Computational Economics **6**: 201–217, 1993

Economic Incentives in Software Design

HAL R. VARIAN
Department of Economics, University of Michigan, Ann Arbor, MI 48109, U.S.A.

(Received: August 1992)

Abstract. I examine the incentives for software providers to design appropriate user interfaces. There are two sorts of costs involved when one uses software: the fixed cost of learning to use a piece of software and the variable cost of operating the software. I show that a monopoly provider of software generally invests the right amount of resources in making the software easy to learn, but too little in making it easy to operate. In some extreme cases a monopolist may even make the software *too* easy to learn.

Key words. Software, industrial organization, monopoly, quality, copying.

The market for computer software is large and rapidly growing. Despite this, there has been little theoretical investigation of the unique economic features of the software market. In this paper I investigate an important aspect of software economics: the extent to which the providers of software have the right incentives to design an appropriate interface for their software.

1. User costs

An important feature of software is that there are large costs to the consumer of using it. First, one must learn how to use a particular software package. Even if one only wants to use the package occasionally, one has to read the documentation, practice a bit, and invest time and energy in learning the basics of how to use the package. This cost of learning the software is a *fixed cost* to the user: it is more-or-less independent of the amount of use that the software gets.

This should be contrasted with the *variable cost* of operating a software package. These are costs that are incurred every time one uses the software. The most obvious of these costs are time costs, such as a delay in loading or saving a file. If it takes 10 seconds to start the package every time you use it, this is 10 seconds of lost time each time the package is used. If one has to wade through an elaborate menu structure to perform a simple task, then this is a cost that must be born every time the task is undertaken.

People who use a particular software package every day incur a large amount of these variable costs, while people who use this software rarely incur little variable costs. However, everyone incurs roughly the same fixed cost of learning the program.

Reviews of software often talk about "ease of use." The above distinction

suggests that there are two dimensions to ease of use: ease of learning and ease of operation. Software that is easy to learn has a lot of menus and elaborate help screens. It provides user prompts and error messages. The documentation is easy to read. Software that is easy to operate generally has fewer menus, replacing them with command key combinations. This means that a given command can be executed quite quickly—once the user has made the investment in learning the appropriate keystrokes. In this sense, the command driven interface is easy to operate, although the menu interface is easy to learn.[1]

Another aspect of ease-of-operation is *performance*: how quickly and how well the software does the job it is supposed to do. When we turn to modeling consumer choice of software we will be interested in the *net performance* of the software: the difference between the benefits from the task the software does and the costs of making the software perform that task.

These two aspects of ease-of-use—being easy-to-operate versus being easy-to-learn—are not mutually exclusive. A well-designed software package can satisfy both goals. However the software designer still has to decide how much effort to put into improving each aspect of the user interface. If the software is supposed to be delivered in one week, is that week better spent improving the speed of some calculation or fine-tuning the menu structure? The answer presumably depends on how improvements in these two dimensions affect profits . . . which is where the economic analysis comes in.

In terms of our previous discussion, the cost of learning a piece of software is a fixed cost, while the cost of using a piece of software is a variable cost. The software provider would like to minimize both sorts of user costs, in order to make the software more attractive to consumers, but it is costly to do so. The question of interest to an economist is whether the market provides the right incentives to the software provider. Will the provider of software invest the socially correct amount of resources in minimizing each type of user cost?

Although we have discussed user costs in terms of software design, it is clear that it also applies to other types of goods. Consider for example, sporting equipment. Equipment designed for casual users may be very different from equipment designed for intensive users. An easy-to-learn tennis racket may be loosely strung, while an easy-to-use racket may be tightly strung. Or consider other sorts of hobbyist equipment such as cameras. Again, a camera for a casual user has a very different design than a camera for a professional user.

2. Market structure

However, there is an important distinction between the market for sporting equipment and the market for software. The markets for tennis rackets and cameras seem to be reasonably competitive. There are a number of different types of products provided, and a given consumer can choose the type of product that is best for him. A casual user wants a product that is easy to learn; an

intensive user wants one that is easy to operate, and the market provides both types of products.

The market for software is a bit different. It appears that for some products at least, the market is very highly concentrated. For example, in the database market, Ashton-Tate has over 50 percent of the market. In the wordprocessor market, WordPerfect appears to have over 60 percent of the market, and Microsoft controls another 25 percent. Until recently Lotus had a 75 percent share of the spreadsheet market.[2]

At a more aggregated level the market for personal computer software seems to be becoming more concentrated. According to the 1990 SoftLetter 100 list, Microsoft had 25% of industry revenue and the next 3 companies (Lotus, Novel, and WordPerfect) had another 25%. The top 12 companies had 77% of industry revenue, as compared to 66% in 1985.[3] This concentration has not gone unnoticed in Washington; the FTC is currently examining Microsoft looking for evidence of unfair trade practices.

In addition, there are compelling theoretical reasons to believe that the software market is unlikely to be a perfectly competitive market, due to the presence of increasing returns to scale. In the production of software, nearly all of the costs are fixed costs—the costs involved in designing, writing, debugging, documenting and marketing the software. Furthermore, most of these fixed costs are sunk costs—they are not recoverable if the firm exits the industry. The variable costs—the costs of duplicating, packaging, and distributing the software—are very small by comparison. The fact that total costs are much larger than variable costs indicates that the likely equilibrium market structure will involve producers of mass-market software having a considerable amount of market power.

Another factor that suggests markets for software will be highly concentrated is the presence of network externalities among users. It is advantageous to me to have the same software as my colleagues since it makes it easier to share files, expertise, etc., and this tends to give the largest firm in the industry an advantage in selling more software.

Here we examine the admittedly extreme case of a monopolist. The case of monopolistic competition is certainly highly relevant, and I hope to examine it in future work, but it is beyond the scope of this paper. In any event, the phenomenon we examine here will apply in monopolistically competitive markets as well. Indeed, it will arise in any market where the producer has some degree of market power. The essential phenomenon we will discuss arises in any market where the producer faces a downward sloping demand curve for its product.

However, it is important for the following results that the manufacturer does not engage in product differentiation. That is, our model assumes that the producer sells only a single version of its product. This appears to be plausible in the case of software provision. There are some exceptions to this rule, such as WordPerfect and LetterPerfect, but these are rare. Generally, there is a single

version of the product which is sold to a whole spectrum of users. This is quite different from the behavior of a typical manufacturer of tennis rackets, skis, or cameras.

3. The basic idea

The basic idea that I want to capture can be stated quite simply. We may think of ease-of-learning and ease-of-operation as two different dimensions of software "quality." It is well-known from the work of Spence (1975) that a monopolist does not in general have the right incentives to provide the appropriate amount of quality. Roughly speaking, the monopolist is interested in how a change in quality affects the willingness-to-pay of the marginal consumer, while the willingness-to-pay of the average consumer is the appropriate concern for social welfare.

Consider a monopolist contemplating investing an additional dollar in software design. Should the dollar go to making the software easier to operate or easier to learn? If the monopolist makes the software easier to learn then he will acquire additional customers—those consumers who previously weren't willing to invest in learning how to use the product, but now find the investment worthwhile. If the monopolist makes the software easier to operate, he will also acquire *some* new customers . . . but most of the benefits of the improvements in ease of operation accrue to the people who would have bought the software anyway. Since the monopolist cannot capture the full marginal benefits from making the product easier to operate, it will, in general, underinvest in this aspect of software design. However, since the monopolist can expand its market by making the software easy to learn, it will have the correct social incentives in this dimension.

4. Design of a word processor

Take, for example, the design of a word processor. Some users may have need for a word processor only once a week. Whether or not they buy a word processor depends on how difficult they think that it will be to learn to use it effectively. How quickly it reformats paragraphs or spell-checks is not of great significance to them. However, these features could be very important to a person who uses the word processor everyday. For an intensive user, the learning costs are small relative to the costs of operation; for a casual user, the learning costs are the dominant consideration.

Ideally, there would be "friendly" wordprocessors for casual users and "powerful" wordprocessors for intensive users. But if there is only one wordprocessor for both casual and intensive users there is an inevitable tradeoff in the design of such software. Should there be a special command to transpose two words? This could be useful to an intensive user, but probably not very useful to an occasional user. Providing and documenting such a command is costly to the

software developer. In at least the first release of the software the developer would probably concentrate more on the quality of the documentation and user interface rather than investing much time in adding rarely-used features. Similarly, the casual user would probably not be too concerned with how rapidly the document could be reformatted, or how quickly the spell-checker worked. But these factors could be very important to an intensive user.

People who use software on an occasional basis don't want a lot of choices—they are willing to give up some features in order to make the software easy to learn. People who use the software intensively are willing to invest in learning a variety of features since they will probably find occasion to use them.

Another important example of this distinction is in user support. "Handholding" support is critical to casual users, but not nearly so important to intensive users. In fact, intensive users would probably prefer to see a company devote more of its resources to improving software performance rather than providing increased handholding for new users. But it's new users who bring in the new dollars—and that's why software companies invest in activities that can reduce the costs facing new users.

When we look at the evolution of personal computer software, we see improvements in both the ease-of-learning and the ease-of-use of software. The current behavior of software providers seems to be much more focussed on the ease-of-learning aspect of software design. In order to sell software to consumers who don't have it already, they have to make the software easier to learn. The attraction of user-friendly shells, such as MS-Windows, to software developers lies in the fact that once users have mastered the shell environment, the fixed costs of learning a new piece of software are much smaller for them. Hence one can expect that the demand for software products will increase.

Contrast the reaction of software vendors to MS-Windows with the reaction, say, to an increase in CPU speed. This might make getting things done a lot easier for intensive users of some software packages, but it probably wouldn't sell much new software. Of course, developers might redesign their software to add help features that were not feasible before—but that simply shows what is important to the software producers.

5. Monopoly provision of quality

The producer of a product chooses both the price and the characteristics of the product he produces. Most work in economics is concerned with the pricing decision. However, considerations of product design are also of great importance. Spence (1975) and Sheshinski (1976) consider the incentives facing a monopolist in choosing the "quality" of its product. Here quality should be thought of as a variable that shifts the demand curve for the product; in our application, "quality" is the ease-of-learning and the ease-of-operation described above.[4]

Spence (1975) computes the derivative of consumers' surplus minus costs

evaluated at the monopoly position and derives two conditions sufficient to sign this derivative. The first involves comparing the impact of a quality change on the marginal versus the average consumer: if the average consumer values the change in quality more than the marginal consumer then the monopolist underprovides quality. The second involves examining the sign of $\partial^2 p(x, y)/\partial x\, \partial q$. These conditions provide an answer to the question, but it is hard to interpret precisely what they mean. As Schmalensee (1979) puts it: "It is very hard to form any general intuition about the sign (let alone the magnitude) of the crucial cross-derivative P_{QX}."

In order to determine the sign it is helpful to develop a microeconomic model of consumer choice, a task I pursue below. However, before doing that it may be useful to derive the Spence–Sheshinski result. The derivation below is different from the method used by Spence and Sheshinski and has the advantage that it focuses attention on the crucial aspect of the problem relevant to the case at hand.

Let x denote the quantity and q the quality of some product. Let $u(x, q)$ be the utility of the product and $c(x, q)$ be the cost of providing it. Let $p(x, q) = \partial u(x, q)/\partial x$ be the inverse demand curve for the product. The social objective function is defined to be

$$W(x, q) = u(x, q) - c(x, q)$$

which is simply benefits minus costs. The monopolist's objective function is given by profit:

$$P(x, q) = p(x, q)x - c(x, q)\,.$$

Let (x^m, q^m) denote the monopolist's profit-maximizing choice of output and quality. We are interested in the derivative of welfare evaluated at the monopolist's choice.

Write the welfare function as

$$W(x, q) = [u(x, q) - p(x, q)x] + [p(x, q)x - c(x, q)] = CS + PS\,.$$

This is simply the sum of consumer surplus plus producer surplus. If we differentiate with respect to q and evaluate the derivative at the monopolist's optimum, we see that the derivative of producer surplus must be zero—since the monopolist is already maximizing profits. Hence the derivative of welfare with respect to quality is simply the derivative of consumers' surplus with respect to quality. This is a significant simplification since it means that we don't have to model the cost side of things at all.[5]

How does consumer surplus change as quality changes? This derivative is given by

$$\frac{\partial W(x_m, q_m)}{\partial q} = \frac{\partial u(x_m, q_m)}{\partial q} - \frac{\partial p(x_m, q_m)}{\partial q} x_m\,.$$

We can write this expression as

$$\frac{\partial W(x_m, q_m)}{\partial q} = x_m \frac{\partial}{\partial q}\left[\frac{u(x_m, q_m)}{x_m} - p(x_m, q_m)\right].$$

Hence the sign of the derivative of the welfare change is just the sign of the term in brackets. The first term in the brackets is the total willingness-to-pay divided by the number of consumers who purchase the good; this is the average willingness-to-pay. The second term in the expression is the price—the marginal willingness-to-pay. The welfare effect of the quality change depends on how a quality change affects the *difference* between these two terms. This proves the first of Spence's observations.

Note that no calculations are necessary; all that is required is the observation that the derivative of profit is zero at the monopoly solution and the observation that consumers' surplus is proportional to the difference between an average and a marginal quantity. We now go on to ask what it is about demand that determines the sign of this quantity. In other words, how does consumers' surplus change as the demand curve moves? As shown in Figure 1, we can decompose a movement of the curve into a parallel "shift" and a "tilt." The shift doesn't change consumers' surplus at all; only the tilt matters. It is easy to see that if the demand curve gets flatter consumers' surplus decreases and if it gets steeper, consumers' surplus increases.[6]

Thus what matters is how a change in quality affects the *slope* of the demand curve; this is given by

$$\frac{\partial}{\partial q}\frac{\partial p}{\partial x} = \frac{\partial^2 p(x, q)}{\partial q \partial x}$$

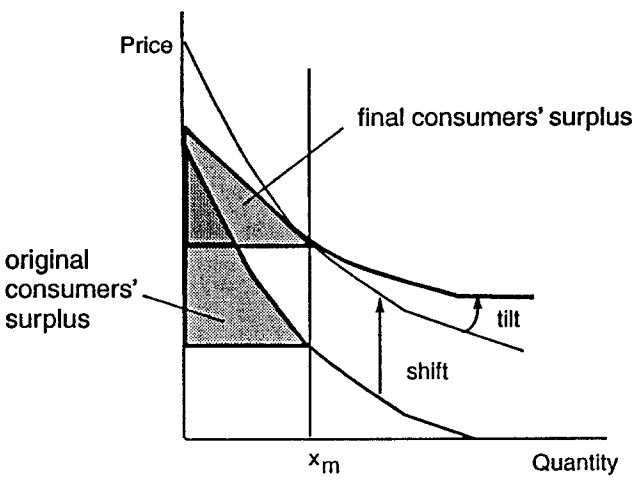

Fig. 1. Decomposing the change in demand into a shift and a tilt.

This is, of course, simply the Spence–Sheshinski condition. However, the interpretation in terms of shifts and tilts turns out to be quite useful below.

We can also relate this back to the earlier discussion of marginal versus average valuations. Changes in quality that shift the demand curve have no effect on welfare since they don't affect the difference between the average and marginal valuation. To affect the average and marginal consumer differently, the change in quality must affect the *slope* of the demand curve.

It follows that to answer the question of how a change in quality affects welfare, we need to construct a micro-model of consumer behavior and see how the quality variable enters the demand curve. Quality variables that shift the demand curve have no effect on welfare; variables that tilt the demand curve increase or decrease welfare depending on which way they tilt the demand curve.

6. The model

I now present a formal model of ease of learning versus ease of operation. I model the user costs in the following way. I suppose that there are a number of different users, each of whom uses the software more or less intensively. Let n be the number of times that a consumer uses a piece of software in some given time period, and let $g(n)$ be the number of consumers who use the software this often. For simplicity we take the frequency-of-use of the software to be independent of the ease-of-operation, although this can be relaxed.

Each time the software is run, the user bears a cost v. This is a variable cost of operation: it could refer to the time it takes to run the program, the complexity of the keystrokes necessary to run it, etc. High-intensity users—those who use the software a lot—pay a high variable cost.

Let F be the fixed cost of running the program. This is the cost that the user must pay regardless of his intensity of use. If she runs the program once or a hundred times, she must pay the same cost F. This should be thought of as the cost of learning to use the program. A program that is easy to learn has a low value of F; a program that is easy to operate has a low value of v.

Let $c(x, v, F)$ be the cost of the manufacturer of selling x copies of a program that has user costs of (v, F). For simplicity, we will suppose that the cost function has the separable form $c(x, v, F) = c_x(x) + c_v(v) + c_F(F)$. The term $c_x(x)$ measures the cost of producing x units of the software. The term $c_v(v)$ is the cost of designing software with variable costs v. The term $c_F(F)$ is the cost of designing software with fixed user costs F. This separable structure is not necessary for most of the results, but it makes the analysis simpler.

For simplicity we assume that the marginal costs of production are constant, and set $c_x(x) = c_x x + K$. Here c_x is the marginal cost of producing an extra copy of the software, once it has been created, and K is the fixed cost of producing the software. We should think of the fixed costs as being large relative to the variable

costs of production. Note that the cost functions c_v and c_F should be *decreasing* functions of their argument since it should cost more to make a package with smaller user costs. It is natural to assume that both of these functions are convex, since the marginal cost of improving a package should increase the better the package is to start with.

Let b be the gross benefit to the user each time he or she uses the program. If a user runs the software n times, the net benefit accruing to the user is then $(b - v)n - F$. This is the gross benefit per use minus the user costs. If the package sells for a price of p, then a person who uses the software n times has a consumer surplus of $(b - v)n - F - p$.

The benefit, b, measures the *performance* of the software. In our formulation, all that matters to the consumer is the difference between the performance, b, and the ease-of-operation, v. We might think of this as the *net performance* of the software: the net benefit of the software per use. In general b is a choice variable—the producer can invest more or less effort in order to increase b. But since all that matters to the consumer is $b - v$, an increase b is equivalent to a decrease in v. Hence there is no need to carry out a separate analysis of the choice of b.

We suppose that a person who has positive consumer surplus will purchase the product, and a person who has negative consumer surplus will not. The marginal user will be the person who has a net surplus of zero. If n^* is the intensity of use by this consumer, then it must satisfy the equation

$$(b - v)n^* - F - p = 0\,,$$

which implies

$$n^* = \frac{F + p}{b - v}\,. \tag{1}$$

This gives us a relationship between the characteristics of the software, (v, F, p), and the number of *uses*. We want to convert this into a relationship between the price and the number of *users* in order to determine how many consumers will buy the product.

Let $G(n)$ be defined by

$$G(n) = \int_n^\infty g(t)\,\mathrm{d}t\,.$$

This measures the number of people who use the software at least n times per period. If the number of uses by the marginal user is n^*, as defined in Equation (1), and there are x users in total, x must satisfy the equation

$$G(n^*) = x\,. \tag{2}$$

Let $H(x)$ be the inverse function of $G(n)$. The function $H(x)$ measures the number

of uses by the marginal person if x units are sold. Applying H to both sides of (2) and using (1), we can write

$$n^* = H(x) = \frac{F+p}{b-v}.$$

Solving for p as a function of x we have the inverse demand function

$$p(x) = (b-v)H(x) - F.$$

Since $G(n)$ is a monotonic decreasing function, so is its inverse, $H(x)$. Hence the inverse demand function is a decreasing function of price. Note that the variable cost affects the *slope* of the demand curve, while the fixed user cost merely *shifts* the demand curve.

This is quite reasonable. If the a software package becomes easier to operate, then all users are willing to pay more for it. But the high intensity users' willingness-to-pay goes up by more than the other users, since they use it more often. On the other hand, if the software becomes easier to learn, then everyone will be willing to pay more for it, regardless of their intensity of use.

We are now in a position to apply the preceding analysis concerning the welfare effect of changing v and F. However, it is useful to spell out the welfare analysis in slightly more detail.

In order to do this we first derive an expression for consumers' surplus. If x users buy the software the gross surplus (the area under the demand curve) is:

$$u(x) = \int_0^x p(t)\,dt = \int_0^x [(b-v)H(t) - F]\,dt.$$

If each package is sold at a price of $p(x)$, the net consumers' surplus is

$$\begin{aligned} u(x) - p(x)x &= \int_0^x [(b-v)H(t) - F]\,dt - [(b-v)H(x) - F]x \\ &= (b-v)\left[\int_0^x H(t)\,dt - H(x)\right]. \end{aligned} \tag{3}$$

Note that F drops out of this expression; it follows immediately that the derivative of consumers' surplus with respect to F is zero. Furthermore, since $H(x)$ is a decreasing function, it is easy to see that the expression in brackets is positive. A reduction in v helps the average consumer more than the marginal consumer since the average consumer uses the software more intensively than the marginal consumer.

The monopolist has the correct incentives with respect to ease-of-learning, but the wrong incentives with respect to ease-of-operation. Why? Essentially, the reason is the standard monopoly distortion pointed out by Spence (1975): the monopolist cares about the marginal consumer, not the average consumer. In our framework, the marginal consumer values ease-of-learning in exactly the same

way as the average consumer; hence, there is no distortion in this aspect of the product design. But the marginal consumer in our model uses the product less intensively than the average consumer, hence the monopolist has too little incentive to invest in reducing this sort of user costs. From the monopolist's point of view, the high-intensity user will buy the product anyway, and the monopolist has no incentive to make the product easier to operate for them. But the monopolist has just the right incentive to make the product easy to learn, since this increases the size of its market, and makes *all* users willing to pay more for the product.

The problem discussed above arises due to the fact that in our model the monopolist doesn't have a way to extract any payment from the inframarginal users, even though they would be willing to pay for improvements in ease-of-operation. In real life, the monopolist does have such an option: it can offer software upgrades. Intensive users will be willing to pay for those upgrades if they offer improved capabilities.[7]

However, typically a new release of the software is sold to both new and existing customers. When trading off investment in ease-of-use and ease-of-learning the software producer will still face the incentives described above: it will be willing to invest less in features valued by consumers who are sure to buy the product anyway. In any event, the fact that software can be upgraded is unique among products and is worth examining in its own right.[8]

The distortion in this model depends on the fact that the consumer cares about the number of *uses* while the monopolist cares about the number of *users*. If the monopolist could charge a price per use, there would be no distortion. To see this, imagine that the software is run on a mainframe computer so that the software provider can monitor the number of uses. The monopolist sets a schedule $\pi(n)$ that indicates the charge per use. The price schedule is given by

$$\pi(n) = \begin{cases} \infty & \text{if } n \leq F/(b-v) \\ (b-v)n + F & \text{if } n > F/(b-v)\,. \end{cases}$$

It is easy to check that this price schedule extracts all the consumers' surplus from the users of the software. Hence the monopolist will choose the socially optimal levels of F and v.

7. Software that is too easy to learn

In the above analysis we've seen an example where the product of the monopolist has too little quality (too hard to operate) and just the right amount of quality (appropriate ease of learning). It would be nice to complement this with an example where a monopolist provides too *much* quality.

In order to do this, let us change the model slightly. Suppose now that there is no difference in intensity of use among customers. For simplicity suppose that all

consumers use the program only once, and each gets the *same* net benefit $b - v$.[9] However, users differ in how difficult it is to learn to use a new program. To be specific, the net surplus from use of the computer program is

$$b - v - \gamma F - p .$$

Here F is a measure of the how easy the software is to learn—the fixed costs—and γ measures the capability of a given individual to learn the software. People with high values of γ find it more costly to learn a new piece of software than individuals with low values of γ. We suppose that γ is distributed in the population according to some cumulative distribution function $J(\gamma) = \int_0^\gamma j(t)\, dt$.

The marginal purchases of the program satisfies the condition that benefits are just equal to the price of the software

$$b - v - \gamma F - p = 0 ,$$

so

$$\gamma^* = \frac{b - v - p}{F} .$$

Everyone with a smaller γ buys the software, so the total sales are

$$x = J(\gamma) = J((b - v - p)/F) .$$

Letting K be the inverse of J, we have

$$K(x) = \frac{b - v - p}{F} ,$$

which implies that the inverse demand function is

$$p = b - v - FK(x) .$$

Note that $K'(x) > 0$ since it is the inverse of a cumulative distribution function.

For this form of demand, changes in v shift the demand function and changes in F tilt the demand function. According to our previous analysis, the monopolist produces the right ease-of-operation, but the wrong ease-of-learning. In fact $\partial^2 p / \partial x \partial F < 0$. From our previous analysis, this implies that welfare increases if F increases – that is, welfare goes up if the software is made harder to learn! In this model the monopolist *overinvests* in making the software easy to learn.

Why is this? In this model the marginal consumer is one who finds the software harder to learn than the average consumer. Hence making the software a little easier to learn benefits the marginal consumer more than the average consumer. Hence the monopolist tends to invest too many resources in attracting marginal consumers rather than, say, improving the functioning of the program for the inframarginal consumers.

8. Policy implications

What are the implications of the ease-of-operation/ease-of-learning distortion from a policy perspective? Obviously it is premature to draw definitive conclusions from such a simple model, but it is worthwhile raising the question to see where a more in-depth analysis may lead.

Since we are examining a monopolist, it is always in the social interest to increase output. One possibility would be to pursue antitrust actions to eliminate the monopoly power. However, it is far from clear that this should be appropriate since it would affect incentives to innovate and perhaps lead to excessive product differentiation.

Accordingly, we adopt the viewpoint that the monopoly output distortion should be tolerated. However, the above arguments suggest that even if the monopoly output remains constant there still will be social benefits to encouraging changes in the monopolist's provision of "quality."

In different models of quality, different tools may be appropriate. For example, in some models, setting minimal (or maximal!) quality standards may be appropriate. However, this instrument seems implausible in our model of software.

One interesting policy choice is to subsidize the provision of software "quality." In practice this should be done by publicly sponsored research grants. We suppose that social policies such as this will reduce the cost of providing software that is easy-to-learn and software that is easy-to-use. What will be the impact of such subsidies on social welfare?

We first examine the original model where consumers differ in the intensity of use. Suppose that we subsidize the cost of developing easy-to-learn and easy-to-operate software at rates s_v and s_F respectively. Welfare can be written as

$$W(x, v, F) =$$
$$[u(x, v, F) - p(x, v, F)x] + [p(x, v, F)x - (1 - s_v)c_v(v) - (1 - s_F)c_F(F)]$$
$$-[s_v c_v(v) + s_F c_F(F)] .$$

The three bracketed terms in this expression are consumers' surplus, producer's surplus, and government expenditure respectively. Differentiating this expression with respect to s_v and s_F and evaluating the derivative at the monopoly equilibrium with $s_v = s_F = 0$ we have

$$\frac{dW}{ds_v} = -\frac{\partial p}{\partial x}\frac{\partial x}{\partial s_v} + \left[\frac{\partial u}{\partial v} - \frac{\partial p}{\partial v}x\right]\frac{\partial v}{\partial s_v} + \left[\frac{\partial u}{\partial F} - \frac{\partial p}{\partial F}x\right]\frac{\partial F}{\partial s_v}$$
$$\frac{dW}{ds_F} = -\frac{\partial p}{\partial x}\frac{\partial x}{\partial s_F} + \left[\frac{\partial u}{\partial F} - \frac{\partial p}{\partial F}x\right]\frac{\partial F}{\partial s_F} + \left[\frac{\partial u}{\partial v} - \frac{\partial p}{\partial v}x\right]\frac{\partial v}{\partial s_F} .$$

In calculating these derivatives several terms drop out due to utility maximization

and profit maximization. The remaining terms are composed of two effects: the direct effect on consumers' surplus of changing v and F and the indirect effect of the induced output change and the cross effect of the subsidy.

We have already calculated the direct effect; it is zero for changes in F and positive for reductions in v. I show in the appendix that both the output effect and the cross effect have a positive effect on utility. Hence there is a case to made for imposing (small) subsidies on both cost functions. However, in terms of the impact on output, a reduction in F is exactly equivalent to imposing an output subsidy on the monopolist. Hence the social benefits of making the software easier to learn are just the same as the benefits from an output subsidy. This is to be compared to the effects of subsidizing v. In this case the subsidy benefits consumers both through the increase in output *and* through the improvement in "quality."

Another way to observe this is to consider the case where $F = 0$. In this case the profit-maximization problem for the monopolist becomes

$$(b - v)H(x) - (1 - s_v)c_v(v) .$$

The first-order condition for output is

$$(b - v)H'(x) = 0 ,$$

which is independent of s_v. Hence output doesn't change when ease-of-operation is subsidized. Nevertheless, welfare increases due to the impact of the quality change on the inframarginal consumers.

We turn now to the second model where consumers differ in the cost of learning. We show in the appendix that $dx/ds_F > 0$ in this case as well. Hence the impact on social welfare is composed of two effects: the benefit from having more output and the cost from the monopolist investing "too much" in making the software easy-to-learn. The combination is ambiguous, but at least it forces attention on the proper tradeoff: one would have to expect a big output effect from a subsidy in order for it to be worthwhile from a social point of view.

I interpret this as saying that it is reasonable to use public funds to subsidize research on how to make software easy to operate. However, there is no particular argument, in the context of this model at least, to subsidize research on how to make software easier to learn—the market gives the monopolist the right incentives with respect to this choice already, at least conditional on the output chosen by the monopolist. Of course, there may be other reasons to subsidize research on this aspect of software design. For example, there may be economies of scale in research, or there may be problems with appropriability of intellectual property that could cause problems for developing easy-to-operate software in the private sector. Or it may be that there are lower costs to developing easy-to-learn software in an educational environment. It would clearly be premature to make policy pronouncements without careful consideration of these possibilities.

Appendix. Comparative statics of the profit maximization problem

Here we study the impact of subsidizing research on ease-of-learning and ease-of-operation. We assume that this research lowers the cost of providing easier-to-operate and easier-to-learn software. We model this cost reduction as being equivalent to subsidies of s_v and s_F on the cost functions. Letting $R(x) = H(x)x$, we can write the monopolist's profit maximization problem as

$$\max_{x,v,F} (b - v)R(x) - Fx - (1 - s_v)c_v(v) - (1 - s_F)c_F(F) .$$

Note that we have set the cost of production equal to zero. Alternatively we could incorporate a constant marginal cost of production into F.

The first-order conditions for this problem are

$$\begin{aligned}
&(b - v)R'(x) - F = 0\\
&-R(x) - (1 - s_v)c_v'(v) = 0\\
&-x - (1 - s_F)c_F'(F) = 0 .
\end{aligned}$$

Totally differentiating this system and evaluating the derivatives at $s_v = s_F = 0$ we have

$$\begin{pmatrix} (b - v)R'' & -R' & -1 \\ -R' & -c_v'' & 0 \\ -1 & 0 & -c_F'' \end{pmatrix} \begin{pmatrix} dx \\ dv \\ dF \end{pmatrix} = \begin{pmatrix} 0 \\ -c_v' \, ds_v \\ -c_F' \, ds_F \end{pmatrix} .$$

In the case of a regular maximum the second-order conditions imply that the determinant of the Hessian matrix on the left-hand side of this expression will be negative and all principal minors of order 2 will be positive. This latter condition implies

$$\begin{aligned}
&(b - v)R''(x)c_v''(v) + R'(x)^2 < 0\\
&(b - v)R''(x)C_F''(F) + 1 < 0
\end{aligned} \tag{4}$$

These conditions are useful in signing the comparative statics effects.

Let $H < 0$ denote the value of the determinant of the Hessian and solve for the various differentials.

$$dF = \frac{-ds_v R'(x)c_v'(v) + ds_F c_F'(F)[(b - v)R''(x)c_v''(v) + R'(x)^2]}{H}$$

$$dv = \frac{-ds_F R'(x)c_F'(F) + ds_v c_v'(v)[(b - v)R''(x)c_F''(F) + 1)]}{H}$$

$$dx = \frac{ds_v R'(x)c_v'(v)c_F''(F) + ds_F c_F'(F)c_v''(v)}{H}$$

It is straightforward to verify

- dv/ds_v and dF/ds_F are negative. Hence subsidizing either ease-of-learning and ease-of-operation will tend to lead to improvements in those variables.
- dx/ds_v and dx/ds_F are positive. Hence the subsidies tend to increase output.
- dv/ds_F and dF/ds_v are negative. Hence subsidizing ease-of-learning will lead to an improvement in ease-of-operation and vice versa.

The other model discussed in the text was based on differences among the consumers in ease of learning. The profit maximization problem in that case is:

$$\max_{x,F} (b - v)x - FK(x)x - (1 - s_F)c_F(F) .$$

The first-order conditions are

$$(b - v) - FR'(x) = 0$$
$$-R(x) - (1 - s_F)c_F'(F) = 0 ,$$

where $R(x) = K(x)x$. Totally differentiating this system and evaluating the result at $s_F = 0$ we have

$$\begin{pmatrix} -FR'' & -R' \\ -R' & -c_F'' \end{pmatrix} \begin{pmatrix} dx \\ dF \end{pmatrix} = \begin{pmatrix} 0 \\ -c_F' \, ds_F \end{pmatrix} .$$

From this we find

$$\frac{dx}{ds_F} = \frac{\begin{vmatrix} 0 & -R' \\ -c_F' & -c_F'' \end{vmatrix}}{\begin{vmatrix} -FR'' & -R' \\ -R' & -c_F'' \end{vmatrix}} > 0 .$$

Acknowledgements

This work was supported by the National Science Foundation. I also wish to thank the Dipartimento d'economia politica at the Universitá di Siena for their hospitality during the period of this research. I would like to thank Jim Adams, Paul Courant and Judy Olson for their comments on an earlier draft of this paper.

Notes

[1] I first heard of this distinction in a discussion with Paul Scott. For some background concerning design of user-friendly software, see Nakamura (1990). See Mantei & Teorey (1988) for a discussion of benefit-cost analysis of software design from the viewpoint of software purchases by large corporations.

[2] Data is for the MS-DOS market only. Database figures come from *Computer Reseller News*, July 15, 1991. Wordprocessing figures come from *Standard & Poor's Industry Surveys*, April 1991. Spreadsheet figures come from *Datamation*, December 15, 1990.

[3] Data from Bulkeley (1991); see also Fisher (1992).

[4] The most common example of quality in the literature is "durability." See the seminal work of Swan (1970) and the survey by Schmalensee (1979). See also the textbook treatment by Tirole (1988).

[5] This observation also has global implications. Start at the profit-maximizing position and consider a

"large" change in quality. Profits go down since we've moved away from the profit-maximizing choice; if consumers' surplus also goes down, welfare unambiguously decreases.

[6] The "tilt" terminology is slightly misleading since the normal usage of tilt implies a *constant* change in slope. Of course this is not necessary for the result; all that is required is that the change in q either increases or decreases the slope of the demand curve at every point—not that it changes the slope of the demand curve by the same amount at every point.

[7] For some background on upgrades, see Bulkeley (1990).

[8] I intend to investigate software upgrades in future work.

[9] An example that fits this model might be tax preparation software. You use this software only once a year, so that there is little difference in intensity of use across the population. Whether or not you choose to use the software depends primarily on how easy it is to learn.

References

Bulkeley, W. (1990). Software users are beginning to rebel against the steady stream of upgrades. *Wall Street Journal*, B4.

Bulkeley, W. (1990). Software industry loses start-up zest as big firms increase their domination. *Wall Street Journal, August 27*, B1.

Fisher, L. (1992). Business turns tough in software. *New York Times, December 14.*

Mantei, M. & Teorey, T. (1988). Cost/benefit analysis for incorporating human factors in the software lifecycle. *Communications of the ACM, 4*, 428–439.

Nakamura, R. (1990). The *x* factor. *Infoworld, November 19*, 51–55.

Schmalensee, R. (1979). Market structure, durability, and quality: a selective survey. *Economic Inquiry, 42*, 177–196.

Sheshinski, E. (1976). Price, quality and quantity regulation in monopoly situations. *Economica, 43*, 127–137.

Spence, M. (1975). Monopoly, quality and regulation. *Bell Journal of Economics, 6*(2), 417–429.

Swan, P. (1970). Durability of consumer goods. *American Economic Review, 60*, 884–894.

Tirole, J. (1988). *The Theory of Industrial Organization*. Cambridge, MA: MIT Press.