The Economics of Software Markets

HAL VARIAN AND CARL SHAPIRO

IT markets and software markets in particular are characterized by a number of features that render them unique in comparison to other more traditional product markets. Those features—namely complementarity of subsystems, significant transaction costs due to *lock-in* and *network effects*, incomplete contracts over future commitments by the vendor, and the trade-off between flexibility and costs—impact heavily on consumer and producer choices in a highly dynamic market context. We believe that it is of vital importance to settle for open standards, for they maximize both competition and choice within the market.

Keywords: Network economy · Switching cost · Open standards

1 Introduction

The presence of strong network effects in platform markets, and the interrelationship between the platform and applications markets, make software a particularly complicated industry. Many of the economic effects that shape the industry's development lie closer to the cutting edge of modern economic thinking than to the basic theories taught in freshman courses. See our book, "Information Rules: A Strategic Guide to the Network Economy" (Shapiro & Varian 1999), for a more complete discussion.

A number of these issues are particularly important: the complementarity among the components of an information system, the use of switching costs to lock in consumers and to guarantee revenue streams, the use of commitment as a negotiating tactic, the basic definition of network effects, the use of licensing terms to facilitate different business models, and the practice of bundling or integrating to increase revenues and alter market structures. The openness of source code can affect a number of these issues. While a review of these important aspects of network

Economic Effect	Definition	Implication
Complementarity	The value of an operating system depends on availability of applications.	Consider the entire system of needs before making choice.
Switching costs	The cost of switching any one component of an IT system can be very high.	Make choices that preserve your flexibility in the future.
Commitment	Vendors may promise flexibility or low prices in the future but not deliver.	Look for firm commitments from vendors, such as a commitment to open interfaces
Network effects	The value of an application or operating system may depend heavily on how many other users adopt it.	For a closed network of users, standardization within the network is more important than choosing an industry standard.
Licensing terms	A perpetual license involves a one-time payment; a subscription involves a yearly payment.	Licenses can be particularly pernicious when switching costs are high.
Bundling	Vendors will want to sell software in bundles to make future entry into the market difficult.	Purchasing a bundle now may reduce your future costs, but will also limit your flexibility and choices.

Table 1: Summary of Economic Effects and Their Implications

economics is beyond the scope of this paper, table 1 summarizes some of the key economic effects shaping software markets.

Policymakers need to appreciate that the decision to open source code is but part of a broader debate raging through the computer and software industries. Some in the industry have adopted the phrase *open computing* to describe an approach, applying to both hardware and software, that emphasizes modularity, interoperability, interconnectivity, and system flexibility. The key to open computing lies in open standards, including plug interfaces in hardware and application programming interfaces (*APIs*) in software. Important open source projects, such as Linux, embody all of these desiderata. Systems built around Linux are thus much better suited to the ideals of *open computing* than are systems built around platforms whose *APIs* are maintained as proprietary secrets. Many of the benefits that we attribute to open source software can be leveraged to even greater advantage when entire computing systems are open.

This is particularly true in considering the economics of openness. Because hardware and software, servers and desktops, platforms and applications, are all parts of a single computing environment, the economics of network effects ripple through the entire world of computing and software. For this reason, the effect of openness on industrial development are profound. Open standards and interoperability, in particular, tend to shift industrial focus from competition for the standard to competition within the standard. Immature industries may need time to experiment with different approaches before deciding upon a standard. Once competitors who began in different places converge, however, a standard exists-whether or not it has been "officially" recognized as such. If that standard remains the property of a single company, little competition may prevail. If, on the other hand, it is open to all industry participants, competition often remains fierce. Consumers and entrepreneurs tend to win, and rewards continue to flow to current innovators, rather than to those whose innovations proved successful during an earlier stage of industrial development.

2 Primer on Economics Concepts in the Software Sector

A decision-maker contemplating the adoption of a particular hardware or software platform must consider the entire information system. Hardware, software, personnel, training, system administration, and other components are all relevant to the adoption decision; looking at any one piece in isolation can be highly misleading. As a result, decisions about software adoption are more complicated than many other purchase decisions. Software adoption not only influences decisions about hardware, training, and personnel, but also implicates concepts related to "lifecycle costing" and "network economics." Different models of software development, and differing degrees of access to source code, can change many of the calculations that should go into a thoughtful software adoption decision.

2.1 Switching Costs and Lock-In

Complementarity implies that the components of an information system are interdependent. Any decision to change a single component is likely to require changing others, as well. New hardware may require a new operating system. A new operating system may motivate new applications. New applications may require retraining. And new server software may necessitate updated desktop application software. These cascading changes impose "switching costs." When switching costs are large, users may be locked in to their current information system, or at least some components of it.

Huge switching costs and user lock-in both arise quite often in the world of information systems. Indeed, in many cases, the total cost to an organization of switching information systems vastly exceeds the purchase price of the hardware and software. When the costs of switching to an alternative system are large, and when the user must rely on a single vendor to provide components of the incumbent system (such as software or hardware upgrades), users may be locked in to a single incumbent vendor, and thus vulnerable to that vendor's whims-and more importantly, to its policies concerning service, support, licensing, and pricing.

Many information technology vendors rely on switching costs as an important part of their business models. Once a user has chosen a particular database vendor or an operating system, it may be very costly to change. This switching cost puts them at the mercy of the vendor. Savvy buyers should look not only at the deal that is offered up front, but also over the whole life cycle of the product. If the costs of switching to an alternative in the future will be very large, the locked-in consumer will possess little bargaining power. Consumers should always expect prices to increase for any future information technology services not included in their initial purchase contracts.

Decisions about information system adoption thus require consumers to *look ahead and reason back*. Focusing only on the current situation can be quite misleading. Because information systems are long-term investments that can lock consumers into vendors for many years, up-front decisions that maximize future flexibility convey true value to consumers.

2.2 Commitment

There is a fundamental tension between buyers and sellers when switching costs are large: buyers want to maintain flexibility while sellers want to encourage lock-in. Vendors recognize the reluctance of buyers to lock themselves in to proprietary solutions and thus try to downplay the extent of the lock-in.

Open source alters the dynamics of these negotiations. It offers a way for sellers to commit not to exploit buyers after they have chosen an information system environment. If the source code for a software system is available, then users, perhaps aided by third parties, have the flexibility to maintain and to extend their own software investments. This ability allows users to adopt open source solutions with some degree of assurance that their switching costs will be relatively low. If they become unhappy with their current vendors, they can switch to others and have considerable control over their own switching costs. In short, low switching costs facilitate competition, thereby forcing vendors to stay on their toes and to provide good service after the initial sale is made. Customized software vendors have long used "source code escrow" to assure customers that they would not be stranded if the company went out of business. Open source is a much stronger assurance. It limits the extent of opportunistic behavior in the future and tends to produce a more competitive environment for vendors.

This effect, though, stems from an openness broader than simply open source code. Some parts of the industry use the phrase *open computing* to describe an approach, applying to both hardware and software, that emphasizes modularity, interoperability, interconnectivity, and system flexibility. The key to open computing lies in open standards, including plug interfaces in hardware and application programming interfaces (*APIs*) in software. Important open source projects, such as Linux, embody all of these desiderata; systems built around Linux are thus much easier to maintain as completely *open computing* systems than are those built around platforms with proprietary, closely-held interfaces. Many of the benefits that we attribute to open source software can be leveraged to even greater advantage when entire computing systems are open.

2.3 Network Effects and Positive Feedback

When the value of a product or service depends on how many other people adopt that product or service, economists say that there is a *network effect*. For example, the value of a fax machine depends on how many other fax machines there are. Similarly the value of an email account may depend on how many of your correspondents use email.

In some cases, the value of a product may depend on how popular some other product is. A DVD player, for example, becomes more valuable as more DVD disks become available to play on it. When networks effects operate through complements in this fashion, they are known as *indirect network effects*.

Computer software exhibits strong indirect network effects, since the value of an operating system depends, to some degree, on how many applications run on it. Similarly, the value of an application is enhanced if it runs on a popular operating system.

Such indirect network effects may be less important for a dedicated server—often what really matters is only whether a particular program, such as a Web server or database, runs on the server. Similarly, most users don't care what operating system is used in their cash register. But in other cases, a user might not know exactly what applications he or she wants when purchasing an operating system. In those cases, the system with the most available applications is attractive because it preserves options for the future; popular applications will be available, file exchanges will be easy, employees, customers, partners, or friends are likely to be familiar with the system, and so on. This inherent attractiveness born of sheer popularity means that the dominant operating system and dominant applications providers tend to have a large advantage compared to alternative providers, even when those alternatives are of similar quality.

2.4 Bundling

Software applications are often sold bundled together. *Microsoft Windows* itself consists of a large number of programs that work together; *Microsoft Office* involves several different *productivity tools* that interoperate. *Red Hat Linux*, a standard Linux distribution, involves hundreds of programs that all interoperate.

Bundling is an attractive policy for both vendors and buyers, though a specific bundle may serve the interests of only one party. Buyers may welcome a bundle because they can get a complete integrated package with some assurance that all the applications work together and that they will be able to satisfy their future needs with this one package. Sellers may offer bundles since bundles allow them to better meet buyers' needs and perhaps to extend sales from one software category to another. Someone may initially buy *Microsoft Office* because she wants *Microsoft Word.* Later on, when she needs spreadsheet capability, she will naturally turn to *Excel*, which she already owns, rather that considering or purchasing competitive spreadsheet offerings. When these effects are strong, it may be extremely difficult for standalone vendors of individual software components, such as spreadsheets, to compete.

If there are switching costs associated with each component of the bundle, the cost of switching bundles will have to be summed across the various components. Even when each individual component has a manageable switching cost, the total summed across bundles may be substantial, leading to lock-in. Also, bundling may discourage users from switching one component at a time as a migration strategy.

References

Shapiro, C. & Varian, H. (1999), *Information Rules: A Strategic Guide to the Network Economy*, Harvard Business School Press, Boston.