>.
- Käs, S. 2008. *Rethinking Industry Practice: The Emergence of Openness in the Embedded Component Industry*. Berlin: Pro BUSINESS.
- Kaplan, F. 2006. Opening the door for the latest NAND flash in open source mobile platforms. *LinuxDevices.com*, <http://www.linuxdevices.com/articles/AT2185129745.html>.
- Kash, D., & Kingston, W. 2001. Patents in a world of complex technologies. *Science and Public Policy*, 28(1): 11-22.

- LaMantia, M. J., Cai, Y., MacCormack, A. D., & Rusnak, J. 2008. Analyzing the evolution of large-scale software systems using design structure matrices and design rule theory: Two exploratory cases. *Seventh Working IEEE/IFIP Conference on Software Architecture*: 83-92.
- Langlois, R. N. 2002. Modularity in technology and organization. *Journal of Economic Behavior & Organization*, 49(1): 19–37.
- Langlois, R. N. 2003. The vanishing hand: The changing dynamics of industrial capitalism. *Industrial and Corporate Change*, 12(2): 351–385.
- Langlois, R. N., & Garzarelli, G. 2008. Of hackers and hairdressers: Modularity and the organizational economics of open-source collaboration. *Industry & Innovation*, 15(2): 125-143.
- Langlois, R.N., & Robertson, P.L. 1992. Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries. *Research Policy*, 21(4): 297–313.
- Lavie, D. 2006. The competitive advantage of interconnected firms: an extension of the resource-based view. *Academy of Management Review*, 31(3): 638–658.
- Lavie, D. 2007. Alliance portfolios and firm performance: a study of value creation and appropriation in the U.S. software industry. *Strategic Management Journal*, 28(12): 1187-1212.
- Lemley, M. A., & Shapiro, C. 2007. Patent holdup and royalty stacking. *Texas Law Review*, 85: 1991-2049.
- Lepak, D. P., Smith, K.G., & Taylor, M.S. 2007. Value creation and value capture: A multi-level perspective. *Academy of Management Review*, 32 (1): 180-194.
- Levin, R.C., Klevorick, A., Nelson, R.R., & Winter, S.G. 1987. Appropriating the returns from industrial research and development. *Brookings Papers on Economic Activity*, 3: 783–820.
- Liebeskind, J. P. 1997. Keeping organizational secrets: Protective Institutional Mechanisms and their costs. *Industrial and Corporate Change*, 6(3): 623-663.

- MacCormack, A., & Iansiti, M. 2009. Intellectual property, architecture, and the management of technological transitions: Evidence from Microsoft Corporation. *Journal of Product Innovation Management*, forthcoming.
- MacCormack, A., Rusnak, J., & Baldwin, C. Y. 2006. Exploring the structure of complex Software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7): 1015–1030.
- Matutes, C., & Regibeau, P. 1988. “Mix and Match”: Product Compatibility without Network Externalities. *RAND Journal of Economics*, 19(2): 221-234.
- McVea, H. 1993. *Financial Conglomerates and the Chinese Wall: Regulating Conflicts of Interest*. Oxford, New York: Clarendon Press.
- Parnas, D. L. 1972. A technique for software module specification with examples. *Communications of the ACM archive*, 15(5): 330-336.
- Pénin, J. 2007. Open knowledge disclosure: An overview of the empirical evidence and the economic motivations. *Journal of Economic Surveys*, 21(2): 326-348.
- Reitzig, M., Henkel, J., & Heath, C. H. 2007. On sharks, trolls, and their patent prey – Unrealistic damage awards and firms’ strategies of ‘being infringed’. *Research Policy*, 36: 134–154.
- Reitzig, M., & Puranam, P. 2008. *Value appropriation as an organizational capability: The case of IP protection through patents*. Working paper, London Business School.
<http://ssrn.com/abstract=957335>.
- Rumelt, R. P. 1984. Towards a strategy theory of the firm. In B. Lamb (Ed.), *Competitive Strategic Management*. Englewood Cliffs, NJ: Prentice-Hall.
- Sako, M. 2003. Modularity and outsourcing: The nature of co-evolution of product architecture and organization architecture in the global automotive industry. In: A. Prencipe, A. Davies, M. Hobday (Eds.), *The Business of Systems Integration*: Oxford, UK: Oxford University Press.

- Sanchez, R. 1995. Strategic flexibility in product competition. *Strategic Management Journal*, 16: 135-159.
- Sanchez, R., & Mahoney, J. T. 1996. Modularity, flexibility, and knowledge management in product and organizational design. *Strategic Management Journal*, 17 (winter special issue): 63–76.
- Sanderson, S., & Uzumeri, M. 1995. Managing product families: The case of the Sony Walkman. *Research Policy*, 24(5): 761–782.
- Schilling, M. A. 2000. Toward a general modular systems theory and its application to inter-firm product modularity. *Academy of Management Review* 25(2): 312–334.
- Schilling, M. A., & Steensma, H. K. 2001. The use of modular organizational forms: An industry-level analysis. *Academy of Management Journal*, 44(6): 1149-1168.
- Simon, H. A. 1962. The architecture of complexity. Proceedings of the American Philosophical Society 106, 467-482; reprinted in: *The Sciences of the Artificial*, 2nd edition. MIT Press, Cambridge, MA (1981).
- Staudenmayer, N., Tripsas, M., & Tucci, C. L. 2005. Interfirm modularity and its implications for product development. *Journal of Product Innovation Management*, 22: 303–32.
- Sturgeon, T. 2002. Modular production networks: A new model of industrial organization. *Industrial and Corporate Change*, 11(3): 451–496.
- Takeishi, A. 2002. Knowledge partitioning in the inter-firm division of labor: The case of automotive product development. *Organization Science*, 13(3): 321–338.
- Teece, D.J. 1986. Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, 15(6), 285–305.

- Teece, D. J. 1988. Technological change and the nature of the enterprise. In G. Dosi, C. Freeman, R. R. Nelson, G. Silverberg, L. Soete (Eds.), *Technical Change and Economic Theory*: 256–281. London: Pinter.
- Teece, D. J. 2000. *Managing Intellectual Capital: Organizational, Strategic, and Policy Dimensions*. Oxford, U.K.: Oxford University Press.
- Ulrich, K. T. 1995. The role of product architecture in the manufacturing firm. *Research Policy*, 24: 419–440.
- von Hippel, E. 1990. Task partitioning: An innovation process variable. *Research Policy*, 19(5): 407–418.
- von Hippel, E. 2001. Perspective: User Toolkits for Innovation. *Journal of Product Innovation Management*, 18: 247–257.
- von Hippel, E., & Finkelstein, S. N. 1979. Analysis of innovation in automated clinical chemistry analyzers. *Science & Public Policy*, 6(1): 24-37.
- von Hippel, E., & von Krogh, G. 2003. Open source software and the “private-collective” innovation model: Issues for organization science. *Organization Science*, 14(2): 209-223.
- Wernerfelt, B. 1984. A resource-based view of the firm. *Strategic Management Journal*, 5: 171-180.
- West, J. 2003. How open is open enough? Melding proprietary and open source platform strategies. *Research Policy*, 32(7): 1259-1285.
- Wheelwright, S. C., & Clark, K. B. 1992. *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality*. New York: Free Press.
- Williamson, O. E. 1979. Transaction-cost economics: The governance of contractual relations. *Journal of Law and Economics*, 22: 233-261.