HARVARD | BUSINESS | SCHOOL

IP Modularity: Profiting from Innovation by Aligning Product Architecture with Intellectual Property

Joachim Henkel
Carliss Y. Baldwin
Willy C. Shih

Working Paper

13-012

November 30, 2012

IP Modularity: Profiting from Innovation by Aligning Product Architecture with Intellectual Property

This version: November 2012

Joachim Henkel[1], Carliss Y. Baldwin[2], Willy C. Shih[2]

[1]Technische Universität München, Arcisstr. 21, 80333 Munich, Germany, henkel@wi.tum.de.

[2]Harvard Business School, Soldier's Field, Boston, MA 02163, USA, cbaldwin@hbs.edu, wshih@hbs.edu.

Abstract

In this paper we explain how firms seeking to take advantage of distributed innovation and outsourcing can bridge the tension between value creation and value capture by modifying the modular structure of their technical systems. Specifically, we introduce the concept of "IP modularity", a special form of modularity that seeks to protect and capture value from intellectual property (IP). We define what it means for a system to be "IP-modular," and illustrate the application of this concept in a number of practical situations. From the examples, we derive a comprehensive framework that can be used to design and evaluate value capture strategies for modular systems.

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

We are grateful to many people who shared their insights or commented on earlier versions of the paper. Special thanks go to Sung Joo Bae, Simon Bolten, Timo Fischer, Deborah Gray, 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, Gianmario Verona 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.

IP Modularity: Profiting from Innovation by Aligning Product Architecture with Intellectual Property

Firms today increasingly practice open innovation, license technology in and out, outsource development and production, and enable users and downstream firms to innovate on their products. Distributing activities across firms with heterogeneous capabilities can increase the total value of a system, but it may also lead to challenges in capturing value and profiting from innovation. Specifically, innovators may grant access to knowledge, giving up or sharing intellectual property (IP) in order to attract external contributions or access external capabilities. But giving up IP can open up Pandora's box by giving others the opportunity to claim most of the jointly created value.

How can firms obtain the benefits of distributed innovation and outsourcing while avoiding the concomitant risks to value capture? Our paper addresses this tension by proposing the novel concept of "IP modularity." IP modularity is a form of modularity that takes into account a firm's need to capture value from IP. We argue that by managing a system's modular structure in conjunction with its IP, firms can reconcile opportunities for distributed innovation and outsourcing with their need to capture value. Simply said, a firm must simultaneously decide on the technical boundaries of product and process modules and the IP deployed in each one. Controlling too much of the system's IP is problematic if it deters innovation by others. But controlling too little—or the wrong parts—may prevent the focal firm from capturing value or expose it to the risk of hold-up. Only by managing a system's modular structure in conjunction with its IP can firms reconcile opportunities for distributed value creation with their need to capture value.

Consider the example of M-Systems, a subsidiary of the leading seller of flash memory, SanDisk.[1] As its sales grew, its customers increasingly demanded access and the right to modify the "driver" software that allows the flash memory to interoperate with the host device. M-Systems wanted to protect sensitive IP which was part of the driver, so they introduced what we call an IP-modular architecture. They split the driver into two modules, based on IP considerations: the flash management code, and a remaining "thin" driver. They placed flash management code on the physical device itself, protecting it via copyright and secrecy, and published the thin driver as open source software. Redesigning the technical modules to match the

desired IP structure allowed M-Systems to reconcile customer demands for openness with their own value appropriation strategy.

M-Systems owned all the IP in the original driver and partitioned it into a part they controlled closely and another part they shared widely. In other cases, the focal firm might not own all the relevant IP. For example, an automaker sourced an antilock braking system (ABS) from a supplier that sold the same system to other firms in the industry. Within the automobile, the ABS was a physically distinct component and all of its IP was owned by the supplier. The automaker developed a stability control system, which relied heavily on features of the ABS. From a technical perspective, it would have been optimal to integrate the braking and stability systems into one module. However this module would not have been "IP-modular" because it would have relied on IP from both the supplier and the automaker. To avoid commingling its own and external IP, the automaker instead chose to develop the stability control system as a separate module, which was added to the braking system during assembly. In the words of an executive of the 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.*" [2]

IP modularity is an important concept for firms that must manage IP across firm boundaries. This includes firms that give their IP away "openly" and firms that use IP obtained from broadly distributed innovation "ecosystems." It includes firms that must seek to share IP with suppliers, employees, and alliance partners; license IP from outsiders; and want to protect their own IP from imitation. Finally, it includes firms that are simply uncertain about the IP opportunities and threats they will face in the future.

In this paper, we explain how firms seeking to take advantage of distributed innovation and outsourcing can use IP modularity to capture value. We first introduce and then formally define what it means for a system to be "IP-modular." We then illustrate the application of this concept in practice. To organize our analysis, we identify four strategies and two sources of IP (internal and external). The interaction of strategy and IP source in turn creates eight generic situations. For each situation, we present case examples from our research that show how IP modularity was used in that context to capture value. At the end of the paper, we pull the separate strands of analysis together into a comprehensive framework, discuss contextual factors, and describe extensions and limitations.

Introducing IP Modularity

To create value in large systems with decentralized innovation, it is necessary for the parts to be independent so that innovation in one part of the system will not require changes in many other parts.[3] Modularity is a means to achieve this end. From a technical perspective, modularity allows tasks to be partitioned and worked on in parallel.[4] This in turn facilitates the division of labor in the innovation process between firms[5]—an approach aptly labeled "open innovation" by Henry Chesbrough.[6] In a modular system, innovations can be introduced by firms and individuals other than the original innovator—firms in related markets, entrepreneurial startups, or even users.[7]

"IP modularity" is a form of modularity that seeks to protect and capture value from IP. Frequently, as in the brake-and-stability system example, the focal firm must compromise on technical dimensions to achieve IP modularity. The concept of IP modularity in turn rests on the concepts of "IP status" and "IP incompatibility."

A module's "IP status" describes the legal rights to and *de facto* accessibility of knowledge embodied in the module. In general, the "owner" of IP can exclude others from using his knowledge through some combination of enforceable legal rights (patents, copyrights, legally-sanctioned trade secrets) and direct control of access to information (controlling who knows what). The owner of IP can grant others access to her IP by selectively transferring some of her rights, or by publishing proprietary information and waiving control of it. For example, the owner may keep knowledge related to a given module secret, make it available only to select parties, disclose it publicly while asserting patents on it, or disclose it freely to all parties.

Different parts of a system can have different IP statuses if they have different owners or if a single owner wishes to treat them differently for strategic reasons. For example, in the case of M-Systems, the original flash memory driver had the IP status "binary code copyright protected, source code kept secret." After introduction of the IP-modular architecture the flash management software retained this status, while the thin driver took on the IP status "source code open, far-reaching rights granted to others under an open source license (GPL)."[8]

Building on this definition, we say that "IP incompatibility" exists between two elements of a system if their IP statuses lead to conflicting obligations or appropriability. For M-Systems, the flash management software needed strong IP protection (realized by copyright and secrecy) to minimize the risk of imitation. But the driver code needed to be open—accessible to those who

would integrate the flash memory into their devices. Thus, the two parts of the system exhibited IP incompatibility in that the desired IP status for one was incompatible with the desired IP status for the other.

The stability control/antilock braking system suffered from a different form of IP incompatibility. From the automaker's perspective, the stability control system had the intended status "own IP, secret and copyrighted (and likely patented)," while the antilock braking system had the status "owned by the supplier." Building a single-module all-purpose braking system might have been the ideal engineering solution, but the resulting product would have relied on both the supplier's and the automaker's IP. In that case, the automaker would have been concerned about leakage of its IP to competitors, while the supplier might worry about how its *other* customers would react to knowledge-sharing with the automaker. Finally, both firms would need to evaluate the risk of "hold-up," that is, the opportunistic withdrawal of rights to use IP.[9] Each of these concerns is a reason to place IP owned by different parties in separate modules. We discuss them in more detail below.

With these definitions, we can now introduce "IP modularity" formally. We define an "IP-modular" system architecture as one in which the boundaries of parts with different IP status coincide with the technical boundaries of modules. In particular, if IP incompatibilities are present within the system, *they exist only between, but not within modules*.

For illustration, let A and B in Figure 1 refer to incompatible forms of IP. (A might refer to M-Systems' proprietary code and B to open source code. Alternatively A might refer to the automaker's patents and B to the supplier's patents.) The system shown in Figure 1(a) is not IP-modular, while the one shown in Figure 1(b) is: a module boundary has been introduced that separates elements with different IP statuses. IP modularity may also be achieved by shifting a module boundary, as shown in Figure 2. Both 2(a) and 2(b) show modular systems, but only 2(b) is also IP modular.

---- *Insert Figures 1, 2 about here* ----

IP modularity avoids conflicts, but does not come free. Segregating different chunks of knowledge according to their actual or desired IP status is more expensive than pursuing the best technical solution. For both M-Systems' driver and the automaker's stability control system, a one-module solution would likely have been technically superior. But as we've said, the best way

to capture value is not necessarily the best way to create value. In practice, there is tension between value creation and value capture, hence the need to trade off benefits and costs.

IP Modularity in Practice

To make appropriate tradeoffs between value creation and value capture, managers must consider their own firm's strategic goals, and whether the relevant IP is owned by their firm or an outside party. There are in turn four strategies that are likely to raise IP concerns in modular systems (these correspond to the rows of Table 1):

(1) The firm may seek to create value *broadly* by publishing its own IP and by utilizing external "open" IP from a surrounding ecosystem. In such cases, the firm must be sure to protect some of its own IP for value appropriation purposes and must also design systems to efficiently manage external IP. A firm pursuing this strategy requires broad, "open" licenses, standardized policies, and efficient procedures for tracking IP within the firm.

(2) The firm may seek to create value *narrowly* by sharing its own IP with selected suppliers, employees, and alliance partners. In such cases, the firm must act to avoid leakage of critical IP (beyond the designated recipients), and also manage the risk of "hold-up" by external IP owners. A firm pursuing this strategy must manage a set of specific contracts and key relationships.

(3) Faced with uncertainty about its own strategy and/or the intentions of third parties, the firm may seek to create options that allow it to change the IP status of particular components in response to future events.

(4) Finally, the firm may seek to exert control over weakly protected IP by combining it with its own strongly protected IP in a single module. The weakly protected IP may be sourced internally or externally; the strongly protected IP must be internal.

The first three strategies require dividing IP across separate modules: the fourth involves combining IP within a single module.

---- *Insert Table 1 about here* ----

Table 1 summarizes the analytic framework we have just laid out. As shown in the table, there are eight combinations of strategies and IP sources. In our research, we have identified cases showing how IP modularity was achieved for each combination of strategy and IP source. We

describe these cases below, with each section keyed to a corresponding cell in the table. At the end of each section, we offer a recommendation appropriate to that strategy and IP source.

However, a particular firm can pursue different strategies simultaneously in different parts of a modular system. For example, at one point in time, a firm may be seeking to reveal some IP to all comers; share other IP only with designated partners; hedge against future uncertainties; and exert control over IP used in critical components. Thus each combination of strategy and IP source illustrates only one dimension of IP modularity. After considering each dimension separately, we will synthesize our findings to arrive at a comprehensive approach that is applicable in practice.

Supporting Distributed Innovation in a Large Ecosystem

Today, many firms participate in so-called "ecosystems" where innovation activities are distributed across many firms and individuals.[10] The strategic challenge here is to attract external innovators who can "co-create value" by identifying new functions and designing novel solutions for a given technological system. In general, firms will both give and take IP from the ecosystem.

Enabling Value Co-Creation by Free Revealing (1-A)

M-Systems revealed some of its proprietary IP to the world when it published the "thin" driver as open source software. On the one hand, this change of policy was costly: the company had to rewrite its software, redesign the physical device, and manufacture the new product design. On the other hand, by revealing how the device read commands from the operating system, the firm made it easier to incorporate its flash memory into mobile devices like telephones. Thus M-Systems facilitated external value co-creation by device manufacturers and open source developers. At the same time, its managers protected what they deemed to be critical IP by creating two modules with an IP-modular architecture.

The benefits of broadly releasing proprietary IP depend on the extent to which co-creators of value are widely distributed. If potential co-creators of value are unknown to the focal firm or if the quality of their efforts cannot be determined in advance, then their contributions cannot be secured by means of standard contracts (including IP licenses). In such cases, unilaterally conveying knowledge and related IP rights to all comers via unprotected open modules may be the only way for the focal firm to gain access to these valuable resources.[11] At the same time, to maintain exclusivity, the focal firm must protect some of its own IP, and thus an IP-modular architecture is called for.

Many computer chip manufacturers, like Nvidia, provide "reference designs" which include sample hardware implementations and driver software. In this fashion, they split their own IP: keeping the majority proprietary, they create "open" modules that include all the designs, electronic files, and test programs that a systems manufacturer might need. (Nvidia manages its open resources carefully. Anyone can enter its "developer zone", but to access useful content, developers must apply to particular areas and submit technical data about their project. Nvidia also retains the right to deny anyone access to the zone: "If you're rude, you can be pretty sure you will get the banhammer. No, you don't get a second chance."[12])

Most graphics cards, network router designs, wireless network components, and other electronic devices today have such reference designs in which a company gives away open IP with or without restrictions.

The benefits of "open IP" modules are especially large when customer needs are heterogeneous and unpredictable. 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 not cover the full range of customer needs or desires. Alternatively, the firm in question can provide customers with "toolkits" or "application programming interfaces (APIs)" supporting customization and user modification.[13] Users' modifications may then range from minor changes to entirely new modules.

Most modern computer operating systems offer blocks of reusable code (i.e., modules) that are separate from the main operating system, but make it easier for programmers to develop new applications. An example can be found in the Java operating system. Its owner (Sun Microsystems, now a part of Oracle) provides class libraries that offer abstract interfaces to tasks. Programmers can use these to build their own customized applications.

The potential for external value co-creation enabled by selective openness and IP modularity leads to our first recommendation:

> ***Potential for value co-creation in the ecosystem.*** *Whenever the potential for value co-creation exists in the surrounding ecosystem, dividing the firm's own IP between relatively unprotected "open" modules and highly protected "closed" modules will be advantageous. This IP-modular strategy is most valuable when ecosystem development resources are widely distributed and customer needs are heterogeneous.*

Distributed, highly innovative ecosystems are frequently observed in industries based on information and communication technology, e.g., computers, software, telecommunications, semiconductors, and (increasingly) book and music publishing. Thus we expect this strategic rationale for IP modularity to be common in these industries.

There is also a strong link between this rationale for IP modularity and so-called platform strategies. In a platform architecture, certain components (the platform) remain fixed, while others (the complements) are permitted to vary.[14] This strategy permits the focal firm to give control over the design of complements to users or third parties. A platform strategy is a specific instance of IP modularization. MacCormack and Iansiti describe how a platform owner will often allocate proprietary IP status to the core of the platform, and more open IP status to the interfaces that allow others to build complementary components.[15]

Distributed Ownership of External IP (1-B)

In large ecosystems, the creators of IP may number in the thousands (or tens of thousands), reside in different IP jurisdictions, and offer up their intellectual property with varying terms and restrictions. In such cases, the transaction costs of utilizing external IP become an important consideration. These costs can be especially high if inconsistent legal or contractual restrictions create incompatibility between different "chunks" of external IP *within* specific modules.

This problem arose at Sun Microsystems when it moved to license key implementations of its Java programming language as open source software. Sun General Counsel Mike Dillon complained that the transition was tedious and legalistic: "*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 involvements. Where Sun didn't have the correct licenses, we had to contact the owners, one by one, and determine the rights.*"[16].

Multiplicity of IP sources is now a common occurrence in large software, computer, telecomm and IT systems. Heterogeneous and conflicting licenses give rise to IP incompatibility between different chunks of external and internal IP. A firm can reduce the impact of IP incompatibilities by designing its whole system—whether it be a code base, a product, or a process—in an IP-modular fashion. This leads to the following recommendation:

> ***Distributed ownership of external IP.*** *IP modularity with external IP is advantageous when the focal firm obtains IP with different terms and conditions 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 IP licenses.

Supporting Innovation by Suppliers, Employees and Alliance Partners

Distributed innovation is not limited to large ecosystems. It also takes place in the context of collaboration and knowledge sharing with suppliers, employees and alliance partners. In these cases, the strategic challenge is to work out what knowledge can or must be shared (and with whom) and to safeguard the firm against various types of opportunistic behavior. Two basic problems for value capture are leakage of the firm's own IP to competitors and hold-up by the external owners of IP. Our examples show how IP modularity can mitigate both these problems.

Avoiding IP Leakage (2-A)

When a firm outsources production to external suppliers or alliance partners, or when it provides critical knowledge to employees, it may be giving opportunistic agents precisely the knowledge they need to compete in its own markets. However, each participant only needs knowledge relevant to the modules it designs or builds.[17] Thus if the focal firm divides the work among different agents who do not exchange information with each other, critical IP can be protected. Each agent will know part of the puzzle, but none can reconstruct the whole. IP modularity in this case requires dividing the product or process into different task modules, and allocating those modules to separate, non-communicating agents.[18]

Recall the case of the automaker presented earlier. Purely technical considerations suggested integrating the stability control system with the braking system. However, this integration of automaker's and the supplier's IP would have exposed the automaker to the risks of leakage or hold-up. Encapsulating the stability control system as a separate module allowed the automaker to avoid these risks.

While the case of the brake system highlights the risk of IP leakage through suppliers, the production of Michelin tires illustrates the same risk vis-a-vis employees. As Julia P. Liebeskind describes: "*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. More subordinate individuals within the organization would need to collaborate in order to obtain knowledge valuable to rivals."[19]

A similar approach to process design has been applied to commercial jet engines.[20] These engines employ multiple stages, first a "cold" section at the front that employs low-pressure turbines to move large volumes of air, followed by a "hot" section where the fuel is burned. The hot section of advanced engines is highly proprietary, as the turbine blades often operate beyond the melting temperature of the metal alloys using proprietary cooling techniques and ceramic coatings. Big engine manufacturers like GE Aviation jealously guard the production of the hot section components, often spreading their manufacture across multiple locations and suppliers.

The F101 engine was developed by GE Aviation for the B-1 bomber, but when the opportunity arose to build a "10 ton" class commercial engine (with 10 tons or 20,000 lbs of thrust) in partnership with SNECMA of France, GE had to apply for an export license. The Department of State's Office of Munitions Control recommended rejection of the license on national security grounds, because of the proprietary technologies used in the hot section and because taxpayers had financed the F101. GE and SNECMA did not give up their goal of partnership, however, and eventually the issue was escalated to a summit meeting between Presidents Nixon of the U.S. and Pompidou of France in Reykjavík in 1973. The solution agreed to was for GE to build the hot section in the U.S., and export it to France where the cold section was added. This deliberate modularization protected proprietary IP related to the hot section and allayed the government's concerns about national security.[21]

Thus modularity based on the logic of "divide and rule" can strengthen IP protection. To achieve this type of IP modularity, the focal firm must design its system as a set of discrete modules with non-overlapping IP. Production of each module can then be allocated to different suppliers, alliance partners, or business units. In summary:

> ***Avoiding IP leakage.*** *When the focal firm must share its own IP with suppliers, alliance partners, and/or employees, distributing its own IP into discrete, non-overlapping modules and allocating responsibility for each module to a different firm or business unit will be advantageous.*

Avoiding Hold-up (2-B)

Rather than sharing its own IP with outsiders, a firm can opt to use external IP under license from its owner. However, short-term licenses create the risk of hold-up.[22] For example, LaMantia, Cai, MacCormack, and Rusnak describe a software company that built a platform component using code licensed-in from another software vendor.[23] Originally, the external code was spread throughout the platform, interwoven with company-generated code. In other words, the platform product was not IP-modular. When the license came up for renewal, the firm expected a steep increase in the royalties demanded. It was vulnerable to "hold-up" by the licensor.

In the months before the license expired, the firm rewrote its platform code, consciously placing its own and the external code in separate modules. Rewriting the platform was expensive, but dividing the codebase in this fashion created the possibility of substituting other code (including open source software) for the licensed-in code without disrupting the performance of the platform. The firm's switching cost was greatly reduced, and risk of hold-up largely disappeared. We summarize:

> ***Avoiding IP hold-up.*** *When the focal firm relies on external IP, it faces the risk of hold-up when IP licenses are renewed. In such cases, placing its own and external IP in separate modules will be advantageous.*

If dividing IP is advantageous for users seeking to avoid hold-up, then integrating IP into larger units should be advantageous for owners seeking to extract rents. Below we consider cases where achieving IP modularity entails combining elements using different IP. But first we discuss the effects of uncertainty.

The Effects of Uncertainty

Each of the strategic rationales for IP modularity may be present under certainty or uncertainty. With uncertainty, IP-modular designs are not immediately useful, but may give firms the ability to change the IP status of modules, hence their strategy in response to new opportunities or threats. In other words, under uncertainty IP modularity creates option value.

Options to change the IP status of proprietary modules (3-A)

Consider the example of the operating system Darwin, originated by Apple Inc.[24] Initially, Apple kept all of its operating system code proprietary. (Its operating system was based on four

bodies of open source code, but the applicable licenses permitted the code to be integrated into a proprietary product.) At this time, from an IP perspective there was no reason not to adopt an integral design. However, in 2000, Apple made part, but not all, of Darwin's source code publicly available under an open source license to take advantage of contributions by outside innovators. As we have explained earlier, IP modularity is desirable in such circumstances, but it is typically difficult and expensive to implement *ex post* (see the discussion of Sun's Java). To the extent that Apple anticipated this move when designing the system, it may have built into the program's architecture the option to selectively change the IP status of certain parts. It could have performed an "anticipatory" IP modularization by creating a design that featured a seemingly excessive degree of modularity.

In settings characterized by high levels of technological or market uncertainty, the desired boundaries of IP modules can change after the fact. As in the case of M-Systems, customers may require openness as a condition of purchase. In other cases, outsourcing some parts of the production process may become attractive. An "overly modular" initial design allows the focal firm to respond to such changes in the environment rapidly and cost-effectively. We note:

> ***Uncertainty: Options to change the IP status of proprietary modules.*** *An "overly modular" product or process architecture creates options to capture value in the future by selectively adapting the IP status of different proprietary parts of the system. Such options are more valuable in the presence of high market or technological uncertainty.*

Options to respond to claims of inadvertent infringement (3-B)

Today firms are increasingly vulnerable to IP-related threats that cannot be predicted when the product or process architecture is chosen. For complex new products in fields like electronics and software, it is often impossible to identify with certainty all patents that the product might infringe. Non-practicing entities (sometimes called "patent trolls") enforce patents against firms that—often unwittingly—use technology covered by the patent.[25] A key aspect of the trolls' strategy is surprise: they wait until the product is successful before filing suit, maximizing the value of any future damages or settlement. For example, in less than three years Forgent Networks collected more than US$100 million in licensing fees for infringement of one of its patents used in the JPEG image coding standard.[26] Practicing firms are also increasingly enforcing patents with

the aim of collecting royalties, as witnessed by the current multitude of lawsuits in the field of mobile communication.

One way to counter an infringement suit is to design around the patent by replacing the allegedly infringing component. Design-arounds are less costly when the underlying product or process is modular.[27] As long as the infringement is confined to one (or a few) modules, those modules can be redesigned or removed without compromising the functionality of the rest of the system.

The MPEG-4 compression standard for audio and video data is a case in point.[28] MPEG-4 is not a single technology, but rather a collection of methods and components. For each application, MPEG-4 is implemented by generating a specific profile that assembles the required parts. The modular structure of MPEG-4 allows firms to leave out parts with a high perceived IP uncertainty or drop parts that infringe on as yet unidentified patents. Thus:

> ***Uncertainty: Options to respond to inadvertent infringement.*** *An "overly modular" product or process architecture creates options to design around patent-infringement claims without redesigning the whole system.*

Extending control: When not to divide IP

Our analysis thus far has dealt with cases where it is advantageous for the focal firm to divide IP across modules. However, it is sometimes desirable to integrate IP that from a technical perspective could have been kept separate. In particular, we will show that combining weakly and strongly protected IP in a single module can be advantageous.

Combining Own IP in a Single Module (4-A)

In his seminal paper on profiting from innovation, David Teece observed that controlling a complementary asset can allow a firm to profit from an innovation that is only weakly protected by state-enforced IP rights.[29] Similarly, placing components with strongly and weakly protected IP in the same module can give strong protection to both. However, a precondition for this "extension of control" is that it must be technologically possible to make the underlying components inseparable — i.e., integration must be a technological option.

Inkjet printers serve well to illustrate this point. At a high level, they consist of two modules, an ink tank/cartridge and the remainder of the printer. Replacement purchases of consumable ink link revenues to the cartridge. The critical question from a product architecture

perspective is where to place the printhead. Cost considerations suggest bundling it with the printer, since printheads last longer than ink cartridges. Yet, HP and Canon chose to draw the module boundary differently, integrating the printhead with the cartridge. Their rationale was that a "minimal" cartridge, without the printhead, would be easy to imitate. It would thus be an "involuntarily open" module. The printhead, in contrast, is quite effectively protected by patents. Integrating both components in one module made the cartridges more difficult to imitate, and effectively extended the printhead's IP protection to the cartridge. This move prevented cartridges from being made by competing suppliers, and thus secured the cartridge revenues for the maker of the printer.

A second example of this principle concerns the decades-long conflict between Intel and AMD over IP related to microprocessors. When IBM selected the Intel 8088 microprocessor for its Personal Computer in 1981, it required Intel to share IP with second-source suppliers, including AMD. Intel entered into a technology exchange agreement with AMD, which allowed AMD to be a second-source for follow-on generations of the "x86" processor architecture in exchange for substantial royalties to Intel.[30]

Beginning in 1983, however, Intel began to back away from the agreement, triggering multiyear court proceedings. Intel subsequently launched several initiatives designed to move the boundary of modules to strengthen its own IP protection. First, it expanded the microprocessor's instruction set to incorporate Intel's "MMX" instructions, effectively moving functionality from software into hardware. AMD had to reverse engineer the new, more complicated microprocessors to maintain compatibility. Intel also started changing the processor sockets in every generation, so that processors could not be plugged into the previous generation's motherboards. (Intel's larger scale allowed the cost of motherboard redesign to be spread over more processor units.) Thus, similar to HP's and Canon's strategy for printers, Intel integrated proprietary IP (the MMX instruction set and its socket design) with shared IP (the processor design, which it was legally required to share with AMD). In this fashion, the strong IP protection of the proprietary components was extended to the weakly protected IP of the chip design. We summarize:

> ***Combining own IP in a single module.*** *Combining components with strong and weak IP protection within a single module extends strong protection to both, thereby increasing the focal firm's ability to capture value.*

Combining Own IP with External IP

We have said that to reduce hold-up risk, a firm should in most cases segregate its own and external IP in separate modules. However, in rare instances, it can be advantageous for a firm to integrate its own and external IP within a single module. As an example, consider the tactics of "embrace, extend, extinguish," often attributed to Microsoft. This describes the practice of embracing an open standard, adding proprietary extensions, and then using market power to extinguish the original open version. Microsoft has repeatedly been accused of employing such tactics with the Java programming language, the encryption technology Kerberos, and Web standards.[31]

In the case of the Java programming language, Sun Microsystems accused Microsoft of combining its own code with Java code in ways that limited the ability of Java programs to run on non-Windows platforms. Interpreted through the lens of IP modularity, Microsoft tried to integrate an IP-protected component (the Windows operating system) with external IP (the Java programming language) to gain control of the language and appropriate a large portion of its value. Because Java was a programming language, its commands and structure were known to all users. Java's IP was protected only by Sun's copyright, a relatively weak form of protection. The strategy was risky, however: Sun sued Microsoft for breach of contract and antitrust violations, and Microsoft eventually paid Sun $2 billion to settle the suits.[32]

In general, the strategy of "embrace, extend, extinguish" invariably harms the reputation of the focal firm and is likely to generate a backlash. For this reason, it should be pursued with caution. We summarize:

> ***Combining own IP with external IP.*** *Integrating an "open" component based on weakly protected external IP with a strongly protected proprietary component may enable the focal firm to capture part or all of the value of the open component. However, the strategy is risky and may backfire.*

A Comprehensive Framework

In the preceding sections, we looked at the tradeoffs between value creation and value capture along a number of dimensions. It is now time to pull the separate threads together into a comprehensive framework. Based on our research, we believe the following systematic procedure can help managers identify tradeoffs between value creation and value capture, achieve

appropriate levels of IP modularity, and increase their firms' ability to profit from distributed innovation and outsourcing.

To begin the analysis, managers must divide their technical systems (both products and processes) into three sectors: (A) a sector where innovation is (or needs to be) broadly distributed among numerous, diverse and heterogeneous participants (call this the "open IP" sector); (B) a sector where innovation occurs through collaboration with suppliers, employees, and alliance partners (call this the "shared IP" sector); and (3) a sector where the future locus of innovation is uncertain (call this the "uncertain" sector).

Then for each sector, managers need to ask the following structured questions.

A. For the open IP sector:
    1. How can we divide our own IP into "open" and "closed" modules to attract diverse, heterogeneous innovators?
    2. How can we simplify the acquisition of external IP to minimize transaction costs?

B. For the shared IP sector:
    1. How can we divide our own IP to grant collaborators selective access to knowledge, while minimizing IP leakage?
    2. How can we position external IP to reduce the risk of hold-up?

C. For the uncertain sector:
    1. What strategic options should we preserve with respect to the status of our own IP?
    2. What external IP is most vulnerable to claims of inadvertent infringement?

At this point, managers will have a provisional plan to achieve IP modularity. Within this plan, each module can be broadly categorized as "open", "external", or "proprietary". (Proprietary modules can be based on shared IP as long as it is protected against leakage and hold-up.)

The firm's ability to capture value rests on its proprietary modules. Managers should evaluate the IP in each proprietary module to see if it is weakly or strongly protected. Modules with weakly protected IP may be "involuntarily open," hence it may be desirable to merge two proprietary modules. (It must be technically feasible to make the components of the merged module inseparable.)

Some managers may take the further step of looking for external modules that are weakly protected and might be vulnerable to the strategy of "embrace, extend, and extinguish." However, such tactics are invariably risky: they may trigger a backlash among the co-creators of value and become the basis of lawsuits and charges of anti-competitive practice.

Contextual Factors, Extensions and Limitations

Under what conditions and in what industries will IP modularity be advantageous? We can turn this question on its head and ask: what firms do *not* need to be concerned with IP modularity? A firm or individual with one indivisible piece of IP (for example, a biotech firm with a single patent or a songwriter with a copyright) has no need to think about IP modularity. Similarly, a firm that never allows IP to cross its boundaries (in either direction) has no need to worry about IP modularity. Its internal IP can remain undivided and external IP remains beyond the pale.

Today, however, single-product "fortress firms" are increasingly rare. In contrast, any firm that wants to encourage value co-creation by opening up its IP; that participates in standards-setting organizations; shares IP with suppliers, employees or alliance partners; obtains IP under license; or is uncertain about future opportunities and threats to IP, should take note of the concept of IP modularity.

The need is most apparent in industries with complex technologies where products are made up of many different components, for example, in the computer, software, semiconductor, telecommunication, and automotive industries. In other industries, such as chemicals and pharmaceuticals, products and product-related IP are relatively indivisible. A specific drug or chemical is generally a single product protected by a single composition of matter patent, not a complex assemblage of parts. However, drugs and chemicals are manufactured via complex processes involving many separable "chunks" of IP. The concept of IP modularity can help firms organize these processes, to better access the capabilities of suppliers, alliance partners and employees, whilst still retaining control of critical IP.

For reasons of length, this paper has focused on IP modularity in products and processes. However the concept extends to organizations as well. "Chinese Walls" are an illustrative example.[33] This term refers to virtual barriers between different organizational units that prevent information exchange between these units—the advisory and trading departments of a bank, for example. These units constitute "IP modules" within the organization, and the Chinese Walls between them are module boundaries.

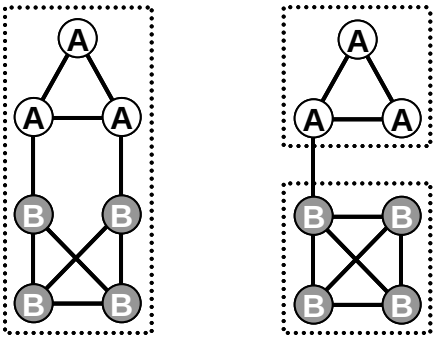
A second generalization concerns accountability. The need to clearly attribute liability in case of product failures may, in a way similar to IP, motivate a modular architecture. In safety-critical products, modules may be segregated due to differing safety certification levels. Classified data may need to be stored and processed separately from unclassified data, again suggesting a modular architecture.[34]

Finally, two caveats limit the scope of our recommendations. First, IP modularity is likely to increase the cost of design and may imply a loss of performance. Managers must evaluate these costs in relation to the benefits of potentially higher value capture. Second, in some circumstances, IP modularity may not be necessary. For example, if a company's strategy for appropriating value rests heavily on ownership of complementary assets (e.g., distribution channels), that company may not need IP protection. Similarly, trust between organizations reduces the risk of opportunistic behavior, and may reduce the need to pursue IP modularity.

Conclusion

Despite these caveats, as technologies become more complex, the reconciliation of technical architectures with IP concerns is becoming an increasingly important problem in knowledge-intensive firms. The management of IP in the form of patents and copyrights is now an essential part of the management of innovation. Yet, at the same time, broadly distributed and open innovation processes are gaining ground in many industries. IP modularity eliminates incompatibilities between IP rights in a given module, while permitting incompatibilities within the overall system. This enables a firm to reap the benefits of distributed innovation while reducing transaction costs and the costs of opportunism on the part of suppliers, partners, and employees. In this fashion, IP modularity overcomes intrinsic conflicts between distributed value creation and value appropriation in increasingly complex and fragmented technological spaces. It deserves the attention of general managers and management scholars alike.

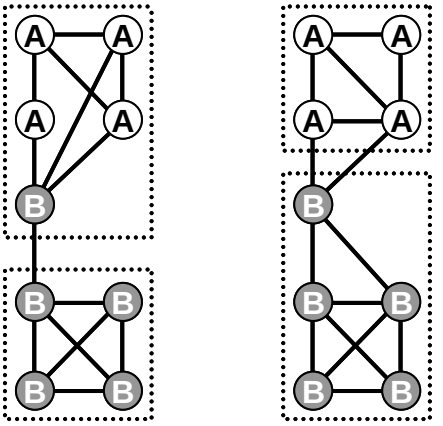
FIGURES AND TABLES



(a) (b)

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

Figure 1. Systems with integral (a) and IP-modular (b) structure.



(a) (b)

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

Figure 2. Systems with modular (a) and IP-modular (b) structure.

	Own IP		External IP	
	Rationale	*Cases*	*Rationale*	*Cases*
Enable distributed innovation in a large ecosystem	(1-A) Exploit resources in the surrounding ecosystem for distributed value creation while protecting sensitive IP for value appropriation… - … if potential for value co-creation in the ecosystem is high - … if co-creators of value are distributed, numerous, anonymous - … if the need for customizations is great and varied	M-Systems NVIDIA Java APIs	(1-B) Simplify the use of distributed external IP	Licensing Java as open source software
Manage shared IP and contractual relations	(2-A) Avoid IP Leakage by granting selective access to - suppliers - employees - alliance partners	stability system Michelin process F101 engine	(2-B) Reduce the risk of hold-up by reducing the cost of desiging around external IP	Server platform with licensed-in code
Address uncertainty	(3-A) Create options to change the IP status of proprietary modules	Apple's Darwin	(3-B) Create options to respond to claims of inadvertent infringement by owners of external IP	Patent enforcement in the cases of JPEG and MPEG
Extend control	Combine parts under weak and strong IP protection into one module to increase appropriability of the former	Inkjet Intel vs. AMD	Meld own IP with weakly protected external IP to establish control of the latter	Microsoft: embrace, extend, extinguish

Table 1. Overview of rationales for introducing IP modularity and corresponding cases

[1] See F. Kaplan, "Opening the door for the latest NAND flash in open source mobile platforms," *LinuxDevices.com*, <http://www.linuxdevices.com/articles/AT2185129745.html>, accessed May 2012; and S. Käs, "Rethinking industry practice: The emergence of openness in the embedded component industry," *Pro BUSINESS*, (2008): 140.

[2] Source: Interview with R&D manager, July 11, 2008.

[3] H.A. Simon, "The architecture of complexity," *Proceedings of the American Philosophical Society* 106, (1962).

[4] H.A. Simon, op. cit., 477; P.J. Gomes and N.R. Joglekar, "Linking modularity with problem solving and coordination effort," *Managerial and Decision Economics*, 29 (2008): 443–457; E. von Hippel, "Task partitioning: An innovation process variable," *Research Policy*, 19 (1990): 407–418; C.Y. Baldwin and K.B. Clark, op. cit.

[5] C.Y. Baldwin and K.B. Clark, *Design Rules, Volume 1: The Power of Modularity* (Cambridge, MA: MIT Press, 2000); R.N. Langlois, "The vanishing hand: The changing dynamics of industrial capitalism," *Industrial and Corporate Change*, 12 (2003): 351–385; R. Sanchez and J.T. Mahoney, "Modularity, flexibility, and knowledge management in product and organizational design," *Strategic Management Journal*, Winter Special Issue 17 (1996): 63–76.

[6] H. W. Chesbrough, *op. cit.*

[7] C.Y. Baldwin and K.B. Clark (2003), op. cit.; C.Y. Baldwin and K.B. Clark, "The architecture of participation: Does code architecture mitigate free riding in the open source development model, " *Management Science*, 52 (2006): 1116–1127; N. Franke and E. von Hippel, "Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software," *Research Policy*, 32 (2003): 1199–1215; L.B. Jeppesen, "Profiting from innovative user communities: How firms organize the production of user modifications in the computer games industry," Working paper No. 2004-03, Copenhagen Business School (2004): <http://ep.lib.cbs.dk/download/ISBN/ 8778690978.pdf>; E. von Hippel, "Perspective: User toolkits for innovation," *Journal of Product Innovation Management*, 18 (2001): 247–257; E. von Hippel and S.N. Finkelstein, "Analysis of innovation in automated clinical chemistry analyzers," *Science & Public Policy*, 6 (1979): 24–37.

[8] The GPL is a commonly used open source license that confers broad rights to use and modify to all receivers of the respective software.

[9] O. E. Williamson, "Transaction-cost economics: The governance of contractual relations," *Journal of Law and Economics*, 22 (1979): 233–261; O. E. Williamson, *The Economic Institutions of Capitalism.* (New York, NY: Free Press, 1985).

[10] J. F. Moore, *The Death of Competition: Leadership & Strategy in the Age of Business Ecosystems* (New York: HarperBusiness, 1996); M. Iansiti, Marco and R. Levien, *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*, (Boston: Harvard Business School Press, 2004); Baldwin (2012) "Organization Design for Business Ecosystems," *Journal of Organization Design*, 1 (2012):20-23; Williamson, Peter J. and Arnoud De Meyer "Ecosystem Advantage: How to Successfully Harness the Power of Partners," *California Management Review*, 55(2012):24-46.

[11] D. Harhoff, J. Henkel and E. von Hippel, "Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations," *Research Policy*, 32 (2003): 1753–1769; O. Alexy, Oliver, G. George and Ammon J.

Salter (2012) "Cui Bono: The Selective Revealing of Knowledge and its Implications for Innovative Activity," *Academy of Management Review*, preprint publication September 28, 2012.

[12] https://developer.nvidia.com/code-of-conduct, accessed November 2012.

[13] E. von Hippel, "Perspective: User toolkits for innovation," *Journal of Product Innovation Management*, 18 (2001): 247–257.

[14] E.g. A. Gawer and M.A. Cusumano, *Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation.* (Boston, MA: Harvard Business School Press, 2002); C.Y. Baldwin and C.J. Woodard, "The architecture of platforms: A unified view," *In Platforms, Markets, and Innovation*, A Gawer (ed). Edward Elgar: Cheltenham, UK (2009): 19-44.

[15] A. MacCormack and M. Iansiti, "Intellectual property, architecture, and the management of technological transitions: Evidence from Microsoft Corporation," *Journal of Product Innovation Management*, 26 (2009): 248–263. See also J. Waltl, J. Henkel and C.Y. Baldwin, "IP modularity in software ecosystems—how SugarCRM's IP and business model shape its product architecture," to appear in: *Proceedings of the 3rd International Conference on Software Business* (2012).

[16] See C. Preimesberger, "Sun Pours Out Java Cup," *Eweek.com*, < http://www.eweek.com/c/a/Application-Development/Sun-Pours-Out-Java-Cup/>, accessed May 2012.

[17] C.Y. Baldwin (2008) op. cit.; A. Tiwana, "Does interfirm modularity complement ignorance? A field study of software outsourcing alliances," *Strategic Management Journal*, 29 (2008a): 1241-1252; A. Tiwana, "Does technological modularity substitute for control? A study of alliance performance in software outsourcing," *Strategic Management Journal*, 29 (2008b): 769-780.

[18] For a formal model of this strategy, see C.Y. Baldwin and J. Henkel, "The impact of modularity on intellectual property and value appropriation," *Harvard Business School Working Paper*, 12-040 (2012).

[19] J. P. Liebeskind, "Keeping Organizational Secrets: Protective Institutional Mechanisms and their Costs," *Industrial and Corporate Change*, 6(3) (1997): 623-663.

[20] "CFM56: Engines of Change," *Flight International*, 19 (25 May 1999): 4-15.

[21] This engine became the CFM56 family of commercial engines, powering the Boeing 737 and Airbus A320. It is the most popular commercial engine in the world.

[22] O. E. Williamson (1979, 1985) op. cit.

[23] M.J. LaMantia, Y. Cai, A.D. MacCormack and J. Rusnak, "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*, (2008): 83–92.

[24] See Source, "Darwin (operating system)," *wikipedia.org*, <http://en.wikipedia.org/wiki/Darwin_(operating_system)>, accessed May 2012.

[25] J. M. Golden, "Patent trolls and patent remedies," *Texas Law Review*, 85 (2007): 2111–2162; J. Henkel and M. Reitzig, "Patent sharks and the sustainability of value destruction strategies," Working paper (2007): <http://papers.ssrn.com/sol3/papers.cfm?abstract_id=985602>; M. A. Lemley and C. Shapiro, "Patent holdup and royalty stacking," *Texas Law Review*, 85 (2007): 1991–2049; M. Reitzig, J. Henkel and C. H. Heath, "On sharks, trolls,

and their patent prey – Unrealistic damage awards and firms' strategies of 'being infringed'," *Research Policy*, 36 (2007): 134–154.

[26] M. Reitzig, J. Henkel and C. H. Heath, op. cit.

[27] C.Y. Baldwin and J. Henkel (2012) op. cit.

[28] Based on an interview with Klaus Diepold, member of the MPEG-4 standardization committee, July 2007; J. Henkel and M. Reitzig, "Patent sharks," *Harvard Business Review*, 86 (2008): 129–133; See "Overview of the MPEG-4 Standard," *mpeg.chiariglione.org*, <http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm>, accessed May 2012.

[29] D.J. Teece, "Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy," *Research Policy*, 15 (1986): 285–305.

[30] "Advanced Micro Devices Inc. v. Intel Corp." Cal. 1994, law.justia.com (No. S033874, 885 P.2d 994, 997–98).

[31] "Deadly embrace," The Economist, March 30 (2000), http://www.economist.com/node/298112?Story_ID=298112; "Opera files antitrust complaint with the EU," Opera Press Releases, December 13 (2007), http://www.opera.com/press/releases/2007/12/13/. accessed May 23, 2012.

[32] http://www.thefreelibrary.com/-a019837687; http://news.cnet.com/2100-1001-855696.html; http://www.theserverside.com/news/thread.tss?thread_id=12379; http://www.microsoft.com/en-us/news/press/2004/apr04/04-02sunagreementpr.aspx; Accessed November 26, 2012.

[33] H. McVea, Financial conglomerates and the Chinese wall: Regulating conflicts of interest (New York, NY: Oxford University Press, 1993).

[34] B. Ames, "Real-time software goes modular," *Military & Aerospace Electronics*, 14/9 (September 2003): 24–29.