COPYRIGHT: A TOO STRONG PROTECTION POLICY FOR COMPUTER SOFTWARE?

Sustaining Long-term Monopoly Rents and Discriminatory Pricing in Computer Software

Prepared By

William V. Rapp

Center for International and Area Studies
Yale University
New Haven, CT 06520

203-432-9395 (Fax: 5963)
e-mail: william.rapp@yale.edu

May 1997
Introduction

Since the 1980s, US trade negotiators have aggressively pursued the use and the enforcement of copyright protection for computer software on a global basis. Recent confrontations with China are a particular testimony to this policy. However, relatively little investigation has been done on whether in fact copyright is an effective form of protection for computer software, including games. Historically, copyright is considered a relatively weak form of intellectual property since it protects only the expression of an idea rather than the idea itself. For this reason, it extends for a very long period of time (50 years plus the life of the author) and applies automatically on a global basis without need for registration. Conversely, patents, which protect the idea and are considered a very strong form of intellectual property, extend for a much shorter period (20 years) and must be applied for at some expense on a country by country basis.

This paper argues that in fact for computer software, particularly packaged software, copyright actually offers very strong and long term protection that extends monopoly rents and allows type two and three discriminatory pricing globally for an almost indefinite period. Thus, from a US strategic trade negotiating position copyright’s adoption and enforcement make sense. Whether this is an appropriate situation from a global welfare perspective is another issue and is not addressed in this paper.

The main reason that copyright has become a strong form of protection for computer software is because a particular expression or program can become an industry standard. Once an expression, i.e. a specific program, becomes the industry standard for an application or operating system, it cannot be copied, translated or used for the life of the author plus 50 years (50 years for companies) in any country who is a member of the Berne Treaty without the permission of the developer. The software producer in turn demands payment for its use, usually in the form of royalties or licensing fees. The force of this intellectual property protection has been further extended under the WTO, but its true impact can only be completely understood by examining the economics of software development and its use from the viewpoints of both the producer and the user.

Software and its use have many unique features which lead to this strong protection result for copyright. These include increasing returns to scale; substantial training time and
expense for developer, customer and user; aspects of a public good; very high development costs; very low reproduction costs; no real depreciation but rapid obsolescence; and close association with particular hardware technologies. The interplay of these factors in a given economic setting usually leads to substantial first mover advantages and the rapid creation of oligopolies, duopolies and even effective monopolies in certain market segments. Given copyright protection, these advantages in turn can generally be extended horizontally to other markets increasing the long term global rents flowing to the developer.

**The Economics of Software, Including Games**

The bulk of a software program’s cost is in the development stage rather than the physical manufacture and hence are fixed costs. After as long as a two or three year development period, actual production and distribution is cheap (Steinmueller 1993) and is getting cheaper with the introduction of CD-ROMs, high density tapes and downloading over the Internet. Indeed, as more of a product’s description and usage is contained within the program itself, there is even less need for expensive printed manuals and large physical packages. In effect, marginal costs are minimal. Unlike computer hardware, therefore, in software incremental improvements in manufacturing and process have little relevance to competitiveness. Instead, the key competitive variables are keeping development costs down and getting broad global distribution so as to reduce fixed costs per unit of sales.

For software, it is the size of the user base or cumulative units installed, not the cost of production, that drives a software developer’s cost structure. A larger user base can more than compensate on a per unit cost basis for higher development costs. These increasing returns to scale if combined with the increased utility of a large user base when the easy transfer of software based information across users is important means software shares many aspects of a public good with a strong tendency towards oligopoly or even monopoly in each software market segment. That is, software use by one person does not diminish its use by someone else (H. See 1992 and S. La Croix 1992) and like the telephone its utility and value may be enhanced by increased usage. So more users means greater ability to exchange programs, to interface through a network using standard protocols and a common programming language, and to hire workers already familiar
with the software. For example, leading word processing, spread sheet, netware, operating
system, tax, graphics, presentation or Internet search programs all benefit from ever larger
user bases which increase their utility and lower their costs. These forces further expand
the user base, again increasing utility and lowering costs. This beneficial cycle results in
rapid consolidation around the leading developers of a particular program with generally
stable market shares once growth slows and there is a large user base centered around
those leading developers.

In sum, this economic structure means costs per unit for a successful program
drop rapidly, establishing a large cost and utility barrier for competitors to overcome. On
top of this, since users must spend considerable time and expense learning to use and
support a new program, they are very reluctant to switch once they have made this
investment. There is thus a strong lock-in effect as well. Prices will trend towards
equilibrium where the cost of adding a customer (reproducing the program) equals the
increased positive utility from being able to share information. However, once a customer
becomes a user, any competitor has to overcome the cost to the user of switching through
lower pricing. This is difficult to do when marginal costs are so low and the leading
producer already has lower average costs due to its larger user base. Indeed, for a large
producer costs become relatively constant at a very low level once they have established a
large user base.

These dynamics also create substantial incentives for software companies to
merge to expand their total user base as well as for large developers to have software
programs which can manage and convert the systems of their largest competitors to
capture the higher utility externalities of increased user interchange (e.g. Microsoft Word
and WordPerfect or Excel and Lotus 1-2-3). Both these trends promote the industry’s
oligopolistic structure.

After initial development, most software program improvements are in turn
evolutionary rather than revolutionary. This is an integral part of the “upgrade” marketing
approach to the existing user base, with low incremental development costs per unit. Such
improvements usually revolve around constantly increasing computing power which is
driven by Moore’s Law where the density of semiconductors on a chip double every
eighteen months. Increased capacity allows for more features and greater speed. Since the existing programs do not depreciate and are doing their job, it is only by offering upgrades that the developer can get the existing user to upgrade. As the need, “utility”, of the program is lower for existing users, the price must also be lower compared to a new user. Usually when the developer improves the program to take advantage of greater computing power about 35% of the existing user base will upgrade. It is accomplished by insisting users who want to upgrade show they own the existing program to qualify for upgrade pricing with those owning the most recent program versions getting the biggest discounts. The developer is able to engage in this type three price discrimination because copyright protects the eighty percent of the program that does not change. So a competitor cannot offer an upgrade of their own the way auto parts producers can offer non-official replacement parts.

Globalization

Because translation is also protected, and it is cheaper to localize an existing program than to develop a new one, this high fixed low distribution cost structure promotes globalization based on localization and adaptation of existing programs as well. The expanded global user base then lowers the average unit cost on both a local and global basis. To facilitate this, most US packaged software companies have developed source codes that are compatible with foreign language translation to facilitate this process. These source codes are protected by copyright too.

Given this interplay between intellectual property rights, technological change and user motivation, copyright, while normally considered a weak form of intellectual property protection, is actually a good very way to protect software on a long term basis. This is because protection in global markets is automatic and for fifty years. With upgrades every two to three years, this protection will last forever. There are no expensive filing requirements on a country by country basis as in the case of patents. Yet, because it is difficult to change the expression of the programming idea without changing the external and internal benefits to the user, it means the infringer does not have ready access to the user base that drives the developer’s economics.
However, because US companies are the main beneficiaries of copyright protection abroad pressures or support for intellectual property (IP) enforcement there are generally weak except perhaps. Thus, software piracy remains a problem and it is part of the reason for US trade pressures. Still, piracy is a two-edged sword because it hurts foreign developers more by restricting their user base and potential cash flow whereas for the US developer with a large global user base, piracy has the effect of undermining potential or actual local competitors while expanding their potential global user base. Further, users gained through piracy may become customers through better IP enforcement in the future due to trade policy pressures from US. However, the foreign competitor can better afford to wait than their local counterpart. This tends to leave the field open to foreign firms with global products that can be adapted to the local market to use their low cost protected software products to enter and control the market for packaged software particularly in personal computers, effectively leveraging off their established user base and their position as the global industry standard in that segment.

They are sometimes helped in this process depending on the market by competition among the leading integrated computer systems producers because adapting US packaged software to their systems helps them to maintain both customers and sales. For this reason, the leading global competitors in emerging high growth markets, such as network servers and work stations are primarily US firms.

Because US firms are the global leaders in most software other than games, US firms control the migration of competitive advantage in programming and usage to both advanced countries and countries such as India and Russia, where they use low cost, highly trained talent. Educational and firm infrastructure also exert unusually strong influences on software development. In this case the US constantly mobile US labor market has promoted the strong development of the “advanced factors” of production such as software engineers and programming specialists needed to develop the industry or a new software developer such as Netscape rapidly. Therefore, the development and expansion in the US of the type of agglomerative structure which Porter notes is necessary to achieve and maintain competitiveness (1990) in an industry sector is present in software.

**Relationship of Software and Computer Hardware**
Due to the great progress in producing smaller and more powerful integrated circuits, the computer industry has produced smaller and more powerful computers. This has led to the development of new types of computers and in turn new computer software to run them. This represents a large source of market growth and as noted above has deeply influenced software’s technical evolution. The paper thus next explores the impact of this technical evolution in some greater depth with respect to the different software operating systems used by different computer hardware systems and their interplay with copyright protection. It also explores the differences among various types of software and how copyright protection has generally lead to a dominant system for their use which in turn has spawned a large number of compatible application programs that have reinforced the acceptance of the dominant operating system for each computer category, expanding that system’s user base, reducing its costs, and increasing its utility. In this way, certain software has been directly linked to the growth in the demand for certain hardware and related functional client requirements.

Packaged software producers spend a great deal of time and money developing specific software programs that can be easily reproduced and distributed. This is an attribute they share with the pharmaceutical industry. For these reasons, they are dependent on strong intellectual property protection to foster their development since otherwise they cannot insure a steady cash flow for upgrades and future projects. Further, large development costs mean the size of the potential and existing client user base has a greater impact on an firm’s cost structure than improved programming skills. The user base also creates external benefits to users needing to share information.

At the same time, software producers face many different constraints and economic issues than those their customers face. Initially producer economics and initiatives governed the development of the computer market and in turn software. However, this paradigm only supports development of a software industry when computer producers’ interest in selling their hardware depends on the development and support of the large common user base necessary to be competitive in packaged software. That is, the computer hardware producers need to help develop the economies of scale for standard software packages through the use of a standard operating system such as IBM
Yet, as customers have developed large installed bases and their own software systems and infrastructure, their own economic criteria have changed too. Compatibility, communication and interoperability among diverse operating systems and multiple hardware/software configurations have become more important as different functions and uses are best met through systems other than mainframes. Still, these installations need to be able to get and share information despite the historical development of specific but disparate hardware/software configurations, often pioneered by different manufacturers. This has fostered the development of quite distinct market segments based on these differences, with some competitive overlap as well as efforts by the leaders in each segment to develop standard communication protocols between segments.

Each hardware/software category has its own technology, competitive and economic forces. The technology trend toward more computing power and greater software capability in each segment in turn affects the usage of all computers, acting to dynamically segment the market relative to mainframes, network servers, workstations and PCs. But whereas integrated systems producers will try to optimize vertically through a family of machines and microprocessors (MPUs), customers are interested in horizontal compatibility and the contribution of each element to their total system. Software developers take their cue from whether they are more closely affiliated with customers or with large integrated systems producers.

The growth and type of software sold is different for each market segment: mainframes, network servers, work stations, PCs, and games with the older technology/machine type tending to be lower growth. The mainframe market is dominated by the established integrated systems producers such as Fujitsu and IBM with the IBM operating system still being the dominant software design. On going payments by Fujitsu and Hitachi as well as a recent case involving Oracle demonstrate the continuing rents due to copyright that IBM is able to collect on its 30-35 year operating systems which continue to drive most mainframes. Hewlett-Packard and Sun are the leading players in workstations. While fragmented, overall the system works and because of existing investments in hardware, software and training it is now extremely difficult
for any country to get a new operating system adopted by users, especially one such as TRON that works across segments.

Instead, the dominant producers in each computer/microprocessor/operating system segment have developed middleware which facilitates interoperability with other systems. That is, between the operating systems and the Application Program Interface (API) lies a range of middleware that includes the communications programs used to run networks. This technical and industry structure is maintained and further developed through cross licensing agreements which build on the fact that such middleware and the dominant operating systems are both copyrighted. Since their programming expressions cannot be changed in any significant way, i.e. to avoid copyright violation, without adversely affecting a user’s total system, the cross-licensing and OS/MW/API system works to perpetuate the fragmented oligopoly structure of each segment and the global dominance of particular firms in each segment.

Of course mainframe software developed first, then software for stand-alone minis and more recently for network servers (usually mini computers that manage the network using a communications management program), workstations and PCs. As a large but mature market, mainframe sales and software are growing much less rapidly than the market for network servers, work stations, and PCs and their related software. Further, mainframes are used more for distributed processing, data base management, security, large computational requirements and overall Electronic Data Processing (EDP) than for specific task-oriented activities which appear to be customers’ current interest as they try to put more computing power throughout their organizations and into the hands of the ultimate in-house user.

Within this industry structure, the application software developers, which logically represents the bulk of the software market (70-80%), pursue a hub and spoke strategy where they adapt their software to different operating systems to increase their user base and improve their competitiveness, which then allows them to pursue additional market share. This reinforces the powerful first mover advantages in this industry so fast follower strategies generally do not work well as a way to develop and grow. Even Microsoft has had trouble developing a share of the money management software market
from Intuit, which led to their attempt to buy the company that in turn was appropriately blocked by the government. Similarly, despite the difference in size, Netscape appears to be holding onto a dominant share of the Internet browser market.

The language programs used to develop application programs such as FORTRAN, COBOL or C++ vary by computer segment and their associated compiler programs are generally referred to as tools and represent another software market segment. While the market for middleware, tools, and application programs are also segmented, the specific dynamics of these subsegments go beyond the scope of this paper. However, they are affected by the same user base economics, institutional influences, copyright, cross-licensing and hardware technology that impact other larger market categories.

Customers interested in reducing their dependence on one supplier and moving to a multiple vendor system support this balkanized structure. Indeed, access to newer more open, flexible systems and technologies such as network servers, work stations and PCs is dependent on successfully integrating them into the user’s own organizational system. Successful packaged software developers have thus again been forced to develop a hub and spoke strategy whereby they adapt their software to the operating systems of the major systems producers, who in turn help them market the product to their large clients to help them retain and expand their own user base.

User base economics plays a large role in these outcomes. As already noted, software has very high development costs and the time to develop a new program or interface can be quite long while the cost of reproduction, packaging and distribution are quite low. This means that unit costs are extremely sensitive to the size of one’s user base. As the software supplier obtains multiple users, the costs per unit sold drop dramatically. The more expensive and complex the program, the more impact a larger user base can have in determining competitive advantage. Selling several copies and then subsequent upgrades to one large customer can be a very effective way to do this. This explains the large customer’s importance to the PC packaged software supplier and the PC packaged software market. This leads to type two discrimination on a volume basis. In the case of Windows “95”, for example, initial systems were sold individually for $99, upgrades for $49 but to large organizations for $20 per copy.
The typical success story of a US software firm is one that develops a new product and quickly achieves market dominance, often 70% or more. Once the customers have learned how to use the program and have installed it as part of an overall organizational system, the customer and their users will be very reluctant to switch. This creates a barrier to competitive entry. While this barrier can be overcome, as in the case of Microsoft’s success with its Word Processing program over WordPerfect or Excel over Lotus 1-2-3, it is usually done from a related user base (in Microsoft’s case MAC) and in a high growth market where new users can be added on the basis of price rather than by trying to take existing users away from a competitor. For this reason, US firms are constantly endeavoring to expand their user base on a global basis, using their cost advantage from the global market to enter each successive local market and then improving this advantage by capturing a large share of that market. Japan, as the second largest software market in the world, has become an important locus for implementing this strategy.

Since IBM was forced to unbundle its software in the 1960s, the global software market has evolved rapidly. As long as software continues to advance rapidly, US producers seem to have an inherent advantage. They can maintain this technical edge more easily in software than hardware because development is less capital intensive and the primary input, skilled people, is fully charged to current costs. Further, user base economics and frequent upgrades constantly favor the advanced country with a large user base, especially if this can be extended globally. This is because the cost of development of a new software product or upgrade is essentially a fixed cost and is very large; while variable costs are low. (Upgrades are of course cheaper than new products which gives the existing developer a big advantage.) Thus, the larger the user base, which is a function of the number and size of one’s customers, the smaller is the fixed cost per unit. Potential productivity differentials between countries are overwhelmed by the effects of user base differences (fixed costs divided by a large number of users). Users in this sense are the actual end users of a software program.

In the case of an operating system, it is the user of the machine; in the case of programming languages, it is the programmers; in the case of application programs, it is the person developing the program or using it to accomplish a specific task. Thus, within
a particular large customer, there will be several users depending on the type of software. However, once the users have a program that accomplishes their needs, and they have developed the skill set to use it or improve it, there is little incentive for them or their employer (customer) to switch to a competing system. For this reason, substantial penetration of the existing customer base has usually occurred only when new hardware and software were purchased to address a new problem the existing system could not accommodate. This solution often involves the development of middleware and cross-licensing of existing systems as described above. Copyright has played a critical role in supporting and perpetuating this structure because of its extended coverage and instant global applicability. That is system integration requires the permission of the existing developers as well as the developer of the new product or upgrade. It is also not surprising that the US has made copyright enforcement a key part of its trade negotiation strategy, reinforcing these industry and technical developments with political leverage since it US firms who are the primary beneficiaries.

The most successful US software developers have taken advantage of this situation and customer demand worldwide for system integration and interoperability to further improve their cost competitiveness by exploiting a hub and spoke strategy that maximizes their market penetration. They first develop a software package that can be sold globally and then localize it for a particular country. The localized package is subsequently adapted to each major mainframe and mini producers’ operating system, often with the producer’s assistance. Competition takes place within new but smaller growth segments at the hub, i.e. network servers, workstations and PCs. But there is little competition to shift core clients from their mainframe systems or operating platforms; so those market shares generally remain stable. IBM has been the prime beneficiary from this trend.

High growth segments are at the center of a hub and spoke diagram representing the industry structure while the stable mini and mainframe markets are represented along the outside of the wheel. Integrated systems producers’ differentiated operating systems represent the spokes along which the US producers market to the systems producers’
mainframe client base. The integrated systems producer’s strategy is to maintain their customer base by supplying hardware and software for all segments (mainframe, servers etc.) to existing customers even to the extent of helping to incorporate other software packages. In sum, each such producer and their operating system represent a spoke, with the mainframe system at the intersection of the spoke with the outside rim.

The independent packaged software developer’s strategy is to try to sell into as many of these client bases (spokes) as possible through differentiated adaptation, generally through writing specialized copyrighted middleware with the cooperation of the integrated systems developer through cross-licensing agreements. At the last stage, the software is integrated for the individual users. Writing this type of middleware, and working with clients to install it so that it really links the entire diverse system of software and various hardware together, has thus become a critical strategy factor in selling into a global multivendor multi-operating system environment. Since this middleware is protected by copyright, these solutions are then proprietary to the developer and help to protect their user base, their products and their software technology.

**ECONOMICS OF SOFTWARE**

As described above, the basic economics of software development and production are that development costs are high and fixed and reproduction and packaging costs are very low and incremental variable costs. There are two basic approaches to analyzing the cost of software development. One is a production model based on improving the efficiency and quality of producing a line of code. The second is dependent on the number of users and those users’ commitment to a particular system, program or software solution. The first might be described as a production based strategy whereas the latter as described above is a user based strategy. However, the production based strategy only makes sense for developing highly customized programs for very particular uses. It is the power of the latter approach that has driven the development of the US software industry and has resulted in copyright becoming a strong source of protection that has enabled firms to engage in both price discrimination and aggressive global rent seeking behavior.
Improvements generally come as upgrades of 20-25% of a previous program's functionality. Since the existing part of the program is protected by copyright, only the original developer can offer these upgrades. Alternatively, a totally new program would require expensive and time-consuming retraining of both users and producers. Thus, lack of support by users and producers for new systems performing the same function better is an important barrier to entry and means software innovation by new entrants takes place primarily in totally new programming and application areas rather than as improvements to existing products. The clear benefits to first movers reinforces this trend. This model of development differs markedly from the product cycle model seen in textiles, steel, ships, automobiles, television or semiconductors and favors environments that promote a stream of new product development rather than incremental or large improvements by new entrants.

The market and new technological developments are closely intertwined. As long as a product is successful, one can finance the next product or upgrade. However, if growth slows, it becomes more difficult, as Lotus discovered. So firms cut back on development, seemingly giving other competitors an opening. Further, it is usually when the market slows and matures that technology begins to stabilize, standards emerge, and it is easier to mobilize resources to catch up. But it also becomes more costly to unseat established leaders, and copyright protection insures they retain the market for their existing product even though upgrades are fewer.

Any trend towards semi-customization reinforces the importance of copyright and the protection offered the original developer as their programs are fully integrated into a customer’s overall system. The cost of redoing this integration once it has been done and is working is just too high. For example, based on interviews with Japanese customers, packaged software that has been localized, customized and debugged in many cases offers a reasonable alternative to full customization while retaining most of its benefits. Based on a set of interviews and questionnaires, while customization can triple the cost of installing a package as a semi-customized solution, full customization can run ten to fifteen times the cost of a package. Thus companies do not view the semi-customized solution as so expensive relative to developing a comprehensive new system from
scratch. However, replacing it with another semi-customized product later is expensive, while any upgrade must be supplied by the developer due to the copyright on the localized and already integrated product.

While initially localization presented some problems for US software developers, the process has now been refined through learning by experience and changes in the format of most Western operating systems and application packages to accommodate double byte systems (AEA July 1992). Such conversion can now be done relatively quickly and easily, especially if it is an upgrade of an existing program that has already been localized. Several companies such as Apple, Microsoft, and Hewlett-Packard have modified the operating systems and software tools they supply to developers globally to facilitate the conversion of application programs to non-Western languages. The time lag between launch dates of English versions of a new operating system or application program and its foreign language counterpart has therefore become shorter. In addition, since many new application programs are upgrades, the amount of new programming to be localized has been reduced, further facilitating the process.

These trends strengthen the advantage for foreign suppliers of packaged software that can be inexpensively localized and demonstrates the effective interaction between copyright protection and implementation of a hub and spoke strategy. To the extent these corporate systems are highly integrated with organizational structures, users have even more reluctance to shift software packages and providers. Customers are only concerned with their own businesses and industry competitors. They are interested in the contribution information systems can make to their competitiveness. For them software is an input, not an output. It is therefore the economics of their businesses rather than the economics of the software industry that ultimately determines large organizations’ demand for and usage of software (Baba et al 1995 and Rapp 1995) while user base economics will determine who supplies a particular package. In turn, those economics continue to stimulate large customers and developers to work together to achieve constantly improving integrated systems. Maintaining competitiveness in packaged software thus seems to depend on the continuation and promotion of certain management practices and institutional arrangements.
PROTECTION, HARDWARE, TECHNOLOGY AND MARKET SEGMENTS

Another factor favoring existing software developers is the strong relationship in the market between particular operating software and certain microprocessors (MPUs) or logic devices that drive computers. This is because US producers dominate the global market for microprocessors. Specific operating systems and related software have frequently been developed to work with certain MPUs, and software programmers are constantly developing software to take advantage of the next generation of more powerful, faster MPUs. There are strong market forces linking MPU producers, programmers, and users; they perpetuate the standards serving a given market segment through several generations of technical development. Customers do not want to abandon their software investment just because they buy new hardware, so new MPUs must be able to run the old software (operating systems, middleware and application programs) as well as the new. Similarly, software developers want to be able to use their existing or a closely related programming languages to upgrade operating systems, middleware and application programs as this increases productivity, improves quality and reduces development time. Then the new MPUs create demand for upgrades of the existing copyrighted programs.

Software developers will logically concentrate on developing new programs and upgrades for the most popular MPUs, which due to their high development costs are subject to the same user base (fixed cost) economics as packaged software. This has led to an interactive oligopolistic cycle, combined with the growth in personal computers, intellectual property rights, networking, and parallel processing, to make MPUs and their related software the defining factors shaping competition in specific software markets. The examples are for PCs: Intel and Motorola; for workstations: Sun and Hewlett-Packard; and for mainframes: IBM, DEC, Intel, and Motorola. This trend probably will not change in the foreseeable future; indeed, it seems to be getting stronger following the introduction of the Power PC and Pentium processors. That is, while MAC and IBM/PC based chips and software dominate PCs, UNIX based systems are preeminent in workstations. IBM compatible systems still lead in mainframes. With these multiple
linkages, there is no single MPU/software interaction upon which a potential follower can focus or can freely copy.

As noted above, an important part of software development is to maintain compatibility with existing programs and operating systems because users do not want to relearn or rewrite particular programs every time they purchase a new machine. Under these circumstances, the most important competitive factor for a software producer is its installed customer base and existing program design or library, combined with its ability to write new programs or upgrades for existing programs for new or improved hardware so that changes are relatively seamless and easy to managed from the users’ viewpoint. Apparently, maintaining this interface while offering many new features was one reason for the extended launch time of Windows “95” and now Windows “97-98”.

The cost of line code per user is the important variable rather than the cost per programmer. If a given amount of line code is spread over a larger number of customers, the cost per customer falls dramatically. When two programs with the same function and roughly the same number of lines of codes are in competition, the cost competitiveness will be directly tied to their respective installed user bases. The firm with the larger installed base can sell at a lower unit price, all other things being equal. Large up front development costs, low physical production and distribution costs, and customer inertia all favor this kind of competitive advantage. This explains why many packaged software firms rush new programs to market at very low prices to establish a large user base or offer abbreviated versions free over the Internet. It also illustrates how new niche companies can quickly establish dominance in a particular area. As explained above, it also makes acquisition of a firm with an existing program and user base very attractive as an entry or market share expansion strategy for another firm with greater distribution and financial power. Several large recent US software acquisitions reflect and confirm these cost structures.

Even outright copying or decompilation may do little to change the strong user base advantage of the original developer. It represents just another form of localization that siphons off programming resources from development to copying and localizing the copied product. This type of re-engineering results in "legalized dumping" of localized
software onto the market. De facto, it continues the original producer’s dominance of software and hinders the evolution of an independent packaged software development capability. It also makes the copying firms dependent on a continuous stream of such reengineering for upgrades and new products. This accentuates the existing global expansion trend as does the evolution of the US computer industry.

Labor force developments also support these structures. Since US EDP specialists in corporations often view themselves as computer specialists first and corporate executives second, they have looked to the computer industry to facilitate their mobility. When they have shifted from one firm to another, they have not wanted to learn a new set of technical skills in addition to learning about a new company. Further, the corporations hiring them wanted them to be productive quickly, favoring standardization in computer software. Universities that use computers and offer courses in computer science have been caught in the same push for standardization and career mobility. These institutional, economic and management forces have promoted further integration.

Given the technical expertise necessary to run large mainframe computers as technicians moved from customer to customer during the industry's early development, careers in the computer industry have been facilitated by customers and the lock-in effects to a particular standard technology and supplier as described above. Suppliers promoted this mobility, training and career development since it helped them secure their own markets longer term. At the same time, this pattern suggests why many corporate EDP departments were reluctant in the 1980s to embrace PCs until the introduction of networking. Many (MIS) Management Information Systems’ specialists saw the migration of computing power and expertise to operating personnel as a threat to their technical monopoly within the large corporation, and thus to their power and budget base. This anxiety continued until it was clear they were still needed to provide the extensive information sharing networks within and between organizations essential to access large mainframe data bases and to provide support to more and more complex PCs and workstations. This partly explains the origins of the CIO, Corporate Information Officer. In addition, the increase in computing power that made the PC revolution possible also made possible very large systems requiring very sophisticated expertise. In all countries,
there is a hard core of mission-critical and other large scale computing uses that require mainframe capabilities into the foreseeable future. Given IBM's dominance of this segment royalties flowing to their operating systems and its improvements should continue for sometime.

FUTURE EVOLUTION

That software must be learned on a particular hardware platform has increased the importance of an installed software and hardware base in attracting programmers to develop new software for that base. This has given significant first-mover advantages to US software producers, which they are extending. For example, American personal computers typically have either DOS or MAC-based operating systems (and Windows builds on DOS to make it look more like MAC, indicating format convergence at the user level). In turn, the most popular MPUs for PCs support these software platforms, e.g. Intel's 286, 386, 486, Pentium and compatible chips or Motorola's 8030, 8040 and Power PC chips. Users' compatibility demands have forced commonality, further helping Intel and Motorola, as computer manufacturers have demanded their chips or chips compatible with their designs.

This has created a self-reinforcing cycle as programmers lower their risk by developing programs for these platforms. Due to copyright protection, this situation has evolved much differently than for textiles, steel, automobiles, TVs, and DRAMs where one could freely enter an existing infrastructure and supply a product which can be used interchangeably with products of other vendors. The need to ensure compatibility has led to numerous alliances among oligopolists, where each firm accesses the other's strength in process and design.

THEORETICAL STRUCTURE

In developing a theoretical structure for this industry and the effect of copyright, the following points would seem to be the most important:

1) The demand function seems to have two parts which leads to discriminatory pricing. One is demand by users who are already using a particular software; they are influenced by their installed base plus price and the market share of the various suppliers. In the
other case, new buyers are only influenced by price and the market shares of the software producers. Market shares influence the buy decision according to a probability function. On the other hand, the supply function for each segment is primarily driven by the size of a firm’s user base, with global and local sales serving as a proxy for the size of the global and local user bases. Based on Japan (Rapp 1995) and its high cost, in foreign markets, the foreign supplier appears to face no more than 1, 2, 5 relationship between the cost of the foreign package, the localized package, and the semi-customized program. The last is used to arrive at a price to the customer and the relationship between a change in the user base and its price impact in the market.

An interesting aspect of this analysis is that the impact of a change in exchange rates acts powerfully at the package level but is heavily diluted as a cost factor within a semi-customized package. Experience curve effects can be factored in as a reduction in development costs as a particular program is improved or upgraded.

2) Software has aspects of a public good in that use by one person does not diminish its use by someone else. This combined with the low cost of reproduction means from a public viewpoint it should be free.

3) In addition, some software shares aspects of public goods like the telephone where increased usage actually enhances the utility and value to all. This is why telephone companies charge more for unlisted numbers.

4) The high cost of development and the low cost of reproduction means that the size of the user base not the cost of production drives the cost structure of a software developer. This fact combined with the increased utility to the user from a larger user base means there is a strong tendency towards oligopoly or even monopoly combined with larger user bases and falling prices by software market segment (e.g. word processing, spread sheet, netware, operating system, or graphics program). Prices will tend towards equilibrium where the cost of adding a user (reproducing the program) equals the increased utility to the user. The latter will become relatively constant once a large user base exists. There is also related incentives for mergers and for large developers to have their programs manage and handle the systems of their largest competitors.
5) This economic structure means the cost per unit of a successful program drops rapidly, establishing a large competitive barrier from a cost and user experience standpoint, short of actual copying.

6) After initial development, most software improvements are evolutionary rather than revolutionary. This facilitates the “upgrade” marketing approach to the existing user base in response to technological change given low incremental development costs per unit.

7) It also promotes globalization based on localization and adaptation of existing programs since the expanded user base lowers the average initial development cost and the average unit cost of upgrades on both a local and global basis.

8) Copyright is an excellent way to protect such software on a long term basis because protection in global markets is automatic and for fifty years. It is difficult to change the expression of the idea since this changes the external benefits to the user and its integration with the customer’s total system. This means potential infringers do not have access to the user base that drives its competitor’s economics. Therefore, practically and legally they cannot compete.

REFERENCES AND BIBLIOGRAPHY


