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

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

August 1, 2012

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

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.

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, Lorenzo Massa, 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, European Patent Office, 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

Distributed value creation through innovative product and process designs is a key reality of the modern world. Firms increasingly practice open innovation, license technology out and in, outsource development and production, and enable users and downstream firms to innovate on their products.[1] However, while distributed value creation can boost the overall value created, it may create serious challenges for capturing value. In order to draw in external contributors, an innovator often grants them access to knowledge covering a product or process.[2] The innovator thus gives up some of its intellectual property (IP), by waiving legal exclusion rights or by revealing formerly exclusive knowledge. But as a result, contributors may appropriate a large share of the jointly created value. In turn, integrating external contributions entails the risk of becoming dependent on outside owners of IP and thus exposed to holdup.

How can firms exploit the advantages of distributed value creation while avoiding concomitant risks for value capture? Our paper addresses this tension by proposing the concept of "IP modularity." We argue that, by managing a system's modular structure in conjunction with its IP, firms can reconcile opportunities for distributed innovation with their need to capture value. That is, the modular architecture of a product or process should not be determined solely by technical and organizational criteria, but also by considerations of IP and value capture. A central element in our argument is that the structure of a technical system is in large measure under the control of the system's architects. Therefore, the technical architecture of a product or process and a firm's strategy for creating and claiming value are inevitably intertwined. In particular, in a modular system, a firm must simultaneously decide on the technical boundaries of the 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 for itself.

Consider the example of M-Systems, a subsidiary of the leading seller of flash memory cards, SanDisk.[3] In 2005, the open source operating system Linux was 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 to simplify development of the host systems. However, M-Systems wanted to

protect sensitive IP which was part of the driver. As a solution, M-Systems introduced what we call an IP-modular architecture: They split the formerly one-module driver into two modules, based solely on IP considerations: the flash management code, and the remaining "thin" driver. They placed the former on the physical memory device itself, thus keeping it protected via copyright and secrecy, and published the driver as open source software. Redesigning the technical modules to be congruent with the desired structure of IP allowed M-Systems to reconcile customer demands for openness with its own value appropriation: they kept control of the "essential module"[4], the flash management software, and were able to open up a "complementary module," the driver, which exhibited the largest potential for distributed value creation.

To achieve IP modularity, in which the technical and IP boundaries of modules are congruent, system architects must distinguish between the focal firm's own IP and external IP. The case of M-Systems is about the firm's own IP, since M-Systems owned the IP to the entire original driver and then opened up a part of it to outsiders. The following case from the automotive industry illustrates how IP modularity works for external IP.

An automaker sourced a braking system from a supplier, which sold it to several customers.[5] Within the automobile, the braking system was both a technical and IP module, since it was a physically distinct component and all its IP was owned by the supplier. Then, however, the automaker developed a stability control system which relied heavily on features of the braking system, in particular on its ABS (antilock braking system). From a strictly 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 rely on IP from both the supplier and the automaker. In the case of an integrated design, the supplier must either sell the integrated system to all its customers, or sell two distinct braking systems—one to the automaker who provided the IP, and the other to everyone else. The first option was unacceptable to the automaker, which considered its proprietary stability system to be an important point of product differentiation. The second would have required the brake supplier to maintain two distinct products, forgoing economies of scale, and would still have posed risks to the automaker of inadvertent loss 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, to avoid a commingling of external IP with its own IP, the automaker adopted an IP-modular design for its braking system. In the words of an executive 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."

In this paper, we explain how modularity affects value capture: Under what conditions, and how, it can support an innovator in profiting from innovation, and when it should be avoided. To this end, we define modularity and show that it enables innovation outside the walls of the focal firm. If not carefully handled, the presence of external innovators creates problems of value appropriation as discussed by Teece.[6] We then introduce the concept of IP modularity, and explore this concept using case studies from various industries. We first address IP modularity related to the focal firm's own IP; then, related to external IP; and finally, IP modularity under uncertainty. We derive strategic recommendations for each category.

Modularity and Profiting from Innovation

For decentralized innovation and value creation to occur in large systems, it is necessary for the parts to be independent so that an innovation in one part of the system will not require changes in many other parts.[7] This condition is captured in the concept of modularity. Modularity reduces the cognitive burden of complexity, and allows tasks to be partitioned and worked on in parallel.[8] With task partitioning, a productive division of labor may arise between individuals within one firm or across firm boundaries.[9] The existence of task modules with well-defined interfaces reduces transaction costs, thus supporting outsourcing.[10] Finally, changes within modules and recombinations of modules are easy in a modular architecture.

Importantly in our context, modularity in the design of products and processes facilitates the division of labor in the innovation process between firms[11]—an approach aptly labeled "open innovation" by Henry Chesbrough.[12] In a modular system, innovations can more easily be introduced by firms other than the creator of the modular architecture—firms in related markets, entrepreneurial startups, or even users.[13]

However, the fact that modularity enables innovation "at a distance" from the focal firm creates implicit tension between value creation and value appropriation. By granting external contributors access to knowledge covering a product or process, 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.[14]

Also the IBM PC had a highly modular architecture, and in the end most of the value was captured by Intel and Microsoft, not IBM.[15] While IBM outsourced most parts of the PC, it kept an essential piece of software proprietary, the BIOS (Basic Input Output System). Had the firm managed to maintain exclusivity of this essential module, history might have been different (in particular if the second source arrangement for CPUs with AMD had worked—more about this later). Yet, in 1983 IBM lost this exclusivity when Phoenix Technologies replicated the BIOS, enabling the creation of legal "clones" fully compatible with the PC and sold at prices far below IBM's.[16]

Protection of the essential module—in the case of the IBM PC, the BIOS—links the topic of modularity to the literature on profiting from innovation, which originated with David Teece's seminal work.[17] According to Teece, the first key driver of a firm's ability to profit from innovation is the difficulty other firms face in imitating the focal innovation. This "appropriability regime" in turn is determined by the effectiveness of legal protection mechanisms (patents, copyright, and contracts), secrecy, and other mechanisms such as product complexity. For the PC BIOS, copyright and secrecy were the relevant mechanisms, but they proved ineffective against independent imitation. Relying on the same mechanisms to protect its own essential module—the flash management software—M-System was more successful.

The second key driver is the control of complementary assets, which are defined as capabilities or assets required for the successful commercialization of an innovation. Typical examples of complementary assets are distribution channels, a brand name, or competitive manufacturing. Ownership of a complementary asset—rather than contracting for it—is recommended if the focal innovation's appropriability regime is weak or the complementary asset is not available in competitive supply.

One can draw a useful analogy between the concept of complementary assets and IP modularity. Just as control of a critical complementary asset may compensate for a weak appropriability regime, placing two components in one module may allow the strong IP protection of one to compensate for weak IP protection of the other. The precondition for such extension of protection is that it must be technologically possible to tightly link the two components— i.e., integration must be a technological option.

Inkjet printers serve well to illustrate this point. On the highest level, they consist of two modules, an ink tank/cartridge and the remainder of the printer. Replacement purchases of

consumable ink links 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 have longer useful lifetimes 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 much 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 revenues linked to the cartridge for the makers of the printer. Integrating the printhead with the simple cartridge created a proprietary IP module that was strongly protected because it was difficult to imitate legally. Note that, in this case, IP considerations suggested merging two modules, while in the case of M-Systems they suggested splitting a module.

Introducing IP Modularity

To elucidate the concept of "IP modularity," we must introduce the notions of "IP status" and of "IP incompatibility."

A module's IP status describes the legal rights to and *de facto* accessibility of knowledge embodied in the module. State-granted property rights and characteristics of the technology define the "maximum" of protection that the module owner can establish. From this basis, he can selectively waive or transfer some of his rights and grant access to knowledge, where "selectively" refers both to the level of information sharing and to the recipients. For example, the module owner may keep knowledge related to the module secret, make it available to select parties, disclose it publicly while asserting patents on it, or disclose it freely to all interested parties. Recall the case of M-Systems described above. 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 maintained this status, while the thin driver had the IP status "source code open, far-reaching rights granted to others according to the GPL.[18]"

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. 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.[19] Interweaving the commercially licensed code and the open source code then leads to conflicting legal requirements. Thus, the two licensed-in blocks of code exhibit "IP incompatibility."

The above example involved external IP, that is, IP obtained under license from others. But there can also be incompatibility in the firm's own IP. 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. In the cartridge/printhead example, in contrast, the two components were IP compatible because the focal firm wished to make both proprietary.

With these definitions, we are now in a position to introduce "IP modularity" formally. We define an "IP-modular" system architecture as one in which the technical boundaries of modules are co-extensive with IP status. That is, if IP incompatibilities are present within the system, then *they exist only between, but not within modules*. For the braking and stability system described in the Introduction, this condition would not have been fulfilled had it been designed as one module: the automaker's stability system had the intended status "own IP, secret and copyrighted (and likely patented)," while the braking system had the status "owned by the supplier."

For illustration, let A and B in Figure 1 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: a module boundary has been introduced that separates elements under different IP status. 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* ----

We need to point out two trade-offs. 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 free. Strict product or process modularity with well-defined boundaries and

interfaces is costly to achieve and maintain. Similarly, segregating different chunks of knowledge according to their IP status is more expensive than pursuing the lowest-cost technical solution. Thus an IP-modular system may not be optimal from an engineering standpoint. Both for M-Systems' driver and for the automakers stability system, a one-module solution would likely have been technically superior. For the cartridge/printhead system, a two-module solution would have been less costly since the printhead would not be replaced with each change of ink. Thus the optimal modular structure for capturing value is not necessarily the same as the optimal structure for creating value. There is a tension between value creation and value capture, and a need to trade off benefits and costs.

IP Modularity with Own IP

We now illustrate and enrich the concepts introduced above with further case examples from practice, summarized in Table 1. We start with "own IP," that is, circumstances in which the focal innovator has to align the modular architecture of its system with the IP status that it desires to convey to the various parts of the system. We will then discuss the management of "external IP," that is, how to incorporate IP owned by others within the system.

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

Resources in the Surrounding Ecosystem

We described M-System's redesign of its flash memory product in the Introduction. Importantly for our argument, the redesign 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. However, by simplifying the integration of its flash memory into mobile devices, the firm facilitated external value co-creation by device manufacturers and open source developers. To summarize:

> ***Potential for value co-creation in the ecosystem.*** *Whenever the potential for value co-creation exists in the surrounding ecosystem, distributing own IP into relatively unprotected "open" modules and highly protected "closed" modules will be advantageous.*

The benefits of IP modularity also depend on the extent to which the pertinent resources 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 resources. 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 a "reference design" which includes a sample hardware layout and the software that is needed, such as the software code that is required for the chip to function, generally called "drivers," into an open module. They split their own IP: they keep the majority proprietary, and publish open modules that include all the designs, electronic files, and test programs that a card manufacturer might need. Most graphics cards, most router designs, wireless network components, and other electronic devices today are such reference designs that the company gives away as open IP. To summarize:

> ***Distributed co-creators.*** *The more distributed, numerous, and anonymous the co-creators of value, the more advantageous it is to establish an IP modular system made up of unprotected open modules and protected closed modules.*

It is worth noting the link between this principle and so-called "platform strategies." In a "platform architecture," certain components (the platform) remain fixed, while others (the complements) are permitted to vary.[20] 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 in the sense that, as MacCormack and Iansiti 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 interfaces that allows others to link complementary components to it.[21]

IP modularity is especially important 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 still may not cover the full range of customer needs or desires. Similarly, granting each user individual rights for the modifications he or she desires would incur high transaction costs.

Another way to address heterogeneous and unpredictable user needs is again to split the system and the related own IP into open and closed modules. This is effectively what firms do when they provide customers with "toolkits" or "application programming interfaces (APIs)" supporting customization and user modification.[22] Users' modifications may then range from minor changes to entirely new modules. Most modern computer operating systems offer modules that are separate from the main operating system that are blocks of reusable code to make it easier for programmers to develop new applications. These modules usually perform functions for well-known tasks, and they are dynamically loaded at runtime, when the finished program runs. Another example is in the Java operating system. Its owner (Sun Microsystems, now a part of Oracle) provides class libraries that offer an abstract interface to tasks. Programmers can use these as they build their own custom modules. In summary:

> ***Customization.*** *The greater and more varied the need for customer adaptations, the more advantageous it is to establish an IP modular system made up of unprotected open modules and protected closed modules.*

Avoiding IP Leakage vis a vis Suppliers, Employees and Alliance Partners

Value co-creation need not refer only to innovation; distributed production is also an important element of today's economy.[23] When a firm outsources a large portion of its production to external suppliers or alliance partners, or when it provides critical knowledge to employees, it must be concerned that it is giving opportunistic agents precisely the knowledge they need to compete in its own markets. However, in a distributed production system, each participant only needs knowledge relevant to the modules it designs or builds.[24] 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 breaking up the product design or production process into different task modules, and allocating those modules to separate, non-communicating agents.[25]

Recall the case of the automaker presented earlier. Purely technical considerations had suggested integrating the stability control system with the braking system. However, this integration of the automaker's own IP with the supplier's IP would have exposed the former to the risk of leakage, since the integrated module would have been in the hands of the supplier. Encapsulating the stability system as a separate module allowed the automaker to avoid this risk.

While the case of the brake system is about IP-modularizing a product to avoid leakage of IP, the production of Michelin tires illustrates how the architecture of a process may be adapted to that same purpose. 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."*[26]

A similar approach to process has been applied to commercial jet engines.[27] These engines employ multiple stages, a "cold" section which is at the front of the engine that employs low pressure turbines that move large volumes of air, intermediate stages, and a high pressure "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 that they are constructed from, so they employ proprietary cooling techniques and ceramic coatings. Major engine manufacturers may source components for the low pressure sections, or even some of the high pressure sections, and maintenance, repair, and overhaul (MRO) operations can tear down and reassemble engines. But big engine manufacturers like GE Aviation jealously guard the production of the hot section components, even spreading their manufacture across multiple locations.

The F101 engine was developed by GE Aviation for the B-1 bomber, but when the opportunity arose to build a "10 ton" class high bypass 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 hot section, and also because the F101 had been built with U.S. taxpayer funds. 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 core hot section in the U.S., and export it to France where the low pressure section was added.

This deliberate modularization of process-related IP protected the appropriability of the hot section.[28]

Modularity based on the logic of "divide and rule" can strengthen IP protection in otherwise weak appropriability regimes. However, in such cases, the principles for designing an IP-modular system are different from those used to attract value-co-creation or address heterogeneous customer needs. As before, all IP within a given module should have the same IP status. But instead of creating purely "open" and "closed" modules, each module now has the IP status "open to supplier, employee group, or alliance partner X, closed to all others." In designing its products and related processes, the focal firm must tailor a set of discrete modules with *divisible tasks* and *non-overlapping IP*. Production of each of these modules can then be allocated to different suppliers, alliance partners, or business units within the firm. In summary:

> ***Avoiding IP leakage.*** *When the focal firm must rely on opportunistic suppliers, alliance partners, and/or employees and its IP is not strongly appropriable, distributing own IP into discrete, non-overlapping modules with little or no stand-alone value and allocating responsibility for each module to a different firm or business unit will be advantageous.*

Extending control: When not to divide own IP

Our analysis thus far has dealt with circumstances in which it is advantageous for the focal firm to divide its own IP into "open" and "closed" modules or a set of modules open to distinct recipients. But as the case of inkjet printers clearly showed, it is sometimes desirable to integrate components and IP that might otherwise have been kept separate. Thus achieving IP modularity might require combining elemental IP as well as dividing it. In particular, we argued, melding a component that has good revenue potential but weak appropriability with another that has less revenue potential but strong appropriability can be advantageous for value capture.

As a second example of this principle, consider the decade-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 that Intel enable a second-source supplier because it did not want to be dependent on a sole source. Intel entered into a ten-year technology exchange agreement with Advanced Micro Devices (AMD). The companies amended their contract in 1984

to allow AMD to be a second-source for following generations of the "x86" processor architecture in exchange for substantial royalties to Intel.[29]

Beginning in 1983, responding to the superior performance of AMD's processors, Intel began a strategy to wall off its IP, triggering multiyear court proceedings. When Intel eventually had to settle in 1995, it launched several initiatives to effectively move the boundary of its IP so that it could secure stronger appropriability. In particular, it changed the microprocessor's instruction set to incorporate Intel's "MMX" instructions, effectively moving functionality from software into hardware. AMD then had to reverse engineer the new microprocessors to maintain compatibility. Intel also started changing the sockets that each generation of processor plugged into, so that AMD would not be able to share the infrastructure of complements provided by motherboard manufacturers. Thus, similar to HP's and Canon's strategy of integrating the printhead with the cartridge, Intel integrated proprietary elements (the MMX instruction set and its socket design) with the "relatively open" component (the processor itself, which under the terms of the settlement it was required to share with AMD). In this fashion, the IP protection of the proprietary components was extended to the more open components of the integrated chip. Through this strategy, with its "Intel Inside" marketing campaign, in the late 1990s, Intel was able to achieve a position of market dominance in microprocessors for PCs. We summarize:

> ***When not to divide own IP.*** *Merging elements of own IP under weak appropriability with other elements under strong appropriability into one module may allow extending the strong appropriability to the former, thus securing the former's revenue potential.*

IP Modularity with External IP

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

Distributed Ownership of External IP

When dealing with external IP, transaction costs may be high simply because the upstream co-creators of value are distributed and numerous. If in addition, inconsistent legal or contractual restrictions create incompatibility between different "chunks" of incoming IP *within* modules (hence, if the architecture is not IP modular), transaction costs can be particularly high. 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: "*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.*"[30].

In Sun's case, the problem was caused not by the opportunism of a single supplier, but by the sheer multitude and variety of IP sources. Such multiplicity of sources is now a common occurrence in large software systems. It may give rise to incompatibilities between different chunks of incoming IP and between the firm's 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. In summary:

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

Hold-up Risk

Integration of own with external IP may be particularly problematic because of the risk of holdup. LaMantia, Cai, MacCormack, and Rusnak describe a software company whose entire product family depended on a central platform component.[31] This platform contained both the company's own code and code licensed-in from another software vendor. Furthermore, the platform was designed in an integral fashion, with the licensed-in code spread throughout the

codebase. Thus, it was not IP-modular. 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, this fact would have exposed the firm to "holdup" on the part of the licensor.[32]

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, nor did it 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. We summarize:

> ***Holdup risk.*** *IP modularity with external IP is advantageous when the focal firm faces the risk of holdup from IP suppliers. The module boundaries should go between the firm's own IP and the external IP.*

Extending Control: When to Avoid Modularity with External IP

As in the case of a firm's own IP, it may be preferable to avoid modular designs using external IP and instead integrate two or more components into one module. The rationale is similar in both cases: by tightly integrating a "relatively open" component with a component under strong appropriability, the latter also affords protection to the open component. If the open component is external, then such integration may allow the focal firm to establish control of it.

As an example, consider the tactics of "embrace, extend, extinguish." This term describes the practice to embrace an open standard, add proprietary extensions, and then use market power to extinguish the original, open version. Microsoft has repeatedly been accused of employing this tactics, for example with the Java programming language, the encryption technology Kerberos, and Web standards.[33] Interpreted in the context of IP modularity, the firm tried to integrate an IP-protected component (its proprietary extension) with an open, external component (the open standard) into one "module" in order to gain control of the external IP under consideration. We summarize this approach in general terms as follows:

> ***Establishing control of external open IP through integration with own IP.*** *Integrating an "open" component under external IP with an own component under strong appropriability into one module may allow to establish control of the open component.*

The effect of uncertainty

Uncertainty may compound the various considerations that played a role so far. By creating IP-modular designs, firms gain the ability to change their IP strategy in specific parts of the system in response to new value-capture opportunities or to threat of expropriation. Thus, IP modularity creates option value.

Uncertainty and Own IP

Consider the example of the operating system Darwin, originated by Apple Inc.[34] 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.[35] As we have explained earlier, 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, it might have built into the program's architecture the option of selectively changing the IP status of certain parts. That is, it may have performed an "anticipatory" IP modularization by creating a design that 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 that make IP-modularity desirable. As in the case of M-Systems, customers may require openness of some of the product's IP as a condition of purchase. Alternatively, outsourcing of production may become attractive for cost reasons. An "overly modular" initial design allows the focal firm to respond to such changes rapidly and cost-effectively. We note:

> ***Uncertainty and Own 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 for IP modularity of own IP are more valuable in the presence of high market or technological uncertainty.*

Inadvertent infringement

When the focal firm uses external IP that is subject to uncertainty, IP modularity again creates option value. We have already explained that IP modularity is a way to counter a known threat of holdup. Today 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. Non-practicing entities (often referred to as "patent "trolls") enforce patents, often acquired ones, against firms that—typically unwittingly—use the technology covered by the patent.[36] A key aspect of the trolls' strategy is surprise: they wait until the IP-related product or process is successful before filing suit, thus maximizing the value of any future damages or settlement. For example, Forgent Networks collected, in less than three years, more than US$100 million in licensing fees after suing users of the JPEG standard for infringement of one of its patents.[37]

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 are less costly when the underlying product or process is modular.[38] 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.

The MPEG-4 compression standard for audio and video data is a case in point.[39] 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. The modular structure of MPEG-4 allows firms, in defining a "profile," to leave out parts with a high perceived IP uncertainty. Furthermore, when components 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 recommendation:

> ***Inadvertent infringement.*** *A product or process architecture characterized by IP modularity is a partial defense against unanticipated external IP claims, because it gives the firm options to "design around" the claims without redesigning the whole system.*

Conclusions

In order to optimize value capture from a new product or process, an innovator must manage the artifact's IP and its modular structure in conjunction. Each module's IP status needs to be defined carefully and its boundaries must be placed accordingly. Based on theoretical

arguments and case studies from various industries we have derived the following insights. With own IP, IP modularity helps to reconcile distributed value creation with value capture, and to avoid IP leakage to suppliers and employees. With external IP, an IP-modular architecture reduces holdup risk and other transaction costs of licensing, and may allow a firm to establish control over external, originally open IP. Finally, under uncertainty, "anticipatory" IP modularity creates option value: it allows a firm to better exploit upcoming opportunities for distributed value creation and to counter threats from inadvertent IP infringement.

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 on IP modularity in the products firms sell and the processes they use. However, the concept extends to organizations. "Chinese Walls" are an illustrative example.[40] 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, in our notion, constitute "IP modules" within the organization, and the Chinese Walls between them are module boundaries.

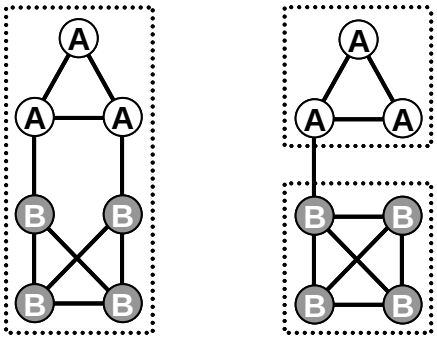
A second generalization concerns the rights and obligations under consideration. While we focused on IP as an important source of such rights and obligations, the latter may also have other legal or contractual origins. For example, the need to clearly attribute liability in case of product failures 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. Classified data may need to be stored and processed separately from unclassified data, again suggesting a modular architecture.[41]

Before closing the paper, we must emphasize two caveats. First, IP 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 benefits of IP modularity in terms of higher value capture. Second, 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, that company may not need IP protection. Similarly, trust between organizations reduces the risk of opportunistic behavior, and thus may reduce the need to pursue IP modularity.

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 "have its cake and eat it too," that is, it can reap the benefits of an open architecture while at the same time reducing the costs of opportunism on the part of suppliers, complementors and employees. In this fashion, IP modularity overcomes intrinsic conflicts between distributed value creation on the one hand, and value appropriation on the other hand.

The reconciliation of technical architecture with IP assignments is increasingly important. The management of formal IP in the form of patents and copyrights has become an essential 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. 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.

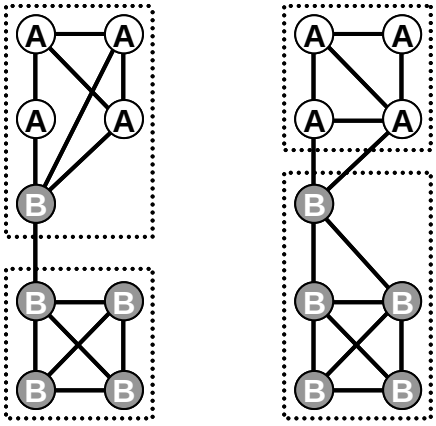
FIGURES AND TABLES



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

(a) (b)

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



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

(a) (b)

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

Own IP		**External IP**	
Rationale	*Cases*	*Rationale*	*Cases*
Resources in the surrounding ecosystem - Potential for value co-creation in the ecosystem - Distributed co-creators - Customization	 M-Systems NVIDIA Java, APIs	Distributed ownership of external IP	Licensing Java as open source software
Avoiding IP Leakage by IP Modularity vis a vis… - suppliers - employees - alliance partners	 braking system Michelin F101 engine	Hold-up risk	Platform with licensed-in code
Extending control: When not to divide own IP	Inkjet Intel vs. AMD	Extending control: When to avoid modularity with external IP	Microsoft: embrace, extend, extinguish
Uncertainty and own IP	Apple's Darwin	Uncertainty and external IP: Inadvertent infringement	Patent enforcement: JPEG, MPEG

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

[1] H. W. Chesbrough, *Open innovation. The new imperative for creating and profiting from technology.* (Boston: Harvard Business School Press, 2003).

[2] Such access may even be granted with little or no restrictions imposed, an approach that has been termed "free revealing," 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.

[3] 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.

[4] We define as "essential" modules those that are indispensable for the functioning of the whole system. This notion is related to that of "bottleneck assets" in the value chain as used by G. Pisano and D. Teece in "How to Capture Value from Innovation: Shaping intellectual property and industry architecture," *California Management Review*, 50 (2007): 287–296.

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

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

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

[8] 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.

[9] S. Sanderson and M. Uzumeri, "Managing product families: The case of the Sony Walkman," *Research Policy*, 24 (1995): 761–782; C.Y. Baldwin and K.B. Clark (2003), op. cit.; R.N. Langlois and P.L. Robertson, "Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries," *Research Policy*, 21 (1992): 297–313; R. Sanchez, "Strategic flexibility in product competition," *Strategic Management Journal*, Summer Special Issue 16 (1995): 135–159; N. Staudenmayer, M. Tripsas and C.L. Tucci, "Interfirm modularity and its implications for product development," *Journal of Product Innovation Management*, 22 (2005): 303–321.

[10] C.Y. Baldwin, "Where do transactions come from? Modularity, transactions, and the boundaries of firms," *Industrial and Corporate Change*, 17 (2008): 155–195; C.H. Fine, *Clockspeed: Winning industry control in the age of temporary advantage.* (Cambridge, MA: Da Capo Press, 1998); G. Hoetker, "Do modular products lead to modular organizations," *Strategic Management Journal*, 27 (2006): 501–518; M.G. Jacobides, "Industry change through vertical disintegration: How and why markets emerged in mortgage banking," *Academy of Management*

Journal, 48 (2005): 465–498; T. Sturgeon, "Modular production networks: A new model of industrial organization," *Industrial and Corporate Change*, 11 (2002): 451–496.

[11] C.Y. Baldwin and K.B. Clark (2003), op. cit.; 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.

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

[13] 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.

[14] C.Y. Baldwin (2008) op. cit.; C.Y. Baldwin and K.B. Clark (2003), op. cit.

[15] C.H. Ferguson and C.R. Morris, *Computer wars: How the West can win in a post-IBM world.* (New York: Times Books, 1993).

[16] R.X. Cringely, *Accidental Empires* (New York: Harper Business edition, 1992); C.H. Ferguson and C.R. Morris, op. cit.

[17] D.J. Teece (1986) op. cit.

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

[19] Free Software Foundation (1991), "The GNU General Public License (GPL) – Version 2", <http://www.opensource.org/licenses/gpl-2.0.php>, accessed May, 2012.

[20] 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.

[21] 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).

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

[23] T. Sturgeon, "Modular production networks: A new model of industrial organization," *Industrial and Corporate Change*, 11 (2002): 451–496.

[24] 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.

[25] 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).

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

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

[28] 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.

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

[30] 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.

[31] 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.

[32] 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).

[33] "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/. Both accessed May 23, 2012.

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

[35] 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 allowed keeping derived work proprietary, which is what Apple initially did.

[36] 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.

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

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

[39] 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.

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

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