9

With a Little Help . . .

Do not hold the delusion that your advancement is accomplished by crushing others.

—*Marcus Tullius Cicero*[1]

INSIDE THIS CHAPTER

- The design of software platform ecosystems
- Advantages and disadvantages of one-sided versus multisided strategies
- Why software platform integration varies across industries and over time

This chapter and the next two focus on three key strategic decisions faced by software platform vendors. The first, considered in this chapter, is the scope and integration of the business. Should it produce complements that work with the software platform or leave that to others? Should it operate a multisided platform, and if so, what sides should that platform have? The second decision, examined in Chapter 10, involves how to price to get all the sides on board and to interact with each other. Should it levy a fixed charge for accessing the platform or a variable charge for using it—or both? How much of its profit should it seek from each side? The third decision, considered in Chapter 11, involves what features and functionality to include in the software platform itself. Should it offer several alternative platforms or just one?

In the preceding five chapters, we saw that some software platform vendors define the scope of their activities quite narrowly. In recent years

1. http://www.orangejobs.com/nz/graduates/articles/interviews.htm.

PalmSource, for instance, has focused on developing and licensing the Palm OS operating system. Others define the scope of their activities more broadly and have offered most or all components of the complete system. That is the case with the iPod/iTunes platform. We have seen differences in integration within industries, Apple versus Microsoft being perhaps the most familiar contrast. Over time, Palm's metamorphosis from a complete systems provider to a supplier of only operating systems (PalmSource) has been perhaps the most striking. Many platform vendors have been partially integrated into applications, producing some themselves and, in a variety of ways discussed in this chapter, encouraging third parties to produce others. Overall, the industries we have studied have tended to exhibit less integration over time, though the process has hardly been steady or uniform. This chapter attempts to make sense of all this.

Ecosystem Participants and Structures

The first strategic issue a software platform vendor must consider is the structure of the ecosystem that surrounds it. What groups participate in this ecosystem or could? How might they best contribute to helping get the platform off the ground and contribute to long-term profitability?

End users, the final customers for the systems built around the software platforms, are common to all platforms. They typically are interested in working systems, not unconnected components, but whether they obtain those systems from one firm or from several varies from industry to industry and from time to time. Sometimes the software platform vendor deals directly with end users (an example is Sony's PlayStation), but it is at least as common to reach end users only through a third party (Microsoft's Windows Mobile). In some cases end users can be profitably divided into subgroups for pricing purposes, as we discuss in the next chapter.

End users apart, every other participating group is a potential *complementor*, a provider of products or services that are complements to the software platform and that therefore enhance the value of the

platform when their quality increases or their price falls.[2] Hotel rooms in Las Vegas and flights to Las Vegas are complements; hotel rooms in Las Vegas and hotel rooms in Honolulu are substitutes. If software platform vendors decide not to produce a particular complementary product themselves, they need to encourage complementors to affiliate with their platform and to invest in making their complementary products better and cheaper.

Producers of basic *hardware*—the systems, containing one or more CPUs, on which the software platform can run—are important complementors in all the industries we have examined so far (they won't be for two industries we look at in Chapter 12). Within platform industries, there is variation in the extent to which the same firms make the hardware and software platforms; the key exception is video game providers, in which all successful firms have made integrated hardware and software platforms. RealNetworks has never been in the hardware business. Apple has never left it.

When an ecosystem includes independent hardware producers, they need to be courted by the software platform provider. Microsoft needs to convince computer makers to build computers that run on Windows. PalmSource must convince companies to design and produce PDAs based on the Palm OS. This courtship takes the form of a rather intricate dance in periods of rapid innovation and changing technical standards, since quick changes in hardware and software need to be coordinated among independent firms if systems are to function well and the ecosystem as a whole is to prosper. At the same time, independent hardware vendors frequently also serve as distributors of the software platform to users. PalmSource does not license the Palm OS directly to end users. Instead, the Palm OS is licensed to device makers, and the latter install them on their products before selling systems to end users.

An interesting exception is the mobile phone industry, in which *mobile network operators* are mainly responsible for the distribution of mobile phones (the basic hardware) and their software platforms to end users. Symbian deals with phone manufacturers, and they in turn deal with the network operators. Microsoft, on the other hand, has not had much luck

2. Karl Case and Ray Fair, "Principles of Economics," 3rd ed. (Upper Saddle River, N.J.: Prentice Hall, 1994), pp. 82–83.

persuading major manufacturers to use its mobile phone operating system; it has therefore concentrated on selling phones made by second-tier manufacturers directly to mobile network operators. In the case of i-mode, the software platform sponsor, NTT DoCoMo, is also the network operator and thus the distributor of mobile phones equipped with i-mode to end users.

The production of *peripheral equipment* is a significant business in many of the industries we have examined. This is just as true for Apple iPods in 2005 as it was for Windows PCs in 1995. The boundary between hardware and peripheral equipment is to some extent dependent on both the state of technology and hardware makers' design decisions, of course. Microsoft's early operating system for the IBM computer didn't have a graphical user interface (GUI) and didn't need a mouse. That changed with the development of Windows. However, it turns out that computer makers have relied on third parties to make mice rather than doing it themselves. Interestingly, many are made by Microsoft, which integrated into mouse production in 1983 mainly to be sure that the sort of mouse specified by its nascent Windows system would be available in the marketplace. Microsoft developed and patented a mouse that could connect to a PC through an existing serial port rather than to a special card installed within the computer. This innovation reduced the cost of the mouse and thus of mouse-using computers running Windows. Apple as a vertically integrated hardware and software platform maker has always produced its own mice.

Applications are the third major category of complements. Applications are products that are typically licensed directly to end users. Ranging from word-processing programs to "shoot-'em-up" games to ringtones, they are key actual or potential participants in software platform ecosystems. Many software platforms begin by providing their own applications. That was the case with the Apple Newton and Palm Zoomer and many of the video game consoles. Others begin with a small stock of third-party applications. The original IBM PC began with Microsoft's languages along with a few applications such as VisiCalc that were ported by their developers from CP/M. Almost all software platforms end up relying mainly on third-party developers as they mature.

Finally, *content* is an important complement for many media-oriented software platforms. There is very little evidence that software platforms produce their own content. They typically encourage third parties to make it available. That is the case with media players: Apple, Microsoft, Real, and others facilitate content providers and content owners to make everything from songs to news to videos available for their media platforms. In some case media-oriented platforms license content, sometimes exclusively, and then provide it to end users. That's the case with Apple's iPod/iTunes platform and RealNetwork's Rhapsody music service. Some software platforms, such as iMode, work as actively with content providers as others work with application developers, to sustain a rich ecosystem of complementary content provision.

When one describes the extent to which a software platform is integrated into its ecosystem, it is important to recognize that partial integration is common and varies importantly in extent. Microsoft writes some applications to run on Windows and some games for the Xbox, for instance, but third-party vendors are important in both cases, and their importance has varied over time. For a time Apple did license the Macintosh operating system for use on third-party hardware, but these licensees were never very important in aggregate, while Symbian licensees produce all of the smart phones that run the Symbian platform. Some changes in integration, particularly partial integration, reflect accidents of history and the marketplace. If third parties had offered a quality two-button mouse in 1983, it seems unlikely that Microsoft would have gotten into the mouse business. Similarly, Microsoft's Word and Excel products were successful on the Macintosh platform when Microsoft was still working with IBM to develop OS/2, and this earlier experience clearly contributed to the success of these products in the Wintel segment. It is hard to imagine that this was the outcome of a conscious long-term plan, as evidenced by Bill Gates's plea to Apple's Scully to make the Mac OS platform ubiquitous (see Chapter 4).

The closeness of the contractual and informal relationships between participants in different parts of a business ecosystem varies from arm's-length, anonymous spot market transactions, as in the textbook wheat market, to long-term, joint-venture–like arrangements that are hard to

distinguish from integration by ownership. In the case of the software platforms we have examined, even the weakest relationships are far deeper than the arm's-length relationships one sees in many one-sided industries. Software platforms can't have direct relationships with the thousands of small developers, hardware makers, and peripheral device makers. Yet they document and make APIs available to developers, provide interface information to hardware and peripheral makers, and make sure their platforms have the relevant drivers for the peripherals. And they develop relationships through large developer conferences and small focus groups that bring some of these smaller players together. At the other extreme, software platforms often have deep relationships with several larger partners. These relationships involve regular exchange of information and joint work on defining new standards and specifications. They may also involve joint investments in product development or marketing. Representatives from Microsoft's Xbox and Sony's PlayStation divisions, for instance, both spend a great deal of time with Electronic Arts. At the furthest extreme, Symbian spends a great deal of time with its major backers, especially Nokia.

Key Determinants of Integration

Transactions Costs

The economic literature on determinants of the scope of firms' activities effectively began with Ronald Coase's classic 1937 paper titled "The Nature of the Firm."[3] Coase argued that competition generally forces firms to operate at least cost. For instance, an auto producer will make its own steel if and only if it is cheaper to make steel than to buy it. This of course has a touch of tautology to it. The novel and powerful aspect of Coase's analysis was his focus not on the production cost of making steel but on comparing the *transactions costs* of alternative methods—firm and market—of organizing the relationship between steel and auto production, of achieving coordination and motivation. Further work by

3. Ronald Coase, "The Nature of the Firm," *Economica* 4 (1937): 386–405. A very nice summary of the main themes in this literature, with references, is provided by John Roberts, *The Modern Firm* (Oxford: Oxford University Press, 2004), chap. 3.

economists following Coase has enhanced our understanding of the determinants of these important costs.[4]

For some kinds of goods and services, the market is generally superior, and integration is correspondingly uncommon. To focus on a polar case, banks use lots of pencils, but not even the biggest banks make their own. On the one hand, entry into pencil production seems relatively easy, so banks can count on competition to hold pencil prices close to costs. Moreover, it is relatively easy to specify pencil quality, to compare pencils from different vendors, and to switch from one pencil supplier to another, so arm's-length spot market competition can be relied on to produce both good quality and low prices. On the other hand, there is no reason to believe that banks are particularly good at producing pencils, and devoting top management time to pencil production takes it away from competing in banking.

The economics literature argues that changing any of the conditions enumerated in the preceding paragraph tends to tilt the organizational decisions toward integration. If the relevant market is not competitive or if the buyer has special advantages in production, for instance, integration becomes more attractive. Partial integration is sometimes used as a device to deal with upstream market power, because it both adds competition and makes an implicit threat of further integration. And when technical change is rapid, or standards are in flux, or interface specifications are evolving, the software platform provider may be able to optimize system performance only by also producing other system components.

A factor that is particularly relevant in this context is the difficulty of writing a contract that deals acceptably with all contingencies. Suppose, for instance, that Sony believes that a *great* submarine game would be a powerful complement to PlayStation 2 and thus sell millions of copies, but that no such game exists yet, so it decides to contract with a third party to develop one. Sony can write a contract for the development of a submarine game with a long string of specified technical properties. It would be reasonably straightforward to verify whether such a contract

4. The work of Oliver Williamson has been particularly important. See Oliver E. Williamson, *The Economic Institutions of Capitalism* (New York: Free Press, 1985).

had been breached. It would also be straightforward to write a contract that required a *great* game by specifying that it had to achieve a particular sales target. But it would be hard to determine if the failure to achieve greatness was the fault of the developer or because of a poor console platform. Moreover, if the project were technically adventurous, it might fail either because the developer didn't do a good job or because the project was infeasible. It would be difficult for a court or anyone else to decide objectively which was the case and therefore enforce the contract. Integration is one way to deal with these sorts of problems. Management decides whether or not the game its own staff produces is great or the reasons why the project failed, little or no time is spent writing a formal contract, and litigation is essentially ruled out.

Contracting problems can be an important impediment to innovation in systems businesses.[5] If, for example, the interface between hardware and operating system is well defined and unchanging, independent hardware and operating system vendors can take it as a specification and innovate more or less independently.[6] But if innovation is architectural and involves changes in that interface and thus in key technical specifications, coordination is essential and contracting generally does not work well, as both the Pentagon and many buyers of custom homes have learned to their sorrow.

The Multisided Solution

Instead of making a complement internally (integration) or buying it from a third party (contracting), platform vendors often induce a third party to supply it directly to end users and, possibly, to pay for the privilege. Suppose Sony persuades one or more independent developers that a great submarine game for PlayStation 2 would sell millions of copies, and it provides development tools that make it easy to develop games. If the royalty Sony charges on PlayStation 2 games is reasonable,

5. Kevin Boudreau, "How Does 'Openness' Affect Innovation? Evidence from Mobile Computing" (MIT Sloan School of Management working paper) (Cambridge, Mass.: MIT, 2005).

6. This is, of course, the modular approach to design discussed in Chapter 2. See Carliss Baldwin and Kim Clark, *Design Rules I: The Power of Modularity* (Cambridge, Mass.: MIT Press, 2000).

developers will have strong incentives to invest in the hope of developing a great submarine game: a great game will make a lot of money (probably a large multiple of what Sony could pay in-house developers), while a lousy game will return little or nothing (certainly less than Sony would likely pay its in-house developers if their game bombed). Because of these strong incentives, which are aligned with Sony's interests, no game-specific contract is necessary. Similarly, Microsoft doesn't have to try to use a contract to persuade Dell to produce high-quality, inexpensive computers, since Dell already has very strong incentives to do just that.

This multisided approach has its own problems, though.[7] First, if there are only a few possible suppliers of a key complement, there is a pricing problem.[8] Suppose firm *A* sells software platform *a*, firm *B* is the only producer of compatible hardware product *b*, consumers are interested only in systems that combine the two products, and *A* has adopted a multisided approach in which both it and *B* set prices independently. Because products *a* and *b* are complements, if either *A* or *B* raises its price, sales of both products will fall. But when *B* considers raising its price, it will take into account only the resulting fall in its sales, not the reduction in *A*'s sales. The reverse holds for *A*. The result will be a total system price (for *a* plus *b*) that is *above* the profit-maximizing level. Thus *A* has to share system profits with *B*, and those system profits are lower as a result of independent pricing than they would be through coordination.

What is the cure? From *A*'s point of view, one cure is to have many competing producers of good *b*. Competition will then hold the price of *b* close to cost (including a reasonable return on capital) regardless of *A*'s pricing, so that *A* both effectively determines the system price (via the price of *a*) and captures all the economic profit. Generally, it is more

7. Annabelle Gawer and Michael Cusumano (Platform Leadership: How Intel, Microsoft, and Cisco Price Industry Innovation [Boston: Harvard Business School Press, 2003]) provide interesting discussions of several of the points made in this paragraph and the next in the context of the computer industry.

8. Testimony of Kevin Murphy in *United States v. Microsoft*, No. 98-1233.—Augustine Cournot, *Researches into the Mathematical Principles of the Theory of Wealth*, trans. Nathaniel Bacon (New York: Macmillan, 1927) (original in French, 1838).

attractive to rely on others to supply a complement (instead of buying it or making it), all else equal, if there are many producers of that complement who compete intensely. Hence the common strategic advice, "Commoditize the complements."

On the other hand, potential complementors will invest only if they expect their investments to earn a reasonable return in the marketplace. That often depends critically on how they expect the platform vendor to behave. This raises some interesting dilemmas. For instance, while partial integration into production of complements can provide a hedge against failure of the market to produce what is needed, it can also, by threatening more intense competition, inhibit desirable third-party investment. (Similarly, Chapter 11 notes that software platforms commonly innovate by adding features and functionality that had previously been supplied by third-party applications. While this process makes platforms more valuable to both end users and application developers, it in effect intensifies the competition expected by the latter.) Expectations are not nearly so important for internal development or development by contract.

Finally, while the provider of a complete system can determine the direction of technical change internally and informally, this process becomes much more complex when system components are supplied by different parties. Sometimes a single entity, often the software platform provider, emerges as the driving force and system regulator. This seems clearly the case in video games, for instance, where console vendors both drive innovation and serve as gatekeepers for game developers to promote both quality and variety. DoCoMo plays a similar gatekeeper role in the i-mode ecosystem. But, as the world of PC games illustrates, the existence of a gatekeeper is not inevitable. The costs of setting up and operating an effective gate vary from case to case, as do the net benefits of managing entry (and thus to some extent limiting creativity).

Moreover, when several ecosystem participants have critical knowledge, leadership in the innovation process is often shared. And the identity of the leaders is not necessarily predetermined. In personal computers, for instance, IBM initially played the lead role in driving innovation but soon lost it to a combination of Microsoft and Intel. Microsoft needs to work with Intel, hardware makers, and application

developers to ensure that the systems in which all have a financial stake take full advantage of advances in a wide range of technologies.

Among the platform vendors we have discussed, DoCoMo has arguably the most complex task of coordinating innovation around its mobile Internet i-mode platform. Whenever it adds a new feature or service (e-payment, for example), it has to work with handset makers to include the corresponding chip or software in their phones. Then it has to explain to content and service providers how to build services based on the new feature. And of course it needs to do some plumbing itself on its network, adding software on its application servers or even upgrading the physical network gear. Nokia, to take another smart phone example, seems the main driver in the Symbian ecosystem. Fear of being thereby disadvantaged at the hands of a leading competitor may have led to Motorola's partial defection from that community.

This last example illustrates a final issue that arises with some regularity in the multisided businesses we've discussed. Before Motorola's defection, Nokia and Motorola collaborated in managing Symbian while competing to sell handsets. Similarly, between 1998 and 2003, Palm both collaborated with Handspring, to make sure the Palm OS and Handspring's hardware worked well together, and competed with it in the sales of integrated hardware-software systems. These are necessarily complex relationships, and keeping both collaborative and competitive dimensions healthy is not simple. It is often hard for individuals to both collaborate with each other and compete effectively against each other, so that these functions are often handled at the working level by different units within the organization. Top management, of course, cannot divide itself in this fashion.

Nokia faces these issues when licensing its Series 60 middleware platform to makers of Symbian-based handsets that compete against Nokia's own phones. As mentioned in Chapter 7, to alleviate licensee concerns, the company decided to raise a high internal Chinese wall between the Mobile software division, which is in charge of developing and licensing Series 60, and the hardware division. This sort of arrangement is not without its costs, of course, since there are efficiency gains from allowing hardware and software developers to communicate.

Merchants or Two-Sided Platforms

Software platforms have a choice between two models when it comes to the provision of applications, games, or content. The first is a multisided model: platforms provide support for interactions among the various customer groups supported. Each customer group needs access to the multisided platform to reach the other groups. The platform doesn't substitute itself for any customer group in these interactions. For example, it doesn't buy applications or games and resell them to end users. This multisided model is used for at least two customer groups by most software platforms we have encountered with the exception of the iPod/iTunes digital media platform.

To take one example, i-mode is a three-sided platform. It sells *access* to both users and content providers. It never takes ownership of content. Each content provider receives revenues directly from i-mode users, depending on how popular or appealing her content is. Each i-mode content provider therefore cares a great deal about how many people buy i-mode phones, because this determines the size of their target markets. And this implies that, in turn, each content provider cares about how many other content providers support i-mode and how good their content is as well, because the overall availability of content drives users to i-mode. Also, although DoCoMo does buy i-mode phones from contract manufacturers and resells them to end users, a significant fraction of i-mode phones are sold by manufacturers through other retail channels. This means that handset manufacturers need to be induced to produce and sell i-mode phones,[9] and each of them necessarily cares about the overall popularity of the i-mode system.

The second business model is a *merchant* or single-sided model. The platform buys the complementary products or services and resells them to users. It substitutes for the user when dealing with the maker of complementary products—that is, it buys and takes ownership of these products—or it makes these complementary products itself. Indirect network

9. As we have seen in Chapter 7, the inducement to produce is a particularly complex one, as DoCoMo works very closely with its associated phone manufacturers to help them include the specifications needed to take advantage of the features offered by the i-mode service.

effects are no less important for the merchant model than they are for the two-sided model. They are just managed directly by the platform owner rather than through the multisided strategies we discussed in Chapter 3.

iPod/iTunes is the only example of a pure merchant model we have seen among software platform–based businesses. Apple buys or makes all the pieces necessary for making the iPod (with the exception of a variety of peripheral equipment), and it in effect buys the music from publishers and owners on behalf of iTunes users. It has acted on behalf of consumers in negotiating directly with the music publishers and owners. At the end of 2005, the publishers were trying to persuade the digital music industry to adopt variable pricing that would charge more for hit songs than for older ones; Apple is vigorously defending its 99 cent price for all model—and is therefore reserving royalty fees that depend on popularity—on the grounds that this is the best model for the ecosystem.

Many software platforms, however, have adopted partial merchant models in the sense that they either integrate into a side or buy the complementary product or services on behalf of consumers. That is Apple's approach on the hardware side for its computers. It makes its own computers and either makes or buys some of the peripheral equipment that come with its computers. And it was Palm's approach as well initially; it made or bought all the relevant pieces for the Pilot.

What Exactly Does It Mean to Be Two-Sided?

According to the general definition we provided in Chapter 3, a platform is running a two(multi)-sided business model whenever it connects two (or more) groups of agents, each of which benefits from the participation of the other(s), *and* the platform provides the support for direct interaction between the two (or more) sides, without taking the place of any of these sides. Sony PlayStation is clearly multisided: the more independent game developers it signs up, the more consumers will buy it, and vice versa. And game developers sell their games directly to PlayStation users. The same holds for i-mode, Windows, and Palm.

As pointed out above, however, Apple's iPod platform functions as a one-sided business, although there are positive indirect network effects from its adoption by music publishers and users. To throw some light on

(continued)

what may sometimes appear as an obscure distinction, consider a retailer like Wal-Mart.

In the first scenario, Wal-Mart functions like your average village merchant who wakes up early in the morning to go acquire (many different types of) produce from suppliers, and then resells it to consumers at its local store. In this case the transactions Wal-Mart conducts with members of the two groups, suppliers and consumers, are largely independent of each other. First, Wal-Mart takes the place of buyers when dealing with suppliers, and then it substitutes itself for sellers when dealing with consumers. Suppliers could not care less about how many people visit Wal-Mart's stores, they are only interested in the price Wal-Mart bids for their products and the quantity it buys. Similarly, consumers care only about the prices at which Wal-Mart sells and the quality of the product it supplies, not the prices at which it buys.

Imagine now that Wal-Mart becomes more sophisticated and offers each supplier a contract specifying the price per unit of its product, a quantity it commits itself to buy, and also a price at which the supplier has to repurchase any unsold units. This type of contract is likely to improve efficiency by allowing Wal-Mart to order larger quantities upfront rather than restricting itself to a minimum in order to avoid accumulating unsold inventories. Short-run risk is now shared between Wal-Mart and its suppliers. At the same time, however, this contract also introduces indirect network effects. In deciding whether or not to accept such contracts, suppliers now have to take into account the visitor traffic Wal-Mart's stores generate, because it determines sales and ultimately their profits: more consumer traffic means fewer unsold items and therefore higher profits. And since consumers are clearly more likely to visit Wal-Mart if it offers a greater variety of products, each individual supplier ultimately cares about how many other suppliers (particularly of complements and substitutes) contract with Wal-Mart.

This example illustrates simply how even in the context of a one-sided merchant, two-sided indirect network effects may appear just by the nature of the contracts it writes with its suppliers. Nonetheless, the merchant's business is still not two-sided in any meaningful sense. Furthermore, this example is not directly relevant to software platforms, since "buying back unsold units" does not make any sense for digital applications, games, or content. Inventory is not an issue in the digital economy.[10]

There is, however, a way in which Wal-Mart's business can become clearly two-sided, which is also directly relevant to software platforms.

10. For a more detailed discussion of the scope of the platform, see Kevin Boudreau, "The Boundaries of the Platform: Vertical Integration and Economic Incentives in Mobile Computing" (working paper, MIT Sloan School of Management, Boston, 2005).

(continued)

Imagine that instead of buying all products and reselling them, thus effectively taking ownership, Wal-Mart rents shelf space to some suppliers, say to Kellogg for its cereals, to Coke for its cans, and to Sony for its electronic devices. The suppliers are responsible for supplying, displaying, pricing, and advertising their merchandise within the space allocated by Wal-Mart, and they receive the revenue from sales to consumers. Again, putting aside suppliers' costs of visiting multiple stores, this contract is likely to result in cost savings because it provides suppliers with both flexibility and incentives to use price, advertising, and display to maximize profits, and thus allows Wal-Mart to charge high rents. (On the other hand, some efficiency might be lost because suppliers have no incentive to take into account the effects of their in-store actions on other suppliers.) In this case, Kellogg, Coke, and Sony are more than a little interested in the traffic Wal-Mart generates, since this determines how many consumers are likely to stop by their stands and eventually buy their products. It makes sense to think of them as "on board" the Wal-Mart platform. Renting shelf space to third-party vendors in the material world is akin to being a portal in the digital world. Conversely, i-mode can be described as offering "virtual shelf space" to its content providers, which the latter can manage as they choose. As Wal-Mart surely would in this example, DoCoMo in fact reserves the right to pick and choose which firms are awarded the most prominent spaces (that is, are designated official providers), as well as how the entire space is managed.

Naturally, one can come up with many other contractual specifications that transform pure merchants partially or fully into two-sided platforms. These specifications may enhance efficiency by better aligning interests and incentives between the platform or merchant and its suppliers or complementors.

Patterns of Integration Over Time

At the most general level, the computer-based industries studied in this book have tended to become less integrated over time. The computer industry, for instance, started off with mainframe suppliers such as IBM providing fully integrated, stand-alone systems, including hardware, software platform, some applications, and peripheral equipment. An independent software industry emerged later, along with suppliers of peripheral equipment. Then, as computers became smaller and the workstation and PC revolutions unfolded, third-party suppliers of applications and peripheral equipment became more important. Some software

platform vendors, such as Apple and Sun, continued to provide both hardware and software platforms, while Microsoft produces operating systems and some applications but no computer hardware platforms and no major peripheral equipment.

A similar evolution (although on a much shorter time scale) has taken place in the PDA and smart phone industries: these devices have evolved from single-purpose electronics products supplied by individual manufacturers into small computers based on software platforms that support a variety of application vendors and hardware suppliers. Similarly, the video game industry has evolved from single-game systems such as Home Pong through multiple-game systems provided by the same manufacturer (Fairchild's Channel F, Atari's VCS 2600) and ultimately to video game consoles (PlayStation, Xbox) that integrate hardware and software and are supported by hundreds of third-party game developers and publishers, as well as middleware providers.

The most plausible industry-wide explanation for this trend is the development of competitive markets for system components, which depends on the emergence of both accepted standards that are relatively stable and platforms that are perceived as viable. The analysis of changes in integration over time goes back at least to Adam Smith, who asserted in *The Wealth of Nations* that "the division of labor is limited by the extent of the market." In a famous 1951 paper, Nobel Laureate George Stigler argued that this proposition implies that "vertical disintegration is the typical development in growing industries, vertical integration in declining industries."[11]

Stigler noted that at the inception of new industries, vertical integration is necessary because the technologies involved are unfamiliar. It is therefore hard for firms to persuade outsiders to participate in a business with uncertain prospects and with which they have had little or no experience. If and when the industry grows and becomes viable, many of the tasks involved in the production processes are sufficiently well defined and are performed on a sufficient scale to make it possible for an integrated early entrant to turn them over to specialized firms, either as suppliers or as complementors. It is also profitable to do so *provided*

11. George Stigler, "The division of labor is limited by the extent of the market," *Journal of Political Economy* 59 (June 1951): 185–193.

the market of specialists is sufficiently competitive, as we discussed above, and that generally depends on the industry being large enough to support multiple specialist firms. Disintegration frees previously integrated firms to concentrate on those parts of the final product on which they have a comparative advantage. They may become specialists, or system vendors that buy components from specialists and assemble them.

Naturally, the industries Stigler had in mind were traditional one-sided ones, such as cotton textile machinery: the system vendors just bought parts and sold final integrated products. However, his insights apply to the modern digital industries we discussed here, and with particular force in some respects. It is not just that software platform vendors *can* rely on specialist firms to provide complements as an alternative to buying them. Rather, it seems that they *must* rely on third parties: increasing technological complexity and consumer demand for more diverse and better products make it impossible for the same firm to innovate effectively throughout the entire system; enlisting the cooperative participation of outsiders via well-defined interfaces becomes a must. In the words of Takeshi Natsuno, i-mode's chief strategist, "given the complexity of today's IT businesses, one technology or one firm alone cannot lead a new service."[12]

As the applications software industry matured and became more competitive, platform software vendors could turn from writing applications for their platforms to managing relations with third-party suppliers. The development of the IBM PC and its clones allowed Microsoft to offer a successful software platform without getting into the hardware business. Disintegration of the video game business both required and enabled the emergence of a vibrant industry of independent game developers. When Palm had managed to establish itself as a viable platform, it could enable the creation of the "Palm economy" of third-party complementors.

The video game industry took at least an additional step further on the disintegration path than the other industries did. A second wave of disintegration followed the emergence of the third-party game providers. As consoles became more complex, resulting in a longer and more

12. Takeshi Natsuno, *i-mode Strategy* (New York: John Wiley & Sons, 2002), pp. 31–48. Interview with the author.

complex development process, the game development side of the market separated into game development studios and game publishers. As described in Chapter 5, the former do the actual coding, whereas the latter specialize in managing relationships with console vendors, providing initial financing for developers and marketing for their games, and managing financial risks by spreading investments over a broad range of games. We have also seen the gradual build-up of a third wave of disintegration with the appearance of pure middleware firms that specialize in providing development tools to game developers for specific console platforms. Their market opportunity was created mainly by the increasing complexity of 3D graphics programming and especially the rise of online games, which require sophisticated networking solutions. And, of course, consoles such as PlayStation and Xbox welcomed the appearance of these firms, as they lowered the entry costs and expertise required of game developers.

Digital media platforms represent an interesting partial exception to this general tendency. In adopting its integrated iPod/iTunes platform, Apple apparently believed, for example, that the relationships among the various sides—such as inducing music companies to license their songs for online distribution—were too delicate to leave to others. It also may have believed that Apple's organizational advantage comes from producing cool integrated solutions. Moreover, it may be naive to think that Apple, whose genes come from Steve Jobs, could run a software-centric ecosystem as it is to think that Microsoft, whose genes come from Bill Gates, could make fashion accessories.

To explore these trends in more detail, it is useful to consider integration between platform software and applications separately from integration with hardware and peripheral equipment.

Applications Software

Our overviews of the evolution of computer-based industries suggest that the main reason why software platform vendors integrate into applications is to overcome the difficulty of attracting multiple complementors to new platforms with uncertain market prospects—the so-called chicken-and-egg problem, with technical uncertainty and complexity on top. If a PC operating system, for instance, lacks enough attractive appli-

cations, users will not adopt it, and therefore independent developers will not have any reason to write applications for it. Similarly, nobody wants to buy a video game console for which there are no good games, and nobody wants to write video games for a console that nobody will buy. There may be other ways to make developers more optimistic, of course, and evangelization certainly has a role.

In fact, DoCoMo relied on a "pure evangelization" strategy before the launch of its i-mode platform and has followed the same pattern when introducing major new features. It initially solved the chicken-and-egg problem by enlisting sixty-seven providers of attractive and diverse content; it was perhaps able to do this because its large share of the gadget-loving Japanese mobile phone business made i-mode seem highly likely to succeed. DoCoMo has always made it a point neither to buy content nor to provide any by itself, in order to preserve the incentives of third-party content providers. The company thinks that if it entered the content business, third parties would be reluctant to participate for fear that any innovative service they might come up with would eventually be imitated and competed away by DoCoMo. Likewise, if it bought content, providers would have an incentive to write content likely to appeal to DoCoMo, not to i-mode users, and would stop development once they sold it.[13]

On the other hand, integration—and the actual production of applications—provides more credibility to a software platform with uncertain prospects than evangelization by itself. In combination with evangelization, integration amounts to putting the platform vendor's money where its mouth is. Thus, most of the platforms we have encountered in this book initially opted for an approach involving writing some attractive applications or good games internally, so that end users had a reason to buy the platform and independent software vendors had a concrete reason to bet on its viability. Naturally, if and when its viability is established, it often makes sense to focus development efforts on the platform itself and tools for developers, and to work to bring independent developers on board as participants in the platform's ecosystem. Even in mature systems, though, doing some applications in house may provide

13. Natsuno, *i-mode Strategy*, pp. 62–63. Interview with the author.

valuable information on the quality of its development tools (eating one's own dog food, so to speak) and be a useful device for showcasing new platform features.

Palm provides the best example of gradual disintegration as a strategy. As we saw in Chapter 6, Palm thought that, in the wake of the failure of previous PDAs, including its own Zoomer, application developers would be quite reluctant to develop applications in time for the launch of PalmPilot. Therefore Palm decided to write the core applications itself and bundle them with the device. When it had established a significant base of users it could turn its attention to getting independent software application developers on board and creating the Palm economy. Palm's emphasis on getting others to write applications is particularly noteworthy because Palm began as an application developer for small computing devices. Even when it developed the PalmPilot it contracted most of the work for pieces other than the applications out to others.

Another example is provided by the video game industry. When Sony entered with its highly successful PlayStation in 1994, the dominant players, Nintendo and Sega, each had more than half the games for their respective consoles developed in-house. In large part this stemmed from the fact that both companies had been major players in the arcade industry before entering the home video game console market and therefore had a lot of in-house game development talent and stocks of proven games. Before launching the PlayStation and the Xbox, Sony and Microsoft acquired several prominent game publishers in order to ensure the presence of a few good games at launch: Psygnosis (Lemmings) for Sony, Rare (Battletoads, Golden Eye 007) and Bungie (Halo) for Microsoft. But only one of the twenty-six games was developed in house when PlayStation 2 was released in 2000,[14] and both Sony and Microsoft relied primarily on effectively courting the independent game developers that had emerged since the industry's creation. After the launch of major consoles, the fraction of games by console vendors has generally remained small: the proportion of games

14. David Canter and Jeb Haught, "Fans of Sony Will Find PlayStation 2 Satisfies," *The San Diego Union-Tribune*, November 14, 2000.

available in North America and developed in house is about 10 percent for GameCube, 8 percent for PlayStation, and 8 percent for Xbox.[15]

In the PC world, Apple's early computers mainly ran Apple-developed applications, but there soon emerged a set of independent vendors—of which Microsoft was an important member—creating applications for the Apple II and then the Macintosh. By the late 1980s Apple had less than a 10 percent share of the market for Macintosh applications. Today Apple makes less than 1 percent of the applications available for Mac OS.[16]

Microsoft's role over time as a supplier of applications for PCs running its operating systems was more complex, both because of its history and because a vigorous applications software industry was already in place when the IBM PC was launched. As we noted in Chapter 4, Microsoft began as a developer of programming languages and tools for 8-bit computers and the CP/M operating system. While it was selling the DOS operating system for IBM PCs and their clones, it was also selling applications with GUIs for Apple computers. In 1988, seven years after the IBM PC was launched, independent software vendors provided the leading word processor (WordPerfect), spreadsheet (Lotus 1-2-3), and database (Ashton-Tate dBase) for the DOS software platform.[17] Microsoft accounted for only 18 percent of the sales of spreadsheets and 8.5 percent of the sales of word processors for DOS-based PCs. It played a larger role in applications for the Apple platform, accounting for 75 percent of Macintosh spreadsheet sales and 60 percent of the Macintosh

15. http://www.us.playstation.com/games.aspx (note that SCEA is a division of Sony); http://www.xbox.com/en-us/games/default; http://www.nintendo.com/games.

16. Of the 18,247 software products available for Mac OS X, Apple Computer created only 43 (http://guide.apple.com/action.lasso; advanced search of software performed December 29, 2005).

17. "Strategies for Microcomputers and Office Systems: PC Spreadsheet Software: Market Review and Forecast, 1988" (IDC report no. 4389), November 1999; "Word Processing Software, 1989" (IDC report no. 5019), December 1990; "PC Database Management Systems Software" (IDC report no. 4258), September 1989; "PC File Management Software: Market Review and Forecast, 1988" (IDC report no. 4413), November 1989.

word-processing sales.[18] With the rise of Windows 3.0 and its GUI and the appearance of processors capable of running graphics-intensive programs at acceptable speeds, Microsoft's experience in the Macintosh world became relevant to computers running its own operating systems. Its word processor (Word) and spreadsheet (Excel) quickly became category leaders, and the Office package built around them rapidly attained a similar status. By the late 1990s, Microsoft accounted for about 20 percent of the revenue earned by makers of Windows applications.[19]

Although integration into applications/games/content can help a software platform vendor deal with the chicken-and-egg problem of end-user and complementor skepticism at launch, it is neither necessary nor sufficient for the solution of this problem. In addition to i-mode, examples of apparently successful platforms that did not initially integrate into applications include Linux and Symbian. Likewise, the digital media platforms offered by Apple, Microsoft, and RealNetworks didn't integrate into the provision of content (although there was plenty of content available) or into applications that used the APIs provided in those platforms.

Most of these platforms had other sorts of integration that served to reduce initial doubts as to their viability. For example, in mobile phones Symbian had the backing of the major hardware manufacturers, and Windows CE was sponsored by a company that had already built up a reputation for persistence in the face of market apathy. Others had different strategies that didn't require the development of an ecosystem. Linux has developed mainly as an operating system used for specialized applications, such as doing the special effects for the film *Titanic*, operating the Google search engine, or running task-specific servers. Its similarity to UNIX and its open nature also made it relatively easy to port existing applications. RealNetworks started as an application itself that ran on top of Windows. For a while it was the only streaming audio game in town and could leverage that to attract content.

18. Ken Siegmann, "Microsoft's Dominance Continues: Resellers, Competitors Cite Concerns with Developer's Huge Share of Mac Market," *Macintosh News*, December 11, 1989.

19. "Worldwide Software Market Forecast Summary, 2000–2004," (IDC report no. 22766), August 2000. Calculation excluded "system software."

The history of media devices suggests that integration into content, for example, can even be harmful if incentives within the integrated enterprise are not in line. In this business, Sony trails Apple's iPod by three years and tens of millions of consumers. Before iPod's launch, Sony had the successful Walkman mobile media device and possessed an unmatched amount of content (Columbia, CBS Records, and MGM), whereas Apple was prominent in neither players nor content. But Sony's media player and music download service, designed to compete with iTunes, were significantly slowed by internal conflicts between its hardware and media divisions, the latter claiming that digital music players would increase the threat of piracy and therefore would significantly hurt its revenues. Meanwhile, Apple demonstrated that a well-designed software platform such as QuickTime/iTunes can be made attractive both to consumers, by allowing them to purchase music through the iTunes Web site, encrypt it in AAC or MP3 format, and transfer it to their iPod, and to music publishers, by managing digital rights effectively and thus providing a healthy source of revenues.

Interestingly, we are aware of no examples of software platforms that initially integrated into the applications/games/content that subsequently exited that business entirely. On the other hand, almost all such platforms have adopted a two-sided strategy and made significant investments in attracting third-party suppliers. Partial integration is the norm. The only exceptions are those successful software platform vendors that launched without integration; they have remained out of the applications business.

Hardware and Peripherals

The tendency of computer-based industries to disintegrate over time is even clearer—with interesting exceptions—when we consider integration with the supply of basic hardware and peripherals. In the early mainframe days, these were tightly linked to operating system software. Not only were there no specialist vendors to attract in these early days and no standardized hardware-software interfaces to provide to them, but different vendors made different technical choices, and there was necessarily great uncertainty regarding the viability of individual platforms. For instance, until 1998, when Apple introduced the iMac with USB

ports, Apple used nonstandard interfaces to connect peripherals such as printers, keyboards, and mice, and it bundled a computer mouse and display with those systems.[20] It continues to bundle hardware with its Mac OS.[21] Because Microsoft entered the platform business later and in partnership with IBM, whose viability was not in much doubt at that date, it has never been in the hardware business and is only a marginal supplier of peripherals.

Accidents of history and birth shape what firms are good at, their core competencies, and this has a long-lived effect on overall strategy and integration decisions. One can't presume that Apple could have created the rich ecosystem that Microsoft did, or that Microsoft could have created the style and veritable cultural icons that have helped Apple survive where so many others failed.

As we discussed in Chapter 4, Apple's strategy enables it to coordinate hardware and operating system changes internally, without intercorporate negotiation, and, because the Mac OS runs only on Apple hardware, to test all hardware/OS packages thoroughly before they are offered to end users. Microsoft's strategy requires it to invest heavily in working with hardware vendors and makes it impossible to test its software platforms with all hardware configurations end users will encounter. On the other hand, its hardware complementors have very strong incentives to produce quality machines at low cost, as both Dell shareholders and those who lost their investments in firms that used to compete with Dell can attest. One can imagine Apple's strategy winning in a world in which technological change drove frequent, radical changes in hardware and software architectures, so that the ability to manage those changes internally and produce more reliable systems was an

20. "Hands On–Mac–Universal Solution," *Personal Computer World*, February 1, 1999; "Mac Ports, Past and Present," http://charm.cs.uiuc.edu/users/olawlor/ret/mac_ports.

21. As we noted in Chapter 4, Apple briefly (from 1994 to 1996) pursued a policy of partial integration in hardware. But this pursuit was both brief and timid: Apple's licensees never accounted for more than 21 percent of the sales of computers running Apple operating systems. "Macintosh Clone," http://www.answers.com/topic/macintosh-clones; John Poultney, "Dataquest Posts 1997 Mac Tally," February 23, 1998.

enormous advantage, but that is not the world in which the computer industry has operated in the last few decades.

The Microsoft strategy of having the hardware complement its operating system produced by a competitive, technologically dynamic industry has served to make its operating systems more valuable and to speed their market penetration. Microsoft is not above using integration on occasion to stimulate important markets for complements, as its entry into mouse production, discussed earlier, illustrates.

Palm, discussed in Chapter 6, provides another interesting example of disintegration over time. It began as a developer of PDA applications that teamed up with others to create a PDA. It learned the downsides of lack of integration from that failed experience: it was hard to design a product by committee. So when it made another try at this business, it adopted a fully integrated strategy, not in the sense that it made everything but in the sense that it controlled all aspects of the software and hardware design process and treated other firms as subcontractors rather than partners. Following its success, it intentionally let this tight integration unravel. It started licensing its operating system to other hardware manufacturers—makers of PDAs, mobile phones, code bar readers, and other devices—in 2000 in order to increase its attractiveness to application developers. Eventually, Palm went all the way by disintegrating in 2003 into PalmSource, the software platform vendor, and PalmOne, the hardware manufacturer.

It did so mainly to liberate itself from the delicate conflicts of interest it was facing by licensing its operating system to PDA manufacturers that competed against Palm's own devices, the same problem Nokia is addressing by building a Chinese wall around its Mobile Software division. Disintegration allowed PalmSource to focus clearly on building a software platform that would appeal to a variety of device manufacturers.

Palm was responding to the competitive pressure of Microsoft's Windows CE, the software platform that is currently the biggest challenge for Palm OS in the PDA business and that relies entirely on third-party hardware producers. The now independent PalmSource would be in a better position to compete for the attention of those same hardware producers. In the words of David Nagel, PalmSource's CEO "As an

independent company, PalmSource can accelerate the acceptance of Palm OS."[22] In doing so, Palm explicitly stated that it did not want "to go the way of Apple and become a niche player."

Microsoft is pursuing the same strategy in smart phones, treating the hardware (handsets) as a commodity. The major producers of handsets have responded with Symbian, which can be viewed as an attempt to turn the operating system into a commodity. (This interpretation is reinforced by Symbian's design: hardware-specific middleware must be developed in order to make Symbian fully functional [with an end-user interface] with each manufacturer's handsets.) Although Symbian looks from some angles to be a nonintegrated software platform, its close links with Nokia have naturally made other handset producers nervous. Continuing this theme, i-mode can be viewed as an attempt, using Java middleware and other additions to its platform (e-payment in particular), to commoditize both handsets and operating systems to some extent; if end users and content providers can interact through i-mode interfaces regardless of their underlying hardware and software, then operating systems become less important. Thus, for instance, i-mode users can download Java applications through the i-appli service regardless of the particular handset model they have and the operating system it uses. And, as in other sectors, Linux is both a nonintegrated software platform and a nonstandard sort of competitor.

In addition to Apple in PCs, there are two other clear exceptions to the general pattern of disintegration of software platforms from hardware over time. The first of these is Apple's iPod media device, which integrates a hardware platform, software platform, and, through Apple's iTunes service, content. As noted earlier, it is hard to know whether this is no more than Apple's staying true to a corporate strategy that favors integration across the board or whether it reflects some nonobvious technical advantage to coupling hardware and software tightly—or even whether this tight integration is a permanent state of affairs.

The second clear exception is the video game industry. All successful providers of video game systems integrate hardware and platform

22. Palm, "PalmSource Spin-Off and Handspring Acquisition Approved by Stockholders," Palm press release, October 28, 2003.

software. We believe the explanation for this departure from the norm stems from special features of video game consoles. These are computers designed for only one type of application, interactive games in which graphics are typically very important. To provide the best possible gaming experience, sophisticated graphic designs must be rendered with high speed and great accuracy. Poor performance is visible, literally, to customers. To be competitive, system vendors have designed all platform components, hardware and software, to squeeze the best possible gaming performance out of the underlying microprocessor, and market leaders clearly have no interest in developing industry standard designs or interfaces. For example, the Emotion Engine processor at the core of PlayStation 2 and the entire architecture surrounding it were new, "the result of brilliant, out-of-the-box thinking."[23] It was built for one purpose: to generate amazing 3D graphics and digital sound. It was not as fast as Intel's Pentium II CPU for some operations, but, not surprisingly, its graphics processor had 1,000 times more bandwidth than PC graphics processors at the time of its introduction. Just as devotees of PC games have typically demanded the hottest PCs, so it seems that on average, video game systems need to be closer to the technological cutting edge (or, as is sometimes said, the bleeding edge) than PCs. This is just the sort of environment in which the flexibility in innovation provided by integrating hardware and platform software is most likely to pay off.

Still, it would be a great surprise if the enormous success of Microsoft's "commodity hardware" strategy in the PC business had not attracted imitators in the video game industry. This is at least one way to look at the short-lived 3DO Multiplayer discussed in Chapter 5. Trip Hawkins, its creator, had focused on specifying the best possible gaming technology and intended to license hardware production to multiple manufacturers—Panasonic, Sanyo, Creative Labs, and AT&T. As we noted, the other novelty in Hawkins's strategy was Multiplayer's pricing: the console was very expensive for end users ($699), while developers were charged remarkably low royalty rates ($3). It is hard to know which novelty was the main cause of failure, and, as we discussed in Chapter

23. Steven L. Kent, *The Ultimate History of Video Games* (Roseville, Calif.: Prima Publishing, 2001), p. 560.

5, the pricing strategy was clearly unsound. On the other hand, Michael Katz, former CEO of Sega of America, suggests that licensing hardware production was also unworkable in video games:

> I'd like to think that we would have had the smarts at Sega to market the 3DO in the conventional, successful way and not create this, if you'll forgive the expression, ridiculous new model that Trip came up with of how to market and sell 3DO. I mean, not manufacturing the hardware himself and licensing the technology to other people—it's ridiculous. Why would more than one company want to compete against someone else with exactly the same product? Why would a retailer want to buy the same product from more than one company? Everyone in the industry thought that was ludicrous.[24]

At first blush, Mr. Katz seems to be saying that the exact strategy that made Bill Gates the richest man in the world was "ludicrous." But the sectors are different. Third-party hardware production may work in the PC world because machines that are differentiated in buyers' eyes can still run Windows successfully. We talk of Wintel PCs as commodities, but this is only approximately true: the major manufacturers produce many models and configurations and advertise their superior features and functionalities loudly. This is perhaps even clearer in PDAs and smart phones, where different devices running the same applications on the same software program can be highly differentiated via differences in hardware design. The ability to differentiate and innovate in these sectors holds out the prospect of high profits. In the video game arena, however, it may be necessary for all vendors to produce "exactly the same product" in order to generate a satisfactory gaming experience. And, as Mr. Katz indicates, manufacturers would much rather differentiate their products, if only slightly, than produce a pure commodity.

Still, Microsoft itself, although it seems to have been fully aware of the failure of 3DO's Multiplayer eight years before, decided to try the same model again when it launched the Xbox in 2001. After all, if someone could successfully bring the PC platform model into the video game industry, who else better than Microsoft? But, as we saw in Chapter 5, Microsoft was quickly rebuffed by the third-party PC manufacturers it tried to enlist—quite probably a blessing in disguise. In a rephrasing

24. Kent, *The Ultimate History of Video Games*, p. 486.

of Mr. Katz's words, Michael Dell told Microsoft upon refusing the Xbox deal offered to him:

> When Sony cuts the prices on their PlayStations, their stock price goes up. Every time I cut prices, my stock price goes down. If you don't understand why that happens, you don't understand the console business. I understand why this is strategic to Microsoft. I don't understand why this is strategic to Dell.[25]

Competing Ecosystem Structures

In most of the preceding chapters, we have encountered examples of competing software platform vendors with very different scope and integration strategies and thus, typically, very different business models. Apple versus Microsoft in PCs is the most familiar example, and Microsoft versus Symbian versus i-mode is perhaps the most complex. Rivalry of this sort raises economic issues that do not arise in the traditional world of bricks, mortar, and widgets. How does competition *across* the various layers of the mobile phone industry (mobile network, handset hardware, handset operating system) differ from competition *within* the same layer?

The managerial and strategic issues this sort of competition poses are interesting and difficult. As PCs and video games compete to become dominant as home entertainment centers, should video game manufacturers encourage or discourage conversion of video game consoles into PCs? As PDAs and smart phones both seek to become the single indispensable device in every purse and pocket, what sorts of complementors should they seek to attract, or should they focus on integrated strategies? Should Apple's competitors emulate its one-sided model, or should they be confident that multisided strategies will ultimately win out in media devices, as they have elsewhere?

INSIGHTS

• Software platform vendors are participants in complex ecosystems that include end users as well as producers of complements (such as

25. Dean Takahashi, "Opening the Xbox," *Prima Lifestyles*, 2003, p. 168.

hardware devices, software applications, and content), and, in mobile phones, mobile operators.

- Platform vendors are more likely to find a multisided strategy most effective when technical interactions with complementary products are stable and well defined, and the markets for those complementary product markets are competitive. These conditions tend to be more prevalent in more mature markets.
- At the start, software platform vendors often also produce applications and hardware because third parties can't be attracted until a base of end users has been built. Over time, software platform vendors generally become less integrated, though partial integration into applications is very common.
- In a multisided strategy, the software platform mainly facilitates interactions between the sides of the platform (particularly applications vendors and end users). In a single-sided (or merchant) strategy, the platform either produces the complementary products itself or buys them and resells them to end users.
- Most software platform vendors adopt a multisided approach with respect to at least two sides, although they may take a merchant approach with other sides. The major exception is the iPod/iTunes platform, which operates as a merchant with regard to all sides: hardware, software, and content provided to its customers.

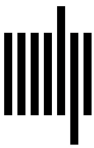
Invisible Engines

How Software Platforms Drive Innovation and Transform Industries

By: David S. Evans, Andrei Hagiu, Richard Schmalensee

Citation:

Invisible Engines: How Software Platforms Drive Innovation and Transform Industries
By: David S. Evans, Andrei Hagiu, Richard Schmalensee
DOI: 10.7551/mitpress/3959.001.0001
ISBN (electronic): 9780262272421
Publisher: The MIT Press
Published: 2008



The MIT Press

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu or write to Special Sales Department, The MIT Press, 55 Hayward Street, Cambridge, MA 02142.

This book was set in Sabon by SNP Best-set Typesetter Ltd., Hong Kong.
Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

Evans, David S. (David Sparks)
Invisible engines : how software platforms drive innovation and transform industries / David S. Evans, Andrei Hagiu, and Richard Schmalensee.
p. cm.
Includes bibliographical references and index.
ISBN 0-262-05085-4 (alk. paper)
1. Application program interfaces (Computer software). 2. Industries—Data processing. I. Hagiu, Andrei. II. Schmalensee, Richard. III. Title.
QA76.76.A63 E93 2006
005.3—dc22

2006046629

10 9 8 7 6 5 4 3 2 1