

Research Note—Sell First, Fix Later: Impact of Patching on Software Quality

Ashish Arora, Jonathan P. Caulkins, Rahul Telang

H. John Heinz III School of Public Policy and Management, Carnegie Mellon University, 5000 Forbes Avenue,
Pittsburgh, Pennsylvania 15213 {ashish@andrew.cmu.edu, caulkins@andrew.cmu.edu, rtelang@andrew.cmu.edu}

We present a model of fixing or patching a software problem after the product has been released in the market. Specifically, we model a software firm's trade-off in releasing a buggy product early and investments in fixing it later. Just as the marginal cost of producing software can be effectively zero, so can the marginal cost of repairing multiple copies of defective software by issuing patches. We show that due to the fixed cost nature of investments in patching, a software vendor has incentives to release a *buggier* product early and patch it later in a larger market. Thus, a software monopolist releases a product with *fewer* bugs but later than what is socially optimal. We contrast this result with physical good markets where market size does not play any role in quality provision. We also show that for comparable costs, a software monopolist releases the product with more bugs but invests more in post-patching support later than the physical good monopolist.

Key words: patching; software quality; time of entry; fixed costs; software bugs

History: Accepted by Rajiv D. Banker, information systems; received August 5, 2003. This paper was with the authors 3 months for 2 revisions.

1. Introduction

Software has become an intrinsic part of our lives: In 2002, the U.S. software market was worth almost \$180 billion. As software becomes ingrained in daily business activities, its failures become ever more critical. A recent study puts the annual cost of major software bugs to the U.S. economy at over \$60 billion (NIST 2002). Media reports of catastrophic effects of software failure abound. For instance, a buffer overflow bug caused the Ariane 5 rocket to blow up 40 seconds after lift-off in 1996 (estimated loss of over \$500 million).¹ The business community also pays close attention to software malfunctions. Fully 97% of the 800 managers surveyed by *Information Week* (2002) reported software flaws in their systems in the past year. More than 90% blamed faulty software for lost revenue or higher costs. Some 62% said they believed that the software industry was doing a bad job of producing bug-free software (*Information Week* 2002).

The overall quality experienced by a software user is a combination of the bugs in the product and the efforts the vendor makes to patch the bugs after shipping. Indeed, most models of the software development cycle explicitly incorporate the possibility of improving the product after release through patches and upgrades (Reddy and Evans 2002).² In fact,

patching has become so common that many organizations have "patch-management" systems in place to efficiently integrate patches in existing software (*Information Week* 2004). Patching is most common in packaged software. With customized, mission critical, or embedded software, patching is rarely an option because of the very high cost of failure.

It is a commonplace that "time to market" pressures are an important, if not the most important, reason that software has bugs. Delay is costly to vendors because, all else equal, users would rather get a product sooner (see, e.g., Baskerville et al. 2001). For example, a typical view of the manager is, "I would rather have it wrong than have it late. We can always fix it later" (Paulk et al. 1995, p. 4). However, reducing bugs requires time. Indeed, in the basic model presented here, bugs can only be reduced by delaying release.

It is tempting to assume that firms can reduce the time to market without affecting the product quality by simply increasing the number of developers.³ However, Brooks' famous study of the IBM OS/360 suggests the opposite: The greater costs of coordinating among more developers may outweigh any gain in efficiency (Brooks 1995). Further, Amdahl's law points to a natural limit to the size

¹ More examples, some possibly apocryphal, can be found at <http://www.softwareqatest.com/index.html>.

² Sun released more than 200 patches for Solaris 9.0 to fix more than 1,200 bugs in a six-month period (downloaded from <http://sunsolve.sun.com>).

³ Organizations can, and do, reduce bugs by adopting processes such as CMM or ISO9001. Ultimately, however, having fewer bugs at the time of release depends on careful design, coding, and extensive testing under different conditions and for different tasks. This takes time.

of the product development team, especially for new and unprecedented software. Therefore, our model is more applicable to off-the-shelf packaged software when reducing bugs takes time and when the disutility of the remaining bugs is not so catastrophic that no one would buy a buggy product.

1.1. Research Questions

In this paper, we formally analyze how the ability to patch bugs later affects the vendor's decision about when to enter the market, and how buggy the initial product is. We then contrast these decisions when the product is a tangible good like an automobile instead of software. A key finding is that the ability to fix bugs later always leads to an earlier release of the product, and a larger market increases the incentive to release early with more bugs. A more striking finding is that it is indeed socially efficient to release the product early and to patch extensively later. Hence, a monopolist releases products later and with fewer bugs, but also patches less than is socially efficient.

The ability to fix or otherwise mitigate software defects after release is an underappreciated difference between software and physical goods (Picker 2004). Physical goods can, of course, be recalled to fix defects. However, the key distinction is that while the cost of fixing a software bug *ex post* is roughly independent of the number of copies of the software sold (even though the cost of actually identifying a bug and writing and testing the code to fix it can be significant, it often has fixed cost nature), the cost of recalling and fixing an automobile will vary directly with the number of cars sold.

Occasionally patching costs may vary by number of copies sold if, as was sometimes true for Y2K fixes, a consultant must visit each user to help install the patch. However, a more common practice is to e-mail the patch to customers or simply make it available on the vendor's website with no customer-specific adaptation. So, to caricature, in our model the software aftermarket repairs entail only fixed costs, whereas for physical goods variable costs alone matter.

The fixed cost nature of patching investment also implies that the effective market size plays a crucial role. Indeed, we formally show that all else equal, a large market size implies earlier shipping (longer "time *in market*") but also more investment in patching (after-sale support) and higher overall value to users. This result provides intuition for the otherwise unexpected result that, relative to the socially efficient outcome, a software monopolist ships products later with fewer bugs, but commits to less patching. In contrast, we show that for physical products, market size is irrelevant; the monopolist provides socially efficient levels of defects and post-sale support. Finally, for comparable costs, we show that the

software monopolist enters the market early with a buggier product but invests more in patching than the tangible good monopolist.

We assume that a vendor can commit to a chosen level of patching, either contractually or through a need to maintain its reputation. Absent such commitment, the vendor has no incentive to patch and rational users would recognize this. Further, ours is a full information model where users are fully aware of existing bugs as well as the producer's commitment to fix some fraction of them later. Hence, any suboptimality is not due to incomplete information. Even if the number of bugs were stochastic, our model would still apply provided the vendor did not have private information.

This paper is organized as follows. Section 2 reviews the literature. We outline the model, including the monopolist's and social planner's decisions, in §3. We also provide the comparative analysis with a tangible good in this section. Section 4 explores the robustness of the results to various generalizations, including fixed development costs, uncertainty, and more general functional forms. Section 5 summarizes managerial insights and concludes.

2. Prior Literature

In software engineering literature it has been argued that better processes lead to higher quality and lower costs and maintenance, and that it is cheaper to find and fix bugs earlier rather than later (Krishnan et al. 2000, Banker and Slaughter 1997). Our focus is not on process economics, but on the trade-off between delaying product release to reduce bugs and investing in patching bugs after release. Banker and Slaughter (1997) show that managers are often unwilling to delay product release, even though such a delay could save them money in the long run. Hendricks and Singhal (1997) show that stock markets also punish the firms who delay their products. Cohen et al. (1996) analyze the performance and time-to-market trade-off for physical products when repairing defects after product release is exorbitantly costly. The literature shows that higher performance provides higher customer value (Zirger and Maidique 1990), but it also causes significant development delay (Griffin 1993, Lilien and Yoon 1985). Our analysis differs from much of the work to date in that we allow the vendors to issue patches after the product has been released.

3. Model

3.1. Consumers' Utility

All else equal, users prefer software products that are released earlier and have fewer bugs. We define user

utility function as⁴

$$U = \theta[V - k(\delta)B(t)]t - p. \quad (1)$$

V is the baseline utility per unit of time if the product is bug free, $B(t)$ is the number of bugs when the product is released,⁵ $k(\delta)B(t)$ is the cost of the bugs to users remaining after patching, and t is the amount of time a user uses the product. For simplicity, we ignore time discounting. Time t is measured backwards, so high t means that the product is released early giving the user more time to derive utility. Note that rushing to release the product early results in more bugs and at an increasing rate, so we assume that $B(t)$ is increasing and convex in t , i.e., $B' > 0$ and $B'' > 0$.

The proportion of bugs patched by the vendor is δ ; thus, $(1 - \delta)$ proportion of bugs are still unresolved. To capture the notion that patches cannot be a complete substitute for the bug-free software in the first place, we instead use $k(\delta)$ as the proportion of defect costs (to the user) that has *not* been reduced by the firm through patching. Because presumably the utility loss decreases with more patching (higher δ), albeit at a diminishing rate, we let $k'(\delta) < 0$ and $k''(\delta) > 0$. The proportion of *defects costs remaining*, $k(\delta)B(t)$, ought to be interpreted as an average over time or an expectation. For some products, it might reflect the proportion of defects fixed immediately via a “patch” available for download. For others, the proportion of defects remediated might initially be zero but might increase over time so that unremediated defect cost, averaged over the time over which the product is used, t , is $k(\delta)B(t)$.⁶

The price paid by customers is p , and the coefficient θ captures customer heterogeneity as they derive different utility per unit time from the software’s functionality. Thus, high θ users, such as frequent users, derive more utility from the product but also incur higher loss from bugs. We assume that θ is uniformly distributed in $[0, 1]$. In §4, we relax this assumption. We assume that the marginal cost of producing the product is zero so that the vendor’s objective function consists of revenue minus the fixed cost of patching. The fixed cost of patching is assumed

to be proportional to the number of bugs patched (or proportional to the number of patches).⁷ Letting $D(p, t, \delta)$ represent the demand function, the vendor’s objective function is

$$\pi = D(p, t, \delta)p - FB(t)\delta, \quad (2)$$

where F is the average cost of each patch.

3.2. Market Equilibrium Under a Monopoly

Suppose that there are N customers in the market. Given (1) and the uniform distribution of θ , the monopolist vendor’s demand is

$$D(p, t, \delta) = N \left(1 - \frac{p}{[V - k(\delta)B(t)]t} \right).$$

Hence, the profit Equation (2) for the monopolist after substituting the profit-maximizing price is

$$\pi(t, \delta) = \frac{1}{4}N[V - k(\delta)B(t)]t - FB(t)\delta. \quad (3)$$

It is useful at this stage to introduce $\Psi(\delta, t; m) = mN[V - k(\delta)B(t)]t - FB(t)\delta$. We assume that

$$-\frac{d \log(k)}{d \log(\delta)} < 2\{d \log(-k')/d \log(\delta)\} \cdot \frac{d \log(B)}{d \log(t)}$$

to ensure that $\Psi(\delta, t; m)$ is concave in t and δ . Roughly, this condition requires that the elasticity of bugs with respect to delay be high, and that the utility loss from the bugs not be very responsive to δ relative to the change in marginal utility loss. This condition is satisfied for commonly used functional forms, such as $B(t) = t^2$ and $k(\delta) = \exp(-\lambda\delta)$. The first-order conditions for the optimal t and δ are

$$\begin{aligned} \frac{d\pi(t, \delta)}{dt} &= \frac{1}{4}N[V - k(\delta)B'(t)t - k(\delta)B(t)] - FB'(t)\delta = 0, \quad (4) \end{aligned}$$

$$\frac{d\pi(t, \delta)}{d\delta} = -\frac{1}{4}Nk'(\delta)B(t)t - FB(t) = 0. \quad (5)$$

Also note that for δ satisfying the first-order conditions, because $-(1/4)Nk'(\delta^*)t^* = F$ from (5),

$$\frac{d^2\pi(t, \delta)}{d\delta dt} = -\frac{1}{4}Nk'(\delta)[B'(t)t + B(t)] - FB'(t) > 0. \quad (6)$$

This “complementarity” follows naturally from how the defect costs are defined and obtained with very general utility functions (see §4). Note that it holds only at the optimal values of δ and is useful (but not necessary) for our results.

⁷ Section 4 extends the model to include development costs, which affect both release time and bugs.

⁴ Utility has the form $U = \theta q - p$, where $q = [V - k(\delta)B(t)]t$. Such structure is widely used in literature (Moorthy 1988).

⁵ We defined bugs as defects in the original products. The model is unchanged if defects are also interpreted as a “lack of features” which can be added after product release (e.g., Bessen 2002).

⁶ Our results are unchanged if users incur a cost to apply patches. One can also model the case where the vendor decides on the number of patches (δ) instead of the proportion. In this case, the utility function is $U = \theta[V - B(t) + k(\delta)]t - p$ and the cost function is $F\delta$. The results remain unchanged.

3.3. Benchmark: The Socially Efficient Outcome

It is interesting to compare the choices of the monopolist to the socially efficient level. Efficiency requires that the product is priced at marginal cost, which is assumed to be zero.⁸ At this price, the entire market will be covered. Therefore, the social welfare function is

$$\begin{aligned} S(t, \delta) &= \int_0^1 \theta N[V - k(\delta)B(t)]t \, d\theta - FB(t)\delta \\ &= \frac{1}{2}N[V - k(\delta)B(t)]t - FB(t)\delta. \end{aligned} \quad (7)$$

Both (3) and (7) have the form $\Psi(\delta, t; m)$, where $m = 1/2$ for the social planner and $m = 1/4$ for the monopolist. The appendix shows that the δ^* and t^* that maximize $\Psi(\delta, t; m)$ are increasing in m . This implies that it is socially efficient to release the product earlier than the monopolist (and hence with more bugs) but patch more aggressively, as summarized in Proposition 1 below.

PROPOSITION 1. *For a software product, the monopolist releases the product later with fewer bugs but invests less in patching than the socially efficient level.*

This is the principal result of this paper. It is surprising in that, contrary to popular belief, the monopolist releases less buggy product than what is socially optimal. But the monopolist provides less value by entering the market later and providing less ex post support.

3.4. Role of Market Size

Note that in $\Psi(\delta, t; m)$, m (proportion of market covered) and N (potential market size) have similar effects. It follows that a producer facing a larger market (larger N) will enter earlier (larger t) with a bugger product but patch more extensively (larger δ). The intuition is that because the cost of patching can be amortized over a larger sales volume, a larger market induces a greater investment in δ and earlier product release. One can make a similar argument for lower F . Further, this intuition holds for a social planner as well. Hence, we get:

COROLLARY 1. *A software producer facing a larger market (N) or lower fixed costs per bug patched enters the market earlier with more bugs but provides higher patching support later.*

Thus, our results provide an empirically testable hypothesis that when the market is large, or when the fixed cost of patching is small, or when the vendors have better support infrastructure, they enter

the market earlier and with buggier products. For instance, Microsoft has a large market for many of its products and it provides extensive patching support. Many of its patches are released in “service packs” that makes it easy for users to download and install new patches. These service packs can identify bugs and vulnerabilities on a computer and automatically download the right patches. Microsoft is willing to incur these large costs in after-sale support because they can be amortized over a large user base.

3.5. Quality in Tangible Products

It is worthwhile to compare these results with a tangible good producer who incurs a positive marginal cost of fixing a defect. We follow the same model structure except that we replace the fixed cost per “patch” with a cost that varies with the number of units fixed as well as with the number of bugs. If $fb(t)\delta$ is the marginal cost of fixing a defect per unit sold, the profit function for a tangible product monopolist (after substituting the profit-maximizing price) is

$$\pi(t, \delta) = N \frac{[(V - k(\delta)B(t))t - fb(t)\delta]^2}{4[V - k(\delta)B(t)]t}. \quad (8)$$

Social efficiency requires marginal cost pricing, so that the social surplus is

$$S(t, \delta) = N \frac{[(V - k(\delta)B(t))t - fb(t)\delta]^2}{2[V - k(\delta)B(t)]t}. \quad (9)$$

Because the objective function of the monopolist and the social planner differ by only a multiplicative constant, they will make the same choices with respect to δ and t .

PROPOSITION 2. *When the ex post cost of fixing a defect is a marginal cost (e.g., for a tangible good), the monopolist's choice of product release time (t) and fraction of defects fixed (δ) is socially efficient and independent of market size.*

When patching costs do not vary by quantity sold, as in software, the monopolist, who produces less than the socially optimal quantity, also invests less in patching (smaller δ) and enters late (lower t). Complementarity between δ and t ensures that there is additional incentive to delay the product to reduce bugs. Clearly, the fixed cost nature of patching defects after product release can create a “market failure” (difference between the market equilibrium and the socially optimal outcome), different from the tangible product case.⁹

⁹ An alternative, but not mutually exclusive, interpretation of the difference between software and tangible goods has to do with differences in the legal environment. For many tangible goods, a product defect exposes the manufacturer to liability claims and, in general, requires restitution to the user. Software vendors are typically not exposed to such claims, although the legal situation is still unclear.

⁸ The result holds even if we impose a “break-even” constraint because, as we explain, the key is that the monopolist serves a smaller fraction of the market than is socially efficient.

3.6. Comparing a Software Monoplist with a Tangible Good Monoplist

Comparing the optimal choices of t and δ for software and tangible good monopolists requires some way of comparing a fixed cost (in units of \$) with a variable cost (in units of \$ per unit of output). Suppose that the optimal choice of the tangible good monopolist is t_p and δ_p so that its total patching cost is $Q_p \cdot fB(t_p)\delta_p$, where Q_p is the optimal quantity sold. Define \hat{F} such that the software monopolist incurs the same total patching cost when it chooses t_p and δ_p . Thus,

$$\hat{F}B(t_p)\delta_p = Q(t_p, \delta_p) \cdot fB(t_p)\delta_p. \quad (10)$$

We show in the next proposition that for all $F \leq \hat{F}$, a software monopolist enters earlier (i.e., chooses $t > t_p$) and invests more in patching (i.e., chooses $\delta > \delta_p$).

PROPOSITION 3. *For all $F \leq \hat{F}$, where \hat{F} is defined by Equation (10), a software monopolist enters earlier with a buggier product and invests more in patching support than does a tangible good monopolist.*

Recall that the total number of patches (or number of recalls for a tangible good) issued by the firm is $B(t)\delta$. Because a software monopolist chooses a higher t and δ , it issues more patches than a tangible good monopolist issues recalls. Casual empiricism suggests that product recalls for physical goods are rarer and economically more significant than software patches. A number of studies show that a product recall adversely affects the stock market performance of the seller (e.g., Jarrell and Peltzman 1985, Davidson and Worrell 1992). On the other hand, a recent article pointedly noted the news that Microsoft had let one month go by without releasing a single patch (*Information Week* 2005); software vendors such as HP, Sun, and Microsoft issue so many patches that only a failure to issue patches is noteworthy.

In the online appendix (<http://mansci.pubs.informs.org/ecompanion>) we show that this result is robust to generalization of distribution of θ .

4. Robustness

In this section, we explore the robustness of our results to a variety of generalizations. The formal proofs are provided in the online appendix.

4.1. Fixed Development Cost

Let $P(t, x)$ be the product development cost, where x is the resources invested in development (e.g., number of programmers). We assume that $P_t > 0$ and $P_x > 0$: Releasing the product early costs more and investing more resources is costly. We also generalize B to $B(t, x)$ such that $B_x < 0$ and $B_{xx} > 0$: Higher investment up front reduces the number of

bugs but at a diminishing rate. The rest of the definitions remain same and $\Psi(\delta, t, x; m) = mN[V - k(\delta)B(t, x)]t - FB(t, x)\delta - P(t, x)$. The vendor now optimizes over x as well. In the online appendix we show that $P_{xt} \leq 0$ and $B_{xt} \leq 0$ are sufficient conditions for our results to hold.¹⁰ However, because B also depends on x , the monopolist's product is no longer unambiguously less buggy than socially efficient.

4.2. General Functional Form of Utility

For ease of exposition, we use a specific structure for the utility function. However, our results are robust to a more general utility function such as $U = \theta V(t, y) - p$, where y is the cost of unresolved bugs, and is equal to $k(\delta)B(t)$. We expect that $V_t > 0$ and $V_{tt} < 0$ such that utility is increasing in t but at a decreasing rate. Similarly, $V_y < 0$ and $V_{yy} < 0$; unresolved bugs reduce the utility at an increasing rate. One can show that all results continue to hold even with such a specification.¹¹ However, if we additionally include product development cost P , as in §4.1, then we need additional restrictions. As the introduction notes, often bugs cannot be reduced simply by increasing programmers or testers. Therefore, if B_x were small (where x is the investment as in §4.1), then even with fixed development costs, all results would continue to hold.

4.3. Fixing the Amount of Time a User Uses the Product

The discussion of the more general utility function also provides reassurance that our results are not driven by the assumption that the product has a pre-specified date of obsolescence and that the length of time the product is used enters multiplicatively. We show in the online appendix that results remain intact even if the product were to have a fixed life cycle of length τ from the date of release, as long as users still value a product released earlier.

4.4. Stochastic $B(t)$

The model analyzed assumes full information. However, one can easily allow for uncertainty in the number of bugs $B(t)$ by reinterpreting it as the expected number of bugs and assuming that users observe t , and, hence, know the expected number of bugs. Formally, let the actual number of bugs, b , be a random variable with mean $B(t)$. Then, the expected utility for a user θ is $U = E\theta[V - k(\delta)b]t - p = \theta[V - k(\delta)B(t)]t$. The vendor's profit function is unchanged as well, albeit with $B(t)$ now being the expected number of bugs. In essence, with risk neutral agents, uncertainty

¹⁰ If tangible goods also incur a fixed development cost, then Propositions 2 and 3 will require additional assumptions.

¹¹ The proofs are available from the authors upon request.

does not create any problems.¹² However, private information may involve additional considerations as the informed party tries to signal the number of bugs through various means (e.g., time in market). This is beyond the focus on this paper and is not treated here.

4.5. General Distribution of θ

The model analyzed in the paper assumes that θ has a uniform distribution. We show in the appendix that Propositions 1 and 3 hold for any distribution function $G(\theta)$, but Proposition 2 requires additional restrictions on the probability distribution. Thus, the result that a software monopolist enters late and invests less in patching is robust to the distribution of buyer heterogeneity.

5. Conclusions and Future Research

Recent focus on software quality and vulnerability makes it important to understand the incentives of software vendors to provide quality. An important way in which software products differ from traditional products is that products can be fixed and improved after release, which also affects when products are released and how buggy the product is at the time of release.

We analyze the firm’s trade-off in time to release the product versus investments in ex post remediation of software failures. We show that the fixed cost nature of investments in bug remediation means that firms facing a *larger* market tend to enter early with a bug-gier product but then invest more in after-sale remediation. Because a monopolist restricts output relative to socially efficient levels, this leads to the counterintuitive result that a monopolist supplies a product with fewer bugs but then provides less “after-sale” support and, therefore, reduces customer value. In contrast, in the case of traditional tangible goods with significant marginal costs of remediation, market size does not play any role and, thus, a monopolist’s time of release (and hence, also, ex ante quality) is socially efficient. Interestingly, for comparable costs, we find that a software monopolist will enter the market earlier with a bug-gier product and invest more in patching than a tangible good monopolist. We also show that our results are robust to many generalizations. Consistent with our model, we note that there is widespread dissatisfaction with, for instance, many security vulnerabilities in Microsoft’s products, but few can match the extensive system of updates and patches Microsoft provides.

¹² With more general functional forms for utility (e.g., risk-averse buyers), $B(t)$ would have to be replaced by the certainty equivalent number of bugs in the utility function. The cost function would continue to feature $B(t)$. This complicates, but does not qualitatively change, the analysis.

A further extension would allow firms to release at multiple dates, with later versions being more reliable with fewer bugs, and for customers to optimally decide on which version to buy. Another extension would explore oligopoly and duopoly competition as competition would force firms to differentiate their products.

An online companion to this paper is available on the *Management Science* website (<http://mansci.pubs.informs.org/ecompanion.html>).

Acknowledgments

The authors thank the department editor, associated editor, and two anonymous reviewers for many helpful comments.

Appendix

PROOF OF PROPOSITION 1. Consider the objective function $\Psi(t, \delta; m) = m[V - k(\delta)B(t)]t - FB(t)\delta$. (One can, without loss of generality, consider m to include N . Thus, we assume that $N = 1$ to save on notation.)

The first-order conditions for t and δ are

$$\Psi_t = m[V - k(\delta)B'(t)t - k(\delta)B(t)] - FB'(t)\delta = 0,$$

$$\Psi_\delta = -mk'(\delta)B(t)t - FB(t) = 0 \quad \text{or} \quad -mk'(\delta)t = F.$$

It follows that

$$\frac{dt}{dm} = \frac{\begin{vmatrix} -FB'(t)\delta & -mk'(\delta)B(t) \\ k'(\delta)B(t)t & -mk'(\delta)B(t)t \end{vmatrix}}{H(t, \delta)} \geq 0,$$

because the numerator is positive (because $k''(\delta) > 0$) and the denominator is positive by assumption. Similarly, we have that

$$\frac{d\delta}{dm} = \frac{\begin{vmatrix} [-mk(\delta)(2B'(t) - B''(t)t) - FB''(t)\delta] & -FB'(t)\delta \\ [-mk'(\delta)B(t)] & k'(\delta)B(t)t \end{vmatrix}}{H(t, \delta)} \geq 0$$

because $k'(\delta) < 0$.

PROOF OF PROPOSITION 3. To save on notation, we assume that $N = 1$. For a tangible good monopolist, the profit function from (8) is

$$\pi(t, \delta) = \frac{[(V - k(\delta)B(t))t - fB(t)\delta]^2}{4[V - k(\delta)B(t)]t}.$$

The first-order conditions are

$$\frac{\partial \pi}{\partial t} = \frac{1}{4}[V - k(\delta)B'(t)t - k(\delta)B(t)] - \frac{fB'(t)\delta[V - k(\delta)B(t)]t}{2[fB(t)\delta + (V - k(\delta)B(t))t]} = 0, \tag{i}$$

$$\frac{\partial \pi}{\partial \delta} = -\frac{1}{4}k'(\delta)B(t)t - \frac{fB(t)[V - k(\delta)B(t)]t}{2[fB(t)\delta + (V - k(\delta)B(t))t]} = 0. \tag{ii}$$

Let the optimal choices of the tangible good monopolist be t_p and δ_p . Using the notation $z = [V - k(\delta)B(t)]t$ and $c = fB(t)\delta$, the quantity sold can be written as $Q = (z - c)/2z$. From the definition of \hat{F} , we get

$$\hat{F} = \frac{z - c}{2z} \frac{c}{B(t)\delta}$$

evaluated at t_p and δ_p .

To show that for all such $F \leq \hat{F}$, a software monopolist offers $t > t_p$ and $\delta > \delta_p$, it is sufficient to show that the software monopolist's profit function is increasing t and δ evaluated at $\{t_p, \delta_p\}$. Recall that the profit function for the software monopolist satisfies

$$\frac{\partial \pi}{\partial t} = \frac{1}{4}[V - k(\delta)B'(t)t - k(\delta)B(t)] - FB'(t)\delta, \quad (\text{iii})$$

$$\frac{d\pi}{d\delta} = -\frac{1}{4}k'(\delta)B(t)t - FB(t). \quad (\text{iv})$$

Evaluating (iii) at $\{t_p, \delta_p\}$ implies

$$\frac{\partial \pi(t_p, \delta_p)}{\partial t} = \frac{fB'(t)\delta z}{2(c+z)} - \hat{F}B'(t)\delta.$$

From the definition of \hat{F} , this simplifies to

$$\frac{\partial \pi(t_p, \delta_p)}{\partial t} = \frac{B'(t)c}{2B} \left[\frac{c^2}{(z+c)z} \right] > 0.$$

Thus, the software monopolist wants to offer higher $t > t_p$ for all $F \leq \hat{F}$. Evaluating (iv) at $\{t_p, \delta_p\}$ implies

$$\frac{\partial \pi(t_p, \delta_p)}{\partial \delta} = \frac{fB(t)z}{2(c+z)} - \hat{F}B(t),$$

which simplifies to

$$\frac{\partial \pi(t_p, \delta_p)}{\partial \delta} = \frac{c}{2\delta} \left[\frac{c^2}{(z+c)z} \right] > 0.$$

Thus, the software monopolist also offers $\delta > \delta_p$.

Because Ψ (and, hence, also the profit function for the software monopolist) is assumed to be globally concave, it must be that the optimal t and δ for the software monopolist are greater than those for a tangible good monopolist with comparable cost and demand. \square

References

- Banker, R., S. Slaughter. 1997. A field study of scale economies in software maintenance. *Management Sci.* **43**(12) 1709–1725.
- Baskerville, R., L. Levine, J. Pries-Heje, B. Ramesh, S. Slaughter. 2001. How Internet software companies negotiate quality. *IEEE Comput.* **34**(5) 51–57.
- Bessen, James. 2002. Open source software: Free provision of complex public goods. Working paper, Massachusetts Institute of Technology, Cambridge, MA.
- Brooks, F. R. 1995. *The Mythical Man Month*, 2nd ed. Addison-Wesley, Reading, MA.
- Cohen, M. A., J. Eliashberg, Ho Teck-Hua. 1996. New product development: The performance and time-to-market tradeoff. *Management Sci.* **42**(2) 173–186.
- Davidson, W. L., III, D. L. Worrell. 1992. The effect of product recall announcements on shareholder wealth. *Strategic Management J.* **13**(6) 467–473.
- Griffin, A. 1993. Metrics for measuring product development cycle times. *J. Product Innovation Management* **10**(2) 112–125.
- Hendricks, K. B., V. R. Singhal. 1997. Delays in new product introductions and the market value of the firm: The consequences of being late to the market. *Management Sci.* **43**(4) 422–436.
- Information Week. 2002. Poor software quality: Time to do something about it. (May 20) www.informationweek.com
- Information Week. 2004. Get ready to patch. (August 30) www.informationweek.com
- Information Week. 2005. Microsoft: No patch before its time. (March 8) www.informationweek.com
- Jarrell, G., S. Peltzman. 1985. The impact of product recalls on the wealth of sellers. *J. Political Econom.* **93**(1) 512–536.
- Krishnan, M. S., C. H. Kriebel, S. Kekre, T. Mukhopadhyay. 2000. An empirical analysis of productivity and quality in software products. *Management Sci.* **46**(6) 745–759.
- Lilien, G., E. Yoon. 1990. The timing of competitive market entry: An exploratory study of new industrial products. *Management Sci.* **36**(5) 568–585.
- Moorthy, S. 1988. Product and price competition in a duopoly. *Marketing Sci.* **7**(2) 141–168.
- NIST Report. 2002. The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology, Gaithersburg, MD.
- Paulk, M. C., B. Curtis, M. B. Chrissis, C. V. Weber. 1995. *The Capability Maturity Model: Guidelines for Improving the Software Process for Software*. Addison-Wesley, Reading, MA.
- Picker, R. 2004. Cyber security: Of heterogeneity and autarky. Olin working paper no. 223, University of Chicago Law and Economics, Chicago, IL.
- Reddy, B., D. S. Evans. 2002. Government preferences for promoting open-source software: A solution in search of a problem. Working paper, <http://ssrn.com>.
- Zirger, B., M. Maidique. 1990. A model of new product development: An empirical test. *Management Sci.* **36**(7) 867–883.