Modularity and Product Innovation in Digital Markets

MARC BOURREAU

ENST, Paris, France and CREST-LEI

PINAR DOĞAN *

John F. Kennedy School of Government, Harvard University

MATTHIEU MANANT

ENST, Paris, France

Abstract

Most digital goods have a modular design; that is, they consist of complementary and distinct building blocks, called modules. Modular product design, in contrast to integrated (or integral) design, enables alteration of a specific module that is usually assigned for a specific function without necessarily requiring an entire redesign of the product. This feature facilitates product innovation. The possibility of having common modules embedded in a range of products is likely to affect firms' product innovation strategies and post-innovation competition both in traditional and digital markets. In this paper, we explore such effects with a focus on digital markets.

1 Introduction

The concept of modularity is defined in a wide range of fields: construction, art, software design, etc.[1] Modularity in products implies that products consist of distinct, relatively independent building blocks,[2] among which the interactions are ruled by standardized interfaces. Modular design in products allows the pairing of common units with different modules to create product variants.[3] The possibility of having common modules embedded in a range of products is likely to affect firms' product innovation strategies and post-

* Contact Author. Kennedy School of Government, Harvard University, 79 JFK Street, Cambridge, MA 02138-5801, United-States. Email: pinar_dogan@ksg.harvard.edu We would like to thank the guest editors, Eric Brousseau and Thierry Pénard, and two anonymous referees for their helpful comments on the earlier drafts of this paper. We also wish to thank Robert Mitchell for his editorial assistance.

[1] See Gershenson *et al* (2003) for an extensive study on definitions of modularity.

[2] We call these blocks modules or components, interchangeably.

[3] See Huang (2000) for an overview of modular product development.

innovation competition, both in traditional and digital markets. In this paper, we explore such likely effects with a focus on digital markets.[4]

Most products involve some degree of modularity. Common examples are personal computers, consumer electronics, software, automobiles, and aircraft.[5] Modular product design, in contrast to integrated (or integral) design, enables alteration of a specific module that is usually assigned for a specific function without necessarily requiring an entire redesign of the product.[6] This also means that a variety of products, either produced by a multiproduct firm or by different firms, can use a common set of product components that serve for specific functions.

Component-sharing within firms is rather natural. Firms often use common components to produce a variety of products with cost-side, demand-side, or strategic motives. For example, in the automobile industry, the same car engine is typically used in a range of car models produced by the same manufacturer (the AJ25 engine developed and manufactured by Ford is used in both the Ford Mondeo and the Jaguar X-type). Many software products come with different versions, and yet share a body of common software codes: MacKichan sells three distinct programs (Scientific Workplace, Scientific Word, and Scientific Notebook) that share common components.

Components can also be shared between different firms. For example, a diesel engine, DW10, which was jointly developed and manufactured by the PSA group (Peugeot and Citroën) and Ford was used in variety of PSA passenger models (Citroën Xsara and Xantia, Peugeot 306 and 406), as well as Ford models (for example, Ford Focus and C-Max, Mazda 5).[7] Joint product development or cooperation in product R&D, for specific product components is feasible because the building-block nature of modular product design makes it possible to partition product development tasks among independent, specialized units.[8] The interaction between different product modules is ruled by standard product interfaces; proper definition of standards and rules of interaction is crucial.[9] Open-source software is an example from the digital markets. Software companies can cooperate to design some components of software through an open-source project and develop other components of the software independently. Component sharing between firms is not limited to joint product development – it can also take place through licensing agreements.

Developing product variants by component-sharing, either within or between firms, is likely to affect both the cost structure and the degree of differentiation between the product variants. Development of two products that share many common components can be less costly compared to those that share fewer components, but the degree of differentiation is also likely to be lower, which may affect competition between the products. Firms, therefore, may take both cost and differentiation considerations into account when deciding whether to share components.

---

[4] Most of our discussions, however, also do apply to physical modular products.

[5] See Sanchez and Mahoney (1996, p.67) for a list of modular products.

[6] In this paper, we consider modular products that have a one-to-one mapping between modules and functions (that is, each module is assigned for a specific function). Alteration of a specific module does not affect the functioning of other modules in such products.

[7] See also Clark *et al* (1987) for a detailed study on product development in the automobile industry.

[8] Baldwin and Clark (1997) give an example from the computer industry: a team can design a disk drive on its own as long as it obeys the overall requirements of the final product (for example, data transmission protocols; size and shape specifications of the hardware; interface standards) thereby ensuring that the disk drive (which is a distinct module) functions within the personal computer system as a whole.

[9] See Langlois (2002).

In the next section, we begin with component-sharing within firms; we discuss why firms may choose to develop a variety of products and the possible implications of having common components in their product varieties. In Section 3, we discuss component-sharing agreements between firms that also raise the issue of inter-firm standardization. We explore how firms can share components through licensing and joint product R&D and we consider the implications for competition. In Section 4, we give two examples for component-sharing in digital markets: 3-Dimensional video games and open-source software development. Finally, we conclude.

2 Component sharing within firms

One of the major advantages of modular design is the ease of changing product functionality.[10] Modularity enables alteration of a functional element of the product by simply changing the corresponding component without requiring any further changes in other components. Firms, therefore, can respond to changing markets and consumer demand rapidly and inexpensively by developing new products derived from existing modular products.

This feature of modular products generates a distinct cost structure for both product development and production. First, firms can benefit from economies of scope in product development, to the extent that some modules of the existing products can be used for creating new product variants. In such cases, new product development costs are likely to be inversely related to the degree of commonality in the existing and the new products. Second, modular design may also generate economies of scope in production. In particular, this can happen if there are economies of scale in producing certain product modules that are embedded in different product varieties.

Besides the cost-side benefits, such flexibility in developing product varieties may help firms to price discriminate (a demand-side motive) and to respond to entry (a strategic motive), which we explore in detail in this section.

Finally, product varieties can be achieved by user innovations. A modular architecture is a necessary condition for this to happen. Development of product variants by users can resolve the trade-offs the firms face when they position their product(s) in markets where consumers are relatively heterogeneous. We address this issue in subsection 2.4.

2.1 Economies of scope in production and the development of modular products

Informally, there are economies of scope in production when a single firm can produce a given level of output of each product variety at a lower cost than a combination of different firms that each produces a single variety.[11] Economies of scope are usually generated when some inputs (or facilities) are jointly used for producing different varieties.[12] In the context of modular products, economies of scope may arise if there are economies of scale in producing some product modules that are common in several product varieties. Although economies of scope of this sort is more likely to be present in the production of physical modular products, it cannot be ruled out for digital modular products, since

[10] See Schaefer (1999).
[11] See Panzar and Willig (1981) for a formal definition.
[12] For a series of examples in the context of traditional goods, see Bailey and Friedlaender (1982).

production of a particular (digital) product module that is common in several products may involve large fixed costs.

Eaton and Schmitt (1994) show how the presence of economies of scope in producing differentiated goods (which, in their setting, is implied by a flexible manufacturing system) can promote concentrated market structures. Eaton and Schmitt adopt the linear city model of Hotelling, and consider a flexible production system in which a "basic" product can be modified to produce different variants. Modifying a basic product entails: (i) a cost of switching the production process from one variant to another; and (ii) a per-unit cost of modification. The latter is proportional to the distance between the basic product and the variant,[13] whereas the former is not. The authors show that, with a perfectly inelastic demand, entry deterrence by a monopolist is the inevitable outcome and does not require introduction of too many product varieties.

Eaton and Schmitt assume that the degree of economies of scope depends on the distance between the basic product and the new product variants in the product space. This assumption is also appropriate for modular products. When a firm with a modular product introduces a new variety, it decides on the number of modules of the existing product that will also be embedded in the new product. While a higher degree of commonality (that is, more modules that are common) in different varieties implies a higher degree of economies of scope (due to economies of scale in producing common modules), it may also imply a lower degree of differentiation (a closer distance between the products in the product space). At one extreme, if two products have no common component, it is likely that they will be highly differentiated, but the differentiation will preclude economies of scope due to economies of scale in producing modules. At the other extreme, if all modules are common, then the products are identical. Therefore, in the light of the findings of Eaton and Schmitt, we may expect persisting dominance due to a natural barrier of entry (that is, a short-sighted monopolist may produce a number of varieties that leaves no scope for entry) with a modular product design. Before concluding so, however, one also needs to incorporate economies of scope in product development to the Eaton and Schmitt setting,[14] which we believe is likely to support this conjecture.

Economies of scope in product development may arise if some resources (for example, product knowledge, R&D technology) are jointly used in the development of different products. Such economies are quite common both in physical and modular product development. For example, automobile manufacturers use some product modules (for example, engine, break system) in a variety of models, instead of developing those components for each different model. Even in the absence of strategic motives, economies of scope in production and development of the modular products may suggest a larger number of product varieties offered to consumers compared to those integrated products with no such economies. Therefore, firms are more likely to benefit from modular design in markets where consumers highly value differentiated products.

2.2 Versioning with modular digital products

Versioning is a generic term for a common price discrimination practice that involves creating combinations of price and product characteristics in order to induce self-selection

[13] The modified unit cost of production applies for all varieties that are produced by the same firm.

[14] Although the authors consider a flexible manufacturing system, product development is perfectly inflexible; introducing a new product variant entails an exogenous sunk cost which is independent of how close the variant product is to the basic one, hence, there are no economies of scope in product development.

of the consumers. Providing consumers with varieties of different product characteristics usually entails product differentiation over the quality dimension.[15]

One special case of versioning involves damaging current (full) versions of products in order to introduce inferior versions.[16] Deneckere and McAfee (1996) provide many examples of damaged goods, which range from physical goods (laser printers) to digital goods (software) and show that a damaging strategy may result in a Pareto improvement. As Shapiro and Varian (1998b) argue, when there are high fixed costs involved in developing the full version, the initial fixed costs of product development can be recovered by creating additional revenues from the damaged version[17] if the cost of damaging is reasonably low. Belleflame (2005) also shows, in the presence of fixed costs of damaging, that versioning can be optimal only if damaging costs are sufficiently small. We believe that such fixed costs of damaging are likely to be lower, if present, for digital modular goods (for example, it may merely require the removal of some parts of the code to damage a software program). This implies that modular design, in particular of digital goods, can improve firms' profits by increasing their abilities to price discriminate at a lower cost.

Indeed, most of the examples for damaged goods are modular digital goods. As Shapiro and Varian (1998b) note, value-subtracting, or damaging, is less likely to be observed with physical goods, since versioning with those goods usually involves introducing a basic model first and then introducing add-ons to create superior versions. With physical modular products, this strategy simply implies enhancement or replacement of specific modules in order to create higher-quality versions. Therefore, producing higher-quality versions is more costly than producing basic versions, and may also involve a fixed cost of product development, which can be substantial for integrated products.

In the modularity literature, the ability to create such higher-quality product versions through replacement or upgrades of specific modules is referred to as modular upgradability. For example, Krishnan and Ramachandran (2004) define products with modular upgradability as those whose performance can be improved by replacing a minimal set of components. They argue that due to modular upgradability, firms may favor modular product design over an integrated product design, particularly in rapidly improving technology markets (for example, cell phones, digital cameras).[18] The authors' focus, however, is the possibility of consumer regrets and delays in purchases in markets like consumer electronics, where improved versions or upgrades are introduced sequentially and relatively rapidly, rather than price discrimination by self-selection. They argue that when products are upgradable in modules, products do not become entirely obsolete over time and can be updated incrementally.[19] For example, quality

[15] For versioning of information goods, see Shapiro and Varian (1998a,b), Bhargava and Choudhary (2001) and Belleflame (2005).

[16] Shapiro and Varian (1998a,b) call these "value-subtracted" versions.

[17] Along with such a market-expansion effect, damaging also creates a replacement effect, which tends to decrease revenues (some consumers switch from the high-quality version to the low-quality version). The net effect on revenues depends on the distribution of consumers with respect to their valuation for quality.

[18] See also Mikkola and Gassmann (2003); the authors state "shorter product life cycles" through incremental improvements such as upgrades, add-ons, and adaptations, as one of the main benefits of modular design.

[19] Modular upgradibility can also accelerate time-to-market: the firm can launch a basic version of its product fast and offer upgrade modules later. The new generation of video game consoles provides an example: Sony decided to provide a built-in high definition (HD) player in its new console, but had to wait until the end of 2006, when HD players actually became available, to launch its console. Choosing a different strategy from

improvements of a cell phone can be achieved locally by replacing a subset of components like batteries, storage, etc. Designing and pricing a product that is upgradable in modules (that is, commitment to localized quality improvements) can hence help a firm persuade prospective customers to buy a rapidly-improving product and can make modular design more profitable than an integrated architecture. The authors' arguments apply to both physical and digital modular products.

For both physical and digital products, modular design makes it easier to introduce different quality versions of the same product. Even in the absence of an intention to price discriminate, this feature can be important when firms face an uncertain demand.[20] Where digital goods are concerned, there may be other reasons why firms find it profitable to offer different quality versions of varied quality, such as creating a network in the presence of network effects (firms can provide lower-quality versions to build a customer base, which would increase the willingness to pay for the higher-quality version), building customer awareness (lower quality or restricted versions can be provided at marginal cost, so that potential customers can test out the higher quality version), and gaining in follow-on sales (firms can first build a customer base and then gain through sales of add-ons, upgrades, extensions, etc.)[21]

2.3 Response to entry and entry threats

As discussed earlier, economies of scope in developing modular products may give rise to a large number of product varieties offered by the same firm. Incumbent firms with modular products, therefore, may enjoy dominance in those markets where the entry is discouraged due to a crowded product space. While modular product design may lead to natural entry deterrence more often than the integrated products, its impact on strategic entry deterrence is not obvious.

A number of studies have analyzed introducing product variants for entry deterring purposes in both horizontal and vertical differentiation settings.[22] In his pioneering work, Schmalensee (1978) adopts the circular city model and shows that in the presence of entry threat, incumbent firms may introduce a larger number of varieties than they would otherwise.[23] At the first thought, the easiness of new product introduction implied with modular design may suggest that incumbents with modular products are more likely to engage in such strategies than those with integrated products. However, earlier studies on spatial preemption (see also Omori and Yarrow, 1982), do not consider the credibility problem, which is later pointed out by Judd (1985). Judd argued that in the absence of substantial exit costs, the threat by the incumbent to intensify competition via new product

---

Sony, Microsoft decided to commercialize a HD player as a separate module of its new console, hence was able to launch its console earlier in Spring 2006.

[20] See Sanchez (1998) who argues that demand-side uncertainties can be more efficiently mediated through strategies that improve a firm's ability to offer a range of product variations.

[21] See Shapiro and Varian (1998b), textbox "The Logic of the Free Version," and Bhargava and Choudhary (2001).

[22] For vertical differentiation settings, see Constantatos and Perrakis (1997) with a single incumbent and Canoy and Peitz (1997) for multiple incumbents.

[23] See also Brander and Eaton (1984) for a setting with competing multiple incumbents. They show that in the presence of a potential entrant, firms may commit to a fierce competition by choosing an "interlaced structure" where close substitutes are produced by different firms. Therefore, entry is deterred by competition.

introduction may not be credible.[24] Although entry costs may differ according to the product architecture (integrated or modular), we believe that the exit costs depend on other aspects (for example, reputation) than the product architecture itself, hence, modular design does not bring any advantage or disadvantage to resolve such credibility problems.

Even though exit costs are not quite different for modular and integrated products, costs of relocating products in the product space may be different (and much lower) for modular products. This, in turn, is likely to reduce the commitment value of product selections made by incumbents with modular products. Bonanno (1987) shows that when the product set-up costs are linear in the number of products, an entry deterrence strategy may involve product selection (location choice) rather than product proliferation. Clearly, entry deterrence by product selection constitutes a credible strategy only if product relocation costs are significant, which means that modular design may hinder the incumbents' ability to deter entry with their product selections.

Finally, modular design may provide incumbents with a superior flexibility in their entry accommodation strategies. Several papers, notably the one by Johnson and Myatt (2003), study how a multi-product incumbent may respond to entry by changing their product offerings. Johnson and Myatt show that in markets where there are distinct segments and in those where the entrant firm offers only low-end products (due for example to some technological asymmetry between the incumbent and the entrant) the incumbent firm's response to competition may involve introducing a lower-quality (fighting) brand.[25] In their analysis, Johnson and Myatt do not consider any fixed cost of product introduction. Given that introducing new varieties of integrated products may involve significant fixed costs, whereas such costs are likely to be lower with modular products, firms that adopt a modular product design may have more flexibility in responding to competition by introducing fighting brands. However, when the optimal response to entry involves product pruning, modular design does not necessarily bring more flexibility, since the exit costs associated with pruning (for example, in terms of reputation) are not necessarily different for modular and integrated products.

2.4 Product variants developed by users

Modular product design may eliminate the trade-off the firms face when they engage in customer targeting by simply enabling innovation by users.[26] Doraszelski and Draganska (2006) describe such a trade off when firms choose between introducing a targeted product that is tailored for customers' needs (for some market segment) and a general-purpose product. While targeted consumers (for example, basketball players) enjoy the "fit" product, that is, the product tailored for their specific needs (for example, basketball shoes), non-targeted customers (for example, tennis players) suffer from the "misfit". Such a trade-off is indeed evident in many integrated product settings (such as different varieties

[24] Several studies suggest settings in which spatial preemption can be credible. For example Choi and Scarpa (1992) argue that withdrawal of an incumbent's product may have a negative reputational effect on its other products (that is, exit may entail a reputational cost). Hadfield (1991) shows that a franchising contract can serve as a commitment device through delegation.

[25] This is because, in those markets, the marginal revenue curve may be increasing in some regions, which is the necessary condition for introducing a fighting brand to be an equilibrium strategy. Otherwise, the incumbent never responds to entry with offering a new variety, and may instead engage in product line pruning.

[26] Baldwin *et al* (2006) argue that innovation by users is an important source of both process and product innovations.

of sports shoes). The authors argue that the fixed costs entailed in manufacturing different products and the additional revenues from targeting other segments (which depend on the degree of competition in those market segments) may or may not make it optimal for the firm to target multiple market segments. In a two-firm setting with price competition, they provide conditions (on fixed cost, degree of competition, degree of fit, and degree of misfit) for the following equilibrium outcomes: both firms offer a general purpose product; market segmentation occurs via niche firms; and market segmentation occurs via full-line firms.

Modular design may resolve the problem of how increased fit for some consumers can lead to increased misfit for some others if it enables users to modify a general-purpose product for their specific needs. For physical goods, modular furniture is one common example. In such a case, the firm can produce a single variety while still targeting various different consumer groups. Two examples in modular digital goods are video games and open-source software. Video games often allow for collective creation by allowing the user to modify the game or to add new functionalities. These modifications may vary from basic "skins" (that is, new visual environments) to complete modifications of the game (that is, "mods" or levels), and are typically delivered through the Internet – either through the games' Web site or through those run by gamers themselves.[27] Open-source software is another example of how end users can achieve modular improvements. Bessen (2006) argues that this feature of open-source software is one important advantage compared to proprietary (closed) software that can not be modified by end users. Users' ability to modify such products can be particularly useful when users' needs are highly heterogeneous.[28]

In the digital setting, examples as such are restricted to markets where users are sufficiently sophisticated to undertake such modifications. Therefore, although it may be technically feasible, targeting multiple segments with a single product (user innovations through a modular design) may not prove to be a viable strategy.

3 Component sharing between firms

Different from component sharing within firms, component sharing between firms requires some coordination. This is because component sharing requires the determination of common interfaces between the firms. Such "standardization" processes may involve more firms than those firms that share components. As Farrell, Monroe, and Saloner (1998) put it, standardization of interfaces may occur in a "closed organization" (that is, a subset of firms) or in an "open organization" (that is, all firms in the industry).[29]

The literature on standardization has a focus on compatibility issues. Indeed, standardization implies compatibility, though compatibility can be achieved by other

[27] For example, in 1999, two mod makers, Minh "Gooseman" Le and Cliffe, created a mod for the video game Half-Life. The new mod, called Counter-Strike, soon became one the most popular multiplayer games on the Internet.

[28] For a specific case study, see Franke and von Hippel (2003) on Apache Security Software. In a survey of Apache users, the authors find that one-third of the respondents had integrated additional security modules to the standard Apache software package and about one-fourth had created new code to customize the software.

[29] When interfaces are common to all firms of the industry, standardization can be either achieved by regulation, or imposed by the dominant firm, or can simply be a market outcome. See Axelrod *et al* (1995) for a discussion on the standardization process.

means.[30] Compatibility issues have been studied in two different frameworks.[31] A first approach, which is adopted by most of the literature, considers products or services that exhibit network externalities. Under this approach, compatibility implies that firms share the same network, whereas incompatibility means that firms have different networks. A second approach is the "mix and match" approach, introduced by Matutes and Regibeau (1988), that considers a situation where consumers purchase complementary components of a system from (possibly) different suppliers.

Component sharing with modular products corresponds more to the mix and match approach, except that assemblers are the firms, not the consumers. Such a distinction is important, as consumers assemble components to fit the system better with their needs, whereas firms share and assemble components to economize on development costs.

Given that firms have common interfaces, component-sharing can take at least two different forms: licensing agreements or joint component development.

3.1 Component sharing through licensing

Firms make a binary, zero-one decision: whether or not to license an innovation to their potential competitors when they develop an integrated product. However, when product design is modular, firms may have the ability to undertake partial licensing.[32] Partial licensing may shape post-licensing competition, hence, may enable firms to fine-tune the trade-off between licensing rents and the potential competitive costs.

Digital modular products like software and computer-based information systems are trivial examples of how derivation products can appear around modular components that are shared through licensing agreements. For example, a leading developer of three Dimensional (3D) video games, id Software, initially designed and used 3D engines, which provide real-time computing techniques for special effects, for its own 3D video games. Facing the demand of other game publishers and developers, it licensed its 3D engines to enable other game developers to develop 3D games without developing their own engines. In section 4, we provide a detailed account of how id Software undertook component-sharing licensing.

There are several studies that analyze the strategic use of licensing to alter competitors' R&D and entry decisions.[33] Gallini (1984) considers a process innovation in an ex-ante licensing setting and shows how an innovator may engage in strategic licensing to deter the R&D activity of the entrant, who may develop a better technology. Modular product design brings flexibility to license the innovation partially, hence, may imply only partial R&D deterrence in this setting. However, firms may engage in partial licensing even if they are not threatened to be eliminated by a potentially better product. This is because by licensing their products only partially, firms can extract some R&D cost savings from the entrant firms without intensifying post-licensing competition too much.[34]

[30] Typically, by converters. See for example, Farrell and Saloner (1992).
[31] See Matutes and Regibeau (1996) for a survey on the literature.
[32] With a broad definition, partial licensing may involve any restriction that prevents the licensee to derive the full commercial benefit of the original invention.
[33] See for example, Gallini (1984), Gallini and Winter (1985), Rockett (1990a,b), and Yi (1999).
[34] See also Rockett (1990a), who analyzes the incentives of an innovator to license a process innovation that is inferior to its own technology (a technology that entails a higher cost of production than its own technology). Therefore, her setting suggests partial licensing for process innovations. However, partial licensing of a process innovation puts the licensee to a cost disadvantage, whereas this is not the case for product innovations.

Bourreau and Doğan (2006) consider an innovator who holds the exclusive rights to its innovation and faces a single potential entrant. The innovation has a modular nature and the innovator decides to license an arbitrary partition of it. The entrant can either develop its product from scratch, or it can enter by acquiring the license. If the entrant acquires a license, it pays a fixed licensing fee and invests in product development to complete the product unless a full license is granted. They assume that the degree of differentiation decreases with the size of the common component that is determined by the size of the license. Full licensing, therefore, leads to minimum differentiation unless the entrant develops some components on its own.

Bourreau and Doğan show that the factors that alter the sensitivity of the industry profits to the degree of differentiation (for example, the type of competition, cost asymmetries) affect the size of the license. A higher sensitivity implies a smaller license, hence a smaller common component in competing firms' products.[35] They also compare the incentives to license with a modular product design to those with an integrated design and show that modular product design may or may not imply more R&D deterrence. This is because modularity brings a flexibility in licensing, which implies that both full licensing and no licensing are less likely to occur.

Component sharing through licensing may have implications in terms of timing of entry. When entrants have access to a larger partition of a product innovation, it may require less time to develop and introduce their own products to the market. In such a case, the innovator may license a smaller partition of its product to delay entry, unless entry expands the market.

Finally, component sharing can be achieved though cross-licensing or patent pool agreements. Examples of patent pools include the MPEG-2 Digital Video pool, the DVD-ROM, and DVD-Video pools.[36] By contributing to such pools, firms save development costs (as they can use some components developed by others), but at the same time, the benefits of their innovation efforts also extend to other contributing firms.

3.2 Component sharing through joint product R&D

Similar to the licensing decisions, firms with integrated products make a zero-one decision whether or not to engage in joint product development with other firms. When products are modular, the decision is no longer a binary one, since firms can decide on the product components they wish to develop jointly. Such "partial" cooperation between firms (that may or may not be competitors in the product market) will result in component sharing.

This is a common practice in the automobile industry for obvious reasons; thanks to the modular automobile design, firms can choose to cooperate on the components for which the development costs are high (or those for which firms have complementary research assets) and choose to compete on components that are visible to consumers (that is, those that affect product differentiation). As Slywotzky[37] puts forth:

> "[...] These R&D efforts have been expensive and risky. If the automakers had collaborated from the outset, they could have saved billions of dollars. By collaborating on basic engine technology, which is largely invisible to most car buyers, the automakers would have been able to focus their innovation efforts in areas that have more impact on product design and the dealership experience. They would have been able to compete where it counts - where customers care the most. [...]"

[35] Similar results apply if the entrant has no outside option, though incentives for licensing are lower.
[36] See Lerner and Tirole (2004).
[37] Adrian Slywotzky, "When Bitter Rivals Should Team Up," *Chief Executive*, October 2005, 212, p.12.

As this example illustrates, firms can cooperate to develop some components and then develop the remaining components independently. Most of the literature does not consider this form of partial cooperation in R&D. The only exceptions we are aware of are Atallah (2005) and Goyal *et al* (2005). Atallah studies a standard cooperation model, in which firms can allocate resources both to cooperative and non-cooperative R&D. He shows that cooperative R&D and non-cooperative R&D reinforce each other. Goyal *et al* propose a model in which firms undertake in-house (non-cooperative) R&D on "core" projects and bilateral joint (cooperative) R&D projects with their rivals. One of their results is similar to Atallah: R&D investments in different projects are complementary. They also show that firms' profits decrease when the number of joint projects increases.

However, Atallah and Goyal *et al*, as well as the majority of the contributors to the R&D cooperation literature, ignore any costs of cooperation. Therefore, cooperation is always, at least in a weak sense, desirable for firms (since firms can replicate the non-cooperative equilibrium). There are very few papers that consider such costs, notably, Vilasuso and Frascatore (2000), Lambertini *et al* (2002), and Falvey *et al* (2006).[38] In the context of modular products, there is a cost of joint component development. Sharing more components through joint development is likely to leave less room for differentiation and may intensify competition. Therefore, firms will have to trade off between the benefits of joint R&D (for example, R&D cost sharing and coordination of investments) and such costs implied by component sharing.

4 Examples

In this section, we provide two examples to illustrate how firms can share components with licensing and joint product R&D.

4.1 Game engines and video games

Similar to any other software, a video game software typically consists of different components with standardized interfaces that execute different functionalities. In the video game industry, there has been a trend towards specialization for the development of the functionalities that compose a typical game (for example, sound, music, graphics, and content). The trend in development of "game engines" provides a good illustration for this.

A game engine is the core software component of a video game. It is a software platform that provides the main elements necessary to develop a video game: graphics (rendering), sound, music, physics, artificial intelligence, networking, etc. 3D game engines use standardized interfaces with other hardware or software modules. For instance, a graphics application programming interface like OpenGL or DirectX is used, which ensures compatibility with common hardware equipment. Game engines are themselves often designed in a modular way; an engine consists of various components and one component can be replaced by a more specialized component if needed. 3D game engines

[38] Vilasuso and Frascatore (2000) consider an exogenous fixed cost of forming a Research Joint Venture, which can be attributed to its management or auditing. In Lambertini *et al* (2002) firms develop a single product when they cooperate in product innovation, whereas they produce differentiated products when they do not cooperate. Therefore, cooperative R&D comes with a cost: it leads to a fierce competition post-innovation, unless firms collude at the competition stage. Falvey *et al* (2006) consider that coordination costs increase with the number of participants in the joint venture.

which manage real-time 3D graphics in first-person shooter (FPS) games represent the prominent category of game engines. The game engine programming model was popularized in the nineties following the success of the first 3D first-person shooter games by id Software (Doom[39]) and Epic Games (Unreal). Facing demand from other game publishers and developers, Id Software and Epic Games commercialized their 3D engines, but continued to develop their own 3D video games.

4.2 A shared component: 3D game engine

Currently, many game engines are available for game developers. Some game engines are available as open-source software, but they are hardly used for commercial video games. Others are proprietary software. Some companies like LithTech and NDL specialize in the game engine upstream market, whereas others like id Software, Epic Games, and Crytek, operate both in the upstream market and the downstream game market.

Although some game engines are available for free or for a very low fee, licenses for most engines, in particular the high-end game engines, can be very expensive to acquire. For instance, Epic Games charges a fixed fee of US $350,000 per platform (for example, PC, PS2) and a royalty of 3% of revenues from the game for its Unreal Engine 2. A license of the Quake III Arena engine by id Software includes a fixed fee of US $250,000 for a single title plus a royalty of 5% of the wholesale price.

There are two main reasons for why game developers often prefer to acquire the license of a game engine rather than develop it. First, the development cost of a game engine is high relative to the total development cost of a game. Development usually takes more than a year, hence represents a heavy investment for the game publisher. Acquiring the license of a game engine enables the game developers to share the development costs. This is, in particular, true for high-end game engines, which are the result of years of research and development. Furthermore, for small game developers for which the development cost of engines can be prohibitively high, licensing is the only way of entering the market.

Second, acquiring the license of a game engine instead of developing it from scratch accelerates time-to-market. Since developers spend less time on the game engine, they can spend more time and resources on the game content, which allows for specialization in game design and improves the quality of the final product. Furthermore, a short time-to-market is critical in the game industry, since gamers' tastes evolve quickly.[40]

Acquiring the license of a renowned engine has drawbacks as well, as each engine is limited in terms of the types of games that can be developed with it. Thus, the degree of differentiation between rival game developers that use the same engine is likely to be low, and fierce competition is likely to occur due to a higher degree of substitutability.

Developers who are willing to differentiate may choose either to build their own engine from scratch or to use some components of an engine but not all and develop the remaining engine components from scratch. The trade-off between acquiring a full engine and a partial engine is summarized by Tim Sweeney of Epic Games:

> "[Game engines] are aimed at developers who want a complete, off-the-shelf solution so that they can focus on their gameplay and content. Game components like RenderWare are aimed at

[39] More than 2.9 millions copies of Doom have been sold.

[40] Using a renowned game engine may also serve as a signalling device for the final product. If the game uses a well-known game engine, consumers can expect the standard quality of the games which use this engine.

> developers who are developing their own technology, but don't want to reinvent the wheel in some area of technology that's already well-defined."[41]

On the game engine developers' side, the companies that are vertically integrated (for example, Epic Games and id Software) face a trade-off between extracting additional revenues from licensees and increased competition in the game market due to lessened differentiation. The vertically integrated firms can limit the negative effects of licensing by addressing market niches for their games, or by giving their licensees incentives to differentiate. For instance, id Software allows its licensees to improve their engines and to license their incremental innovations to other id Software game engines licensees.

4.3 Open source software development

Software is a natural candidate for modular design; it consists of several elementary components that can be re-used for other purposes, and interfaces between components can be standardized. Object-oriented programming languages represent one step towards modular software design. Open-source software represents another step. A software is said to be open-source when the source code is available to third parties, and it can be freely distributed, and when derivative works are allowed. Well-known examples of open-source software include the operating system Linux and the web server Apache.

The nature of open-source software creates strong incentives to develop modular software instead of integrated software; indeed, it gives incentives to develop code with functionally separable components so that one developer can modify one component without altering the functioning of other components.[42]

The modular nature of open-source software enables a software company (or a developer) to use some open-source modules for an application and to develop the other needed modules from scratch. However, the license of the open-source software can restrict the type of use of the final software. For instance, the General Public License (GPL) obliges the developer to circulate its software under the GPL (that is, in particular, under an open-source format). Other licenses (such as Berkeley Software Distribution, or BSD) allow the developer to commercialize the final software under a proprietary license. Red Hat Linux is an example of open-source software that was successfully packaged as a commercial product.

4.4 Joint component development in open-source software

Open-source software enables component sharing: software companies can cooperate to design some components of a software through an open-source project, develop independently other components of the software, commercialize the final software and compete in the market.

---

[41] Cited by Jake Simpson, "Game Engine Anatomy 101," April 2002, http://www.extremetech.com/

[42] Whereas with an open-source software an upgrade can be made by replacing an existing module with a new module, with an integrated proprietary software, an upgrade might necessitate to rewrite the software entirely. For instance, Microsoft rewrote a large part of the source code of its operating system when it moved from Windows 98 to Windows XP and then from Windows XP to Windows Vista (which explains in particular why these evolutions took several years). Industry analysts argue that in the future Microsoft will have to develop a more modular operating system to limit the amount of code to be written for an upgrade (hence, to accelerate time-to-market of upgrades). See: Aaron Ricadela, "Windows after Vista," http://www.schoolcio.com/showArticle.jhtml?articleID=192701061 (accessed October 2006) and Matthew Broersma, "Windows Vista the last of its kind," 25 August 2006, http://www.techworld.com/news/index.cfm?newsID=6718 (accessed October 2006).

One example of joint component development is given by the Apache community. Large companies such as IBM or Sun contribute to the open-source Apache project. The Apache Web server application provides them with the needed basic components to develop a Web server. These companies add proprietary components to the Apache application and commercialize Web servers under their brand name. For instance, IBM states that its HTTP servers are powered by Apache, and includes additional functionalities, such as SSL for secure transactions, and support.[43]

IBM presents its strategy as "cooperation on standards, competition on implementation."[44] Adrian Bowles describes IBM's strategy regarding open-source software as follows:

> "By contributing to open standards efforts and open source projects, IBM provides input to the processes and develops a high degree of expertise throughout the organization. IBM can then focus on creating the component pieces of a complete solution that are most appropriate for its current and future needs and resources. At the same time, the open source approach encourages development of ongoing incremental improvements to technology solutions that might otherwise have been thwarted by proprietary barriers."[45]

Another example of partial cooperation is given by the ObjectWeb project. ObjectWeb is an open-source initiative that aims at developing an open-source middleware. A middleware is a flexible and reusable platform which provides the functionalities needed to develop a software application. It is the software layer that lies between the operating system and the application. Therefore, an open-source middleware can be viewed as an example of modular cooperation; developers cooperate to build a common middleware, while developing independent and competing applications.

There are both costs and benefits to joint component development for software. We can identify at least three types of benefits. First, some benefits accrue due to the reduced development costs; instead of bearing the development cost of some software components, a firm shares this cost with other firms (including, perhaps, competitors). Economies of development costs are in particular important for very complex software that takes a lot of time and expertise to write (such as Web servers or operating systems). For this type of software, an open-source project has the following advantages: (i) the large number of developers,[46] and (ii) the modular nature of the software which enables to add new modules or modify existing modules without altering the overall consistency. Second, for very complex software, an open-source design can also lead to higher quality. Third, as Lerner and Tirole (2005) remark, if there are strong network effects or switching costs, there are benefits for consumers to have a "standard" (unique) software component, which is the case with joint component development.

Open source software is not immune to coordination costs. Furthermore, a major cost of cooperative development can be the reduction in the possibilities of differentiation. Therefore, the optimal degree of cooperation would fine-tune the trade-off between economies in development costs and intensified competition due to a lower degree of

[43] For instance, IBM supported the development of Simple Object Access Protocol (SOAP), by devoting technical resources to its specification and to its development (SOAP4J for Java) for the Apache open source software. Then, IBM incorporated SOAP into its WebSphere commercial application server.
[44] See Adrian Bowles, "The open standards imperative: IBM's open-for-business strategy," March 2002, http://www.ibm.com/software/solutions/webservices/pdf/OpenStandards_wp.pdf (accessed October 2006).
[45] Adrian Bowles, "The open standards imperative: IBM's open-for-business strategy," March 2002.
[46] As an example, we may cite the Gnome community case, with almost 500 developers.

differentiation. For example, Bill Weinberg[47] defines the Value Line as "the level at which technology and value supplied by open-source and shared community resources tapers off and opportunities for third parties to add value begin." He identifies three types of stack (software bundle): a short stack leaves most of the development to the company which uses the stack, whereas a tall stack provides most of the needed software. While a tall stack accelerates time-to-market, it leaves little or no room for product differentiation and branding. For Weinberg, the "just right" stack trades off "between completeness and flexibility, between time-to-market and room to add visible value."

5 Concluding remarks

Modular design has been adopted in major industries of both physical and digital goods (for example, automobiles and software, respectively). Different from integrated products, modular design enables product component sharing within and between firms. Such possibility of component sharing is likely to influence firms' innovation strategies. A firm can use common components to develop product varieties, due to economies of scope in production and development, for price discrimination purposes and as a response to entry threat. Different firms can also agree on common interfaces and share components through licensing and joint component development agreements.

The literature on product innovation implicitly considers integrated products but does not account for the distinct feature of modular products. In this paper, we highlighted possible implications of modularity of firms' innovation strategies. We believe that more formal research will enhance our understanding on modular products and innovation.

6 References

Atallah, G. (2004) "The Allocation of Resources to Cooperative and Noncooperative R&D," *Australian Economic Papers*, 43: 435-447.

Axelrod, R., W. Mitchell, R.E. Thomas, D.S. Bennett and E. Bruderer (1995) "Coalition Formation in Standard-Setting Alliances," *Management Science*, 41: 1493-1508.

Bailey, E.E. and A.F. Friedlaender (1982) "Market Structure and Multiproduct Industries," *Journal of Economic Literature*, 20: 1024-48.

Baldwin, C.Y. and K.B. Clark (1997) "Managing in the Age of Modularity," *Harvard Business Review*, September – October: 84-93.

Baldwin, C., C. Hienerth and E. von Hippel (2006) "How User Innovations Become Commercial Products: A Theoretical Investigation and Case Study," *Research Policy*, 35: 1291-1313.

[47] Bill Weinberg, 2006, "The New Era of Mobile Linux Ubiquity," White Paper, PalmSource, http://www.palmsource.com/opensource/WP_MLU_FINAL.pdf (accessed October 2006).

Bargigli, L. (2005) "The Limits of Modularity in Innovation and Production," CESPRI Working Paper, 176.

Belleflamme, P. (2005) "Versioning in the Information Economy: Theory and Applications," *CESifo Economic Studies*, 51: 329-358.

Bessen, J. (2006). Open Source Software: Free Provision of Complex Public Goods. In J. Bitzer & P.J.H. Schröder (eds), *The Economics of Open Source Development*. Amsterdam: Elsevier B.V.

Bhargava, H.K. and V. Choudhary (2001) "Information Goods and Vertical Differentiation," *Journal of Management and Information Systems*, 18: 89-106.

Bonanno, G. (1987) "Location Choice, Product Proliferation and Entry Deterrence," *Review of Economic Studies*, 54: 37-45.

Bourreau, M. and P. Doğan (2006) "Component Sharing Through Licensing," Mimeo, ENST.

Brander, J.A. and J. Eaton (1984) "Product Line Rivalry," *American Economic Review*, 74: 323-34.

Canoy, M. and M. Peitz (1997) "The Differentiation Triangle," *Journal of Industrial Economics*, 45: 305-28.

Choi, C.J. and C. Scarpa (1992) "Credible Spatial Preemption through Reputation Extension," *International Journal of Industrial Organization*, 10: 439-47.

Clark, K.B., W.B. Chew and T. Fujimoto (1987) "Product Development in the World Auto Industry," *Brooking Papers on Economic Activity*, 3: 729-71.

Constantatos, C. and S. Perrakis (1997) "Vertical Differentiation: Entry and Market Coverage with Multiproduct Firms," *International Journal of Industrial Organization*, 16: 81-103.

Deneckere, R.J. and R.P. McAfee (1996) "Damaged Goods," *Journal of Economics and Management Strategy*, 5: 149-74.

Doraszelski, U. and M. Draganska (2006) "Market Segmentation Strategies of Multiproduct Firms," *Journal of Industrial Economics*, 54: 125-49.

Eaton, B.C. and N. Schmitt (1994) "Flexible Manufacturing and Market Structure," *American Economic Review*, 84: 875-88.

Ernst, D. (2004) "Limits to Modularity: A Review of the Literature and Evidence from Chip Design," East-West Center Working Papers, Economics Series, No. 71.

Falvey, R., J. Poyago-Theotoky and K. Teerasuwannajak (2006) "Coordination Costs: A Drawback for Research Joint Ventures?" Working Paper, Loughborough University.

Farrell, J. and G. Saloner (1992) "Converters, Compatibility, and the Control of Interfaces," *The Journal of Industrial Economics*, 40: 9-35.

Farrell, J., H.K. Monroe and G. Saloner (1998) "The Vertical Organization of Industry: Systems Competition versus Component Competition," *Journal of Economics & Management Strategy*, 7: 143-182.

Franke, N. and E. von Hippel (2003) "Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software," *Research Policy*, 32: 1199-1215.

Gallini, N.T. (1984) "Deterrence by Market Sharing: A Strategic Incentive for Licensing," *American Economic Review*, 74: 931-941.

Gallini, N.T and R.A. Winter (1985) "Licensing in the Theory of Innovation," *RAND Journal of Economics*, 16: 237-252.

Gershenson, J.K., G.J. Prasad and Y. Zhang (2003) "Product modularity: definitions and benefits," *Journal of Engineering Design*, 14: 295--313.

Goyal, S., A. Konovalov and J.L. Moraga-Gonzalez (2005) "Hybrid R&D," Working Paper, University of Essex.

Hadfield, G. K. (1991) "Credible Spatial Preemption through Franchising," *RAND Journal of Economics*, 22: 531-43.

Huang, C.-C. (2000) "Overview of Modular Product Development," Proc. Natl. Sci. Counc. ROC(A), 24: 149-165.

Johnson, J.P. and D.P. Myatt (2003) "Multiproduct Quality Competition: Fighting Brands and Product Line Pruning," *American Economic Review*, 93: 748-74.

Krishnan, V.V. and K. Ramachandran (2004) "Combining Product Design and Pricing to Manage Rapid Sequential Innovation," Mimeo, University of Utah.

Lambertini, L., F. Lotti and E. Santarelli (2004) "Infra-industry Spillovers and R&D Cooperation: Theory and Evidence," *Economics of Innovation and New Technology*, 13: 311-28.

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

Lerner, J. and J. Tirole (2004) "Efficient Patent Pools," *American Economic Review*, 94: 691-711.

Lerner, J. and J. Tirole (2005) "The Economics of Technology Sharing: Open Source and Beyond," *Journal of Economic Perspectives*, 19: 99-120.

Matutes, C. and P. Regibeau (1988) "Mix and Match: Product Compatibility without Network Externalities," *The RAND Journal of Economics*, 19: 221-234.

Matutes, C. and P. Regibeau (1996) "A Selective Review of the Economics of Standardization. Entry Deterrence, Technological Progress and International Competition," *European Journal of Political Economy*, 12: 183-209.

Mikkola, J.H. and O. Gassmann (2003) "Managing Modularity of Product Architectures: Toward an Integrated Theory," *IEEE Transactions on Engineering Management*, 50: 204-218.

Omori, T. and G. Yarrow (1982) "Product Diversification, Entry Prevention and Limit Pricing," *Bell Journal of Economics*, 13: 242-48.

Panzar, J.C. and R.D. Willig (1981) "Economies of Scope," *American Economic Review*, Papers and Proceedings of the Ninety-Third Annual Meeting of the American Economic Association, 71: 268-272.

Rockett, K.E. (1990a) "The Quality of Licensed Technology," *International Journal of Industrial Organization*, 8: 559-574.

Rockett, K.E. (1990b) "Choosing the Competition and Patent Licensing," *RAND Journal of Economics*, 21:161-171.

Sanchez, R. (1993) "Strategic Flexibility, Firm Organization, and Managerial Work in Dynamic Markets: A Strategic Options Perspective," *Advances in Strategic Management*, 9: 251-291.

Sanchez, R. and J.T. Mahoney (1996) "Modularity, Flexibility, and Knowledge Management in Product and Organization Design," *Strategic Management Journal*, 17: 63-76.

Schaefer, S. (1999) "Product Design Partitions with Complementary Components," *Journal of Economics Behavior & Organization*, 38: 311-330.

Schmalensee, R. (1978) "Entry Deterrence in the Ready-to-Eat Breakfast Cereal Industry," *Bell Journal of Economics*, 9: 305-27.

Shapiro, C. and H.R. Varian (1998a) "Information Rules," Harvard Business School Press, Boston, Massachusetts.

Shapiro, C. and H.R. Varian (1998b) "Versioning: The Smart Way to Sell Information," *Harvard Business Review*, November-December: 106-114.

Staudenmayer, N., M. Tripsas and C.L. Tucci (2005) "Interfirm Modularity and Its Implications for Product Development," *Journal of Production Innovation Management*, 22: 303-321.

Vilasuso, J. and M.R. Frascatore (2000) "Public Policy and R&D When Research Joint Ventures Are Costly," *Canadian Journal of Economics*, 33: 818-39.

Yi, S.-S. (1999) "Entry, Licensing and Research Joint Ventures," *International Journal of Industrial Organization*, 17: 1-24.