Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

The Influence of Software Process Maturity and Customer Error Reporting on Software Release and Pricing

Terrence August, Marius Florin Niculescu

To cite this article:
Terrence August, Marius Florin Niculescu (2013) The Influence of Software Process Maturity and Customer Error Reporting on Software Release and Pricing. Management Science 59(12):2702-2726. http://dx.doi.org/10.1287/mnsc.2013.1728

MANAGEMENT SCIENCE
Vol. 59, No. 12, December 2013, pp. 2702–2726
ISSN 0025-1909 (print) | ISSN 1526-5501 (online)

http://dx.doi.org/10.1287/mnsc.2013.1728

The Influence of Software Process Maturity and Customer Error Reporting on Software Release and Pricing

Terrence August
Rady School of Management, University of California, San Diego, La Jolla, California 92093, taugust@ucsd.edu

Marius Florin Niculescu
Scheller College of Business, Georgia Institute of Technology, Atlanta, Georgia 30308,
marius.niculescu@scheller.gatech.edu

Software producers are making greater use of customer error reporting to discover defects and improve the quality of their products. We study how software development differences among producers (e.g., varying levels of process maturity) and software class and functionality differences (e.g., operating system versus productivity software) affect how these producers coordinate software release timing and pricing to optimally harness error reporting contributions from users. In settings where prices are fixed, we characterize the optimal release time and demonstrate why in some cases it can actually be preferable to delay release when customer error reporting rates increase. The manner in which a firm's optimal release time responds to increases in software functionality critically hinges on whether the added functionality enhances or dilutes user error reporting; in both cases, the effect of added functionality on release timing can go in either direction, depending on both firm and product market characteristics. For example, when processing costs are relatively large compared with goodwill costs, firms with lower process maturity will release earlier when per-module error reporting contributions become diluted and release later when these contributions become enhanced. We also examine how a firm adapts price with changes in error reporting levels and software functionality, and finally, we provide implications of how beta testing influences release timing.

Key words: software quality; software reliability; software security; software economics; software process maturity; network effects; software error reporting; diffusion of innovation
History: Received August 15, 2010; accepted January 26, 2013, by Lorin Hitt, information systems. Published online in *Articles in Advance* June 14, 2013.

1. Introduction

Over the last decade, the world has achieved phenomenal growth in broadband Internet penetration. The United States rose from 4.4% penetration in 2001 to 27.7% in 2011—a sixfold increase (Organisation for Economic Co-operation and Development (OECD) 2012). The OECD also reports that, averaging across the G7 countries, the penetration was approximately 18 times higher by the end of the decade.[1] This extensive growth in high-speed access internationally has presented both newfound opportunities and challenges. With currently over 900 million interconnected Internet hosts, over 2.2 billion Internet users, and a projected 15 billion devices connected to Internet protocol networks by 2015, the ease with which businesses, individuals, and society at large can communicate and exchange information is unprecedented (Cisco 2011, Internet Systems Consortium 2012, Internet World Stats 2012). Notwithstanding these benefits, the interconnectedness of computers and the speed at which information disseminates across the Internet are actively exploited by malicious individuals and software designed to cause substantial economic damages. According to RTI (2002), the annual cost of faulty software to the U.S. economy alone was $59.5 billion roughly a decade ago and has since grown to an estimated $75 billion (Michaels 2008). Hence, an ongoing challenge faced by both private entities and the public sector is how to continue to obtain increased value from this global interconnected network while mitigating effects associated with software quality.

Recently, software firms have begun to leverage consumers' connectedness to the Internet to enable their users to contribute toward increasing the reliability

[1] Globally, countries are continuing to make substantial investments to further promote this growth and foster economic development. As part of the American Reinvestment and Recovery Act in 2009, the U.S. appropriated $7.2 billion toward expanding broadband access in underserved areas, and the United Kingdom has also made a commitment to delivering broadband into every household by 2012 (Tryhorn 2009, Ransom 2010). Countries leading the way, such as South Korea, already have much higher penetration and can also offer significantly faster broadband speeds (Sutter 2010).

and security of their software products. Applications now routinely have built-in support for automatic error reporting through which application failure information is transmitted from users' systems to software firms. Microsoft's Windows Error Reporting, AutoCAD's Customer Error Reporting, Mozilla's Crash Reporter, and other similar implementations all aim to harness the diverse user population's idiosyncratic environments and behavior to help with software assurance (Markoff 2006). Through these tools, software firms collect substantial input from the user community regarding product failures in the field of operation. For example, Mozilla gets 2.5 million crash reports per day (Thompson 2010). In many instances, software testers cannot reproduce customer-discovered errors using identical input because factors involving "invisible users" or the interactions between the software and the customer's environment (operating system, file system, and libraries) are the actual root cause (Whittaker 2001). Hence, error reports generated by users' systems, which often include a snapshot of important environment information, have a significant potential to reduce the costs of correcting bugs; these costs can account for up to 50%–75% of software development costs (Muthitacharoen and Saeed 2009). In a memo to customers in October 2002, Steve Ballmer, CEO of Microsoft, notes that "in Windows XP Service Pack 1, error reporting enabled [Microsoft] to address 29 percent of errors involving the operating system and applications running on it, including a large number of third-party applications. Error reporting helped [Microsoft] to eliminate more than half of all Office XP errors with Office XP Service Pack 2" (Ballmer 2002).

With modern software assurance strategies that leverage users' systems and their interconnectedness, a software firm must carefully select product release dates that account for several important trade-offs. On one hand, by releasing earlier, the firm's product is available for a longer time in the market before it becomes outmoded, the firm reduces its cost of detecting and fixing software bugs as a result of error reporting, and the firm may even enjoy competitive benefits associated with being first to market (Cohen et al. 1996). On the other hand, earlier-released software bears a lower initial quality, which has several effects: users incur costs associated with security attacks and poor application performance in the interim; users, in turn, impose goodwill costs on the firm; and the speed at which consumers adopt the software, i.e., the rate of diffusion, is reduced (Keizer 2007).

These aforementioned trade-offs will be strongly influenced by the distinct characteristics of the firm, its product, and the corresponding market. One particularly relevant characteristic of a software firm is its software process maturity, which is "the extent to which a specific process is explicitly defined, managed, measured, controlled, and effective" (Paulk et al. 1993, p. 4). The federally funded Software Engineering Institute (SEI) established a set of Capability Maturity Models (CMMs) to help organizations improve processes with an emphasis on those related to software development. Some of these basic models were superceded by SEI's Capability Maturity Model Integration (CMMI), but the essence is much the same. CMMI defines five levels of maturity in software-producing organizations that range from Level 1, which is characterized by ad hoc processes that can complete goals, to Level 5, where processes are already rigorously defined and managed in a quantitative way and the focus now lies on optimizing them (Software Engineering Institute 2006). One useful aspect of the CMMI is that firms at the same level tend to be similar on other dimensions as well. For example, Table 1 illustrates the relationship between a firm's software process maturity and the quality of its software, measured in terms of defects per thousand lines of code. This positive relationship between process maturity and quality as well as other software development characteristics has also been established in the academic literature (see, e.g., Harter et al. 2000, Harter and Slaughter 2003). Hence, by understanding how bug density interacts with the above-mentioned trade-offs that affect software release, we can provide broad implications on how firms at a given level of process maturity should manage software adoption and even identify the value that underlies achieving a higher level of maturity.

Table 1 **The Relationship Between the Average Number of Defects per Thousand Lines of Code (KLOC) and an Organization's Level Achievement in the Capability Maturity Model**

	CMM Level 1	CMM Level 2	CMM Level 3	CMM Level 4	CMM Level 5
Defects/KLOC	7.5	6.24	4.73	2.28	1.05

Source. Davis and Mullaney (2003).

A second important factor that affects how a software firm manages adoption through its release timing and price is the amount of software functionality built into a product. One observation is that, over time, an increasing amount of functionality is being included with each new version of a given software product. For example, Microsoft Windows versions NT 4.0, 2000 Professional, XP Professional, and Vista have 16, 29, 40, and 50 million lines of code, respectively (Henry 2007, Manes 2007). Additionally, different classes of software can be correlated with different amounts of functionality. For example, another mainstream operating system software, Red Hat Linux 7.1, which was released around the same

time as Windows XP, had a comparable 30+ million lines of code (Wheeler 2001). Popular open source database software products PostgreSQL and MySQL are estimated to have approximately 900,000 lines of code each (Babcock 2008, MST 2010).[2]

In this paper, using an analytical model, we formally examine how a software firm adjusts its release timing and pricing in a setting where it can harness customer error reporting. In this context, we specifically study the influence of the firm's pricing power, its software process maturity, and the characteristics of its product class, as well as its relative quality within that class on its timing and pricing decisions.

2. Literature Review

This study is directly related to the literature on optimal software release timing. A vast portion of the literature studies the firm's strategy from a maintenance cost minimization perspective, independent of software demand. Examples include Okumoto and Goel (1980), Koch and Kubat (1983), Yamada and Osaki (1987), Kimura et al. (1999), Pham and Zhang (1999), and Zheng (2002). Several studies (see, e.g., McDaid and Wilson 2001, Ji et al. 2005, Jiang et al. 2012) extend this framework and incorporate opportunity costs of lost sales due to late release through a price-invariant cost component that increases in the time to market. More recently, Arora et al. (2006) implement a total sales function that depends on release time and patching levels. One common characteristic of extant models in the literature is that the expected evolution of software adoption over time does not impact the firm's software release decision, and we relax this assumption in our work. Whereas some more general studies on innovation release (e.g., Kalish and Lilien 1986) do account for the shape of the adoption curve, we further advance our understanding of this topic by focusing on the software industry and its idiosyncrasies, including quality improvement dissemination via patching, negligible reproduction costs, and, importantly, customer error reporting.

Aiming to clarify and study the trade-off between internal testing and debugging costs on one hand, and revenues, goodwill penalties, and consumer error reporting benefits on the other, we complement the existing literature on optimal software release modeling by employing a continuous-time parameterization of software demand and incorporating quality and network effects on adoption. The shape of the adoption curve impacts a firm's profit in multiple ways. First, it directly affects revenues. Second, it impacts consumers' aggregate contribution to bug detection. Many past studies on software release examine a setting where the firm ceases to test the software for flaws after bringing it to market and implicitly incorporate consumers' participation to error reporting by assuming a bug fixing cost associated with flaws discovered in the field of operation, which, in the absence of postrelease in-house detection efforts, are reported only by users. Recently, Ji et al. (2005) and Jiang et al. (2012) explicitly incorporate consumer error reporting, and we extend their work by further incorporating the effect of software demand on this dimension. Specifically, we allow the consumers' contribution to the bug detection rate to grow with the installed base and parameterize the actual error reporting rate to account for users opting out of crash reporting because of privacy and other concerns.

Third, there is an explicit link between the cost to firms of addressing postrelease software bugs and how the network size evolves over time, which is another important contribution of our work. Prior models typically consider these costs as being linear in the number of bugs that are detected after market introduction. Some studies account for it more abstractly as a function of time (Shantikumar and Tufekci 1983), reliability (Pham and Zhang 1999), or the number of remaining flaws (Ji et al. 2005). Ehrlich et al. (1993) also introduce a cost to the software firm resulting from consumer use that depends on the software failure intensity at release, the usage period, and an exogenous demand that is independent of the model parameters. Beyond postrelease bug processing and fixing costs, we further account for the fact that the firm also incurs *goodwill costs* at a rate that is proportional to both the current bug count and network size. In this manner, we are able to capture the fact that an error that is detected early after release, when there are few adopters, is likely to be less costly to the firm compared with an error that resides in the code longer and generates greater damage to a larger consumer base. Over time, the network size increases and the number of resident errors decreases, generating an important dynamic with the goodwill cost rate and how the firm controls adoption.

3. The General Model

A firm offers a software product licensed for perpetual use and supports it until discontinuation time $T > 0$.[3] Beyond T, consumers who have already

[2] The two studies use different metrics; for MySQL, a measure of the effective lines of source code was utilized.

[3] In the rapidly evolving software industry, this discontinuation time is often exogenously determined by the rate of technological advance. In other cases, because of future major releases of a product that are developed in parallel, the discontinuation time can be the result of planned functional obsolescence. For simplicity, in our model, we take T as fixed and focus attention on how community contributions (error reporting) toward quality improvement affect a software producer's release timing and pricing.

purchased may continue to use the software, which still has value, but the firm ceases all quality improvement efforts.[4] The software product's extent of features and overall complexity is related to its software class, ranging from enterprise software to simple end-user applications. We denote the software product's complexity as $Y > 0$, which can be interpreted as the size of the product's codebase or its number of basic units of code (e.g., modules, functions). Stemming from the CMMI and its implementation in industry, a standard measure of software quality is the inverse of defect density, i.e., defects per thousand lines of code (Weszka 2003, Siemens Information Systems Ltd. 2003). We assume that the software product has $\bar{B}$ bugs or defects at the earliest moment it can feasibly be released, denoted as $t = 0$, which is often considered to be the beginning of the *release candidate stage*, where features are no longer being added and the focus turns to final testing and debugging (Petreley 1998). Therefore, using this standard measure, initial software quality is given by $Y/\bar{B}$.[5]

We denote the software process maturity of the firm as $\gamma \in [0, 1]$. Recent empirical studies (Harter et al. 2000, Krishnan et al. 2000, Harter and Slaughter 2003, Harter et al. 2012) document a positive relationship between process maturity and software quality and further identify in which circumstances the effect of maturity is relatively even greater. Following both industry evidence and convention in the literature, we also assume this relationship, $\bar{B}/Y = \sigma(\gamma)$, exists, where we use defect density for convenience and $\sigma(\gamma)$ has the following properties: $\partial\sigma/\partial\gamma < 0$, $\sigma(0) = \bar{\sigma} < \infty$, and $\sigma(1) = 0$. Taken together, firms with more mature software development processes (higher γ) arrive at the release candidate stage with lower bug density (i.e., $\bar{B}/Y$ is decreasing in γ). Both the product's life cycle, $0 < T < \infty$, and the market potential, $0 < m(Y) \leq \bar{m} < \infty$, are finite, the latter being weakly increasing in the complexity or level of functionality provided by the software.

Let $D(t)$ denote the number of unique bugs detected and reported by time $t \geq 0$, with the initial condition $D(0) = \bar{D} < \bar{B}$. Analogous to the relationship between $\bar{B}$ and Y, we assume that previously detected bugs satisfy $\bar{D} = \rho\sigma(\gamma)Y$, where $0 \leq \rho < 1$. We assume that the bug detection process satisfies properties of the mean of the classic nonhomogeneous Poisson process model for software reliability in Goel and Okumoto (1979), whereby at any given time t, the rate at which previously undiscovered bugs are detected is proportional to the number remaining in the code.[6] In particular, the detection process is given by

$$\frac{\partial D(t)}{\partial t} = \theta(t) \times (\bar{B} - D(t)), \tag{1}$$

where $\theta(t)$ can be interpreted as the overall rate of detection per undiscovered bug.[7] Both the firm and existing users can contribute to the bug detection process. The firm contributes $\theta_f > 0$ to the overall rate by incurring testing effort, and any adopter who chooses to provide quality feedback contributes $\theta_u(Y) \in (0, \bar{\theta}_u]$. We assume consumers use the software uniformly over time and across functions and that their usage varies with the software's level of functionality Y, which, in turn, affects the detection rate. Because consumer error reporting is usually optional and some users elect not to participate because of concerns over privacy and possible work disruption (Muthitacharoen and Saeed 2009), we denote the portion of users who do participate as $\alpha \in (0, 1]$.

Denoting the time at which the software is released as t_0, we make a simplifying assumption that the firm ceases detection at t_0 and that all contributions to detection come from the user base going forward. Thus, if $N(t)$ denotes the number of existing users at time t, then the overall detection rate per undiscovered bug is given by

$$\theta(t) = \begin{cases} \theta_f & \text{if } t < t_0, \\ \alpha N(t)\theta_u(Y) & \text{if } t \geq t_0. \end{cases} \tag{2}$$

As a result, $\theta(t)$ is time inhomogeneous and can increase substantially after t_0 due to adoption. That is, user-supplied detection rates, $\alpha N(t)\theta_u(Y)$, can exceed θ_f as adoption increases. Analogous to the benefit of increasing eyeballs with open source software development to find bugs, early release strategies benefit the firm by leveraging an increased number of testers (Raymond 1999).

[4] In particular, consumers no longer impose goodwill costs on the firm after T. In reality, consumers could still impose some goodwill penalties on the firm after this point in time, but these costs would be limited in comparison to those imposed as a result of poor quality due to software defects during the active life of the product. Beyond T, most consumers have moved on to either newer versions of the product or possibly different technologies, which further limits any goodwill costs.

[5] Initial bug density, the inverse of initial software quality, is thus $\bar{B}/Y$.

[6] This assumption is widely supported and built on in other models (see, e.g., Okumoto and Goel 1980, Ehrlich et al. 1993, Pham and Zhang 1999, Jiang et al. 2012). Although other models of bug detection do exist, we follow the above tradition while utilizing a deterministic variant that maintains tractability and focus; such an approach is typical when the stochastic nature of the bug detection process is not critical to the research questions being studied (see, e.g., Ji et al. 2005, 2011; Arora et al. 2006).

[7] This model implicitly captures bug heterogeneity with regard to detection. Starting with $\bar{B}$ bugs initially, some bugs will surface quickly whereas others will take longer to be detected; some will even go undetected for the entire planning horizon. Furthermore, software complexity (Y) affects both $\bar{B} = \sigma(\gamma)Y$ and $\theta(t)$ through its effect on user contributions $\theta_u(Y)$, as seen in (2).

We denote the number of bugs still resident (either undetected or detected but unfixed) in the code as $B(t)$ with the initial condition $B(0) = \bar{B}$. Once a bug is reported, it is assigned to a pool of bugs that have been detected but not yet fixed. At time t, this pool contains $D(t) - (\bar{B} - B(t))$ reported defects, and the firm works toward addressing them by issuing patches at a rate ξ per unfixed defect. Hence, the rate of change for remaining bugs in the code is given by

$$\frac{\partial B(t)}{\partial t} = -\xi \times (D(t) - \bar{B} + B(t)). \quad (3)$$

We implicitly capture the property that bugs are likely to be heterogeneous in the amount of effort required for each to be resolved (Giger et al. 2010). In particular, by the dynamics in (3), once detected, some bugs will be patched fast, others will take longer to be fixed, and some known flaws will not be removed from the code prior to product discontinuation.

Drawing on the vast literature on innovation diffusion sparked by Bass's (1969) model, we parameterize the evolution of the cumulative installed base of users $N(t)$ through a continuous-time hazard rate model as follows:

$$\frac{\partial N(t)}{\partial t} = (m(Y) - N(t))\left(a + b\frac{N(t)}{m(Y)} - c\frac{B(t)}{Y}\right)w(p), \quad (4)$$

over $t \in [t_0, T]$, where $a > 0$. Adoption is influenced by both positive network effects associated with the software and quality effects determined by its reliability.[8] The relative strength of the network effects is given by $b > 0$. Similarly, we denote the extent to which poor software reliability affects adoption as $c > 0$. For example, in recent years, Microsoft's release of Service Pack 2 (SP2) for Windows XP was plagued by compatibility issues that led to a significantly slower rate of adoption (Rooney 2004, Oswald 2005). Similarly, when Microsoft released Vista a few years later, its product suffered from instability and, in some cases, slower performance than XP, which again led to sluggish adoption (O'Neill 2008).

Consumer price sensitivity is reflected by a multiplicative price response function $w(p)$ satisfying (i) $w(p) > 0$, (ii) $w'(p) < 0$, (iii) $\lim_{p\to\infty} pw(p) = 0$, and (iv) $w''(p) \geq 0$. Our choice of a multiplicatively separable price effect is consistent with the literature on the diffusion of innovation (see, e.g., Robinson and Lakhani 1975, Kalish 1983, Bass et al. 1994, Krishnan et al. 1999, Sethi and Bass 2003). The third condition implies that for any level of functionality above a certain price point, both revenues and adoption are negligible. As an example, price response functions of the form $\tau e^{-p\phi}$ with $\tau, \phi > 0$ satisfy all assumed properties on $w(\cdot)$.[9]

The firm incurs four types of costs: testing, error report processing, bug fixing, and goodwill, which we discuss in order. First, we assume that the firm broadly tests the entire codebase, which is to say it does not know a priori where the defects are located. The firm will incur a total cost rate of $C_T \times \theta_f \times Y$, where $C_T \geq 0$, to induce a bug detection rate (resulting from the firm's effort) of $\theta_f \times (\bar{B} - D(t))$ through the testing of all Y units of the codebase. Thus, the total testing cost is given by

$$TC = C_T \times \theta_f \times Y \times t_0. \quad (5)$$

Second, we assume that the firm incurs error processing costs associated with analyzing error reports, assessing the extent of any given defect, and assigning its reparation to an appropriate development team. Bug triaging is far less automated and relies on developers and project managers to actually examine reports and classify them. For example, in a recent effort to reduce memory leaks in Firefox, Mozilla launched an initiative called MemShrink, part of which involves weekly bug triage meetings (Keizer 2011). User-generated reports usually vary in the quality of information provided and may require additional effort in terms of interpreting a report and reproducing an error (Hooimeijer and Weimer 2007, Zimmermann et al. 2010). For these reasons, we denote the per-report processing costs as $C_{P,f} \geq 0$ and $C_{P,u} \geq 0$, depending on whether the report originates from the firm's testing or users' reporting, respectively, with $C_{P,f} < C_{P,u}$. Contemporary automated error reporting systems can help reduce the gap between these two cost rates but do not completely eliminate it (Glerum et al. 2009). In light of consumer protection laws and potential liability, we assume that the firm processes all error reports (Kennealy 2000, Cusumano 2004, Otto 2009). The total processing cost is given by

$$PC = C_{P,f}(D(t_0) - \bar{D}) + C_{P,u}(D(T) - D(t_0)). \quad (6)$$

Third, the firm incurs quality improvement costs by allocating effort to resolve bugs that have been reported but are still unfixed at a cost rate $C_F \geq 0$. For simplicity, we assume that the firm works concurrently on fixing all $D(t) - \bar{B} + B(t)$ defects. By (3), some fixes are issued quickly, whereas others become delayed. As a result, the firm can receive duplicate error reports that must be cross-checked with the bug tracking system at a cost $C_D \geq 0$ before being discarded. For example, in one study's data, researchers

[8] Network effects are captured in various models applied to IT products and services (see, e.g., Zhang and Seidmann 2010, Niculescu et al. 2012, Dou et al. 2013).

[9] This functional form is used by Robinson and Lakhani (1975).

found that it can take over 20 minutes to recognize that a bug report is a duplicate (Cavalcanti et al. 2010). While pending resolution, any detected but unresolved bug can still be redetected (sometimes with substantially different symptoms) at a rate $\theta(t)$, resulting in a duplicate report cost rate of $C_D \times \theta(t) \times (D(t) - \bar{B} + B(t))$. Therefore, the total bug fixing cost is given by

$$FC = \int_0^T (C_F + \theta(t)C_D) \times (D(t) - \bar{B} + B(t))\, dt. \quad (7)$$

Finally, the firm incurs goodwill costs from exposing its users to buggy software. Software crashes typically generate damage to users only when they occur. However, not all flaws lead to crashes or system alerts. For example, security breaches can go undetected without any system interruption, and malicious hackers can exploit security vulnerabilities repeatedly before they are detected and patched. More broadly, what we call "goodwill costs" can be thought of as the cost of quality, which may even include the cost of helping users recover from software failure. We denote the average cost incurred per each user and unit of time as $C_G \times B(t)/Y$, where $C_G \geq 0$, and the total expected goodwill costs are given by

$$GC = \int_{t_0}^T C_G \times N(t) \times \frac{B(t)}{Y}\, dt. \quad (8)$$

Because software is a digital good, we make the traditional assumption that there are no capacity constraints and the marginal cost of reproduction is 0. Given that our time frame starts where all functionality has already been coded, development costs are considered sunk. Revenues are generated at each point in time when a user adopts the product; hence, the firm's profit can be written as

$$\Pi(t_0, p) = pN(T) - (TC + PC + FC + GC). \quad (9)$$

Using the framework laid out in this section, we can begin to explore how firms of varying process maturity and that offer different classes of software products should manage adoption through release timing and pricing.

Throughout the paper and proofs, for clarity in exposition, we will simplify notation for $\bar{B}(\gamma, Y)$, $\bar{D}(\gamma, Y)$, $m(Y)$, $\theta_u(Y)$, and $w(p)$ by omitting the arguments (i.e., using $\bar{B}$, $\bar{D}$, m, θ_u, and w, respectively) whenever the arguments are not relevant to the discussion or analysis at hand. To avoid trivialities, we focus on parameter regions where the firm yields profits above a minimum, positive value. Furthermore, for simplicity, we assume no discounting, but all of the insights presented in this paper extend to a case with discounting.[10]

[10] See Arora et al. (2006) and Jiang et al. (2012) for a similar assumption in cases where discounting does not play a central role.

4. Managing Adoption and Error Reporting

Prior to software release at t_0, the dynamics of the detected and resident bugs, $D(t)$ and $B(t)$, respectively, can be characterized as follows. By (1) and (2), the number of unique bugs that have been detected by the firm by time $t \in [0, t_0]$, whether fixed or not, is given by

$$D(t) = \bar{B} + (\bar{D} - \bar{B}) \times e^{-\theta_f t}. \quad (10)$$

Using (3) and (10), the number of bugs still remaining in the software, either undetected or detected but not yet fixed, exhibits the following trajectory over time:

$$B(t) = \frac{1}{\xi - \theta_f} \times (\xi(\bar{B} - \bar{D})e^{-\theta_f t} + (\bar{D}\xi - \bar{B}\theta_f)e^{-\xi t}) \quad (11)$$

for $0 \leq t \leq t_0$.[11]

After software release at t_0, adoption can begin, and by (4), a necessary condition is that the hazard rate must be positive; i.e., $a \geq cB(t_0)/Y$. Using (11) and the relationship between software maturity and initial bug density, this condition is equivalent to $a \geq c\sigma(\gamma)\Gamma(t_0)$, where

$$\Gamma(t) \triangleq \frac{1}{\bar{B}(\xi - \theta_f)} \times (\xi(\bar{B} - \bar{D})e^{-\theta_f t} + (\bar{D}\xi - \bar{B}\theta_f)e^{-\xi t}).$$

Because $\Gamma(t)$ is decreasing and $\Gamma(0) = 1$, adoption can commence at $t = 0$ if $a \geq c\sigma(\gamma)$. Otherwise, because $\lim_{t\to\infty} \Gamma(t) = 0$, there exists a unique bound $\tilde{t}$ satisfying $a = c\sigma(\gamma)\Gamma(\tilde{t})$ such that adoption can start provided $t_0 \geq \tilde{t}$. In summary, the release time must satisfy

$$t_0 \geq L(\gamma) \triangleq \begin{cases} 0 & \text{if } a \geq c\sigma(\gamma), \\ \tilde{t} & \text{otherwise,} \end{cases} \quad (12)$$

where $L(\gamma)$ is a constraint on release time similar to other constraints on software reliability/quality seen in papers such as Yamada and Osaki (1987), Kimura et al. (1999), and Zheng (2002). $L(\gamma)$ is decreasing; software firms with higher maturity can release their products earlier.

The characterization of $D(t)$ and $B(t)$ over $t \in [0, t_0]$ in (10) and (11) provides initial conditions to the dynamical system over $t \in [t_0, T]$:

$$\begin{aligned} \frac{\partial D(t)}{\partial t} &= \alpha N(t)\theta_u(Y)(\bar{B} - D(t)), \\ \frac{\partial B(t)}{\partial t} &= -\xi(D(t) - \bar{B} + B(t)), \\ \frac{\partial N(t)}{\partial t} &= (m(Y) - N(t))\left(a + b\frac{N(t)}{m(Y)} - c\frac{B(t)}{Y}\right)w(p), \end{aligned} \quad (13)$$

[11] In the remainder of this paper, without loss of generality, we restrict attention to $\theta_f \neq \xi$. For the special case where $\theta_f = \xi$, $B(t) = (\bar{B} + \xi(\bar{B} - \bar{D})t)e^{-\xi t}$, and the subsequent analysis is similar.

where $D(t_0) = \bar{B} + (\bar{D} - \bar{B})e^{-\theta_f t_0}$, $B(t_0) = (\xi(\bar{B} - \bar{D}) \cdot e^{-\theta_f t_0} + (\bar{D}\xi - \bar{B}\theta_f)e^{-\xi t_0})/(\xi - \theta_f)$, and $N(t_0) = 0$.

Because of how the bug detection, fixing, and adoption processes interact, the ability to fully analytically characterize the dynamical system described in (13) is limited, though it can be studied numerically over the complete parameter space. We proceed analytically by focusing on a parameter regime where the quality effect on adoption and the bug fixing rate satisfy certain bounds. First, our focal regime will satisfy $c < \bar{c}$ such that the negative effect of bugs on adoption is not too severe. Consistent with most software products, users often utilize software while cognizant of the quality issues. Adoption is typically not overly hampered and commences, although users certainly impose goodwill costs. Second, $\xi > \underline{\xi}$ will also be satisfied in this regime such that we study cases where bugs are worked on at a reasonable rate.[12] Our model assumes that it is mandatory for firms to process all consumer-reported bugs and resolve them at this rate. Overall, the parameter regime we study is often found in practice, and maintaining focus on it will increase clarity and analytical tractability throughout the paper.[13]

4.1. Exogenous Pricing

First, we study the case where price is fixed over the software product's selling horizon and is determined by the market. Let p be the price of the software for all $t \geq t_0$. Examining the dynamical system in our focal regime, the adoption path exhibits certain monotonicity properties.

Lemma 1. *For all $t \geq t_0$,*

(i) *$N(t)$ is decreasing in t_0 and p.*

(ii) *For a given software firm's process maturity level γ, $N(t)$ is increasing in Y. That is, a firm derives a stronger cumulative adoption from releasing a higher functionality product despite it containing a larger number of bugs.*

Part (i) of Lemma 1 establishes that a delayed release adoption path always lies below an early release adoption path. As a consequence, if the firm ever chooses to delay its release in order to increase initial quality and gain stronger initial adoption momentum, this delayed strategy will never induce the same cumulative sales volume. Part (ii) of Lemma 1 highlights that, all else being equal, the net effect of increased software functionality is greater adoption, despite the existence of more bugs in the software. Examining (4), which governs adoption, we see that an increase in functionality Y increases both the potential market for the software $m(Y)$ and the initial number of bugs $\bar{B}(\gamma, Y) = \sigma(\gamma)Y$. When the negative effect of software quality on adoption is limited, the net effect of a larger market potential is to increase the rate of adoption. Hence, a software firm with a given maturity γ can increase functionality without hurting adoption, provided it maintains this level of capability in its development processes. That is, more bugs certainly get introduced, but additional functionality serves to counterbalance them. For example, holding all else constant, including software maturity, the above result suggests that having both triple the functionality and triple the bugs would still lead to greater adoption; one could compare the adoption of Windows NT 4.0 to Windows 7, which has an estimated three times the former product's lines of code.

4.1.1. Release Time Bounds.

Next, we study the profit maximization problem for the firm as it selects the optimal release time. Given a fixed price p, the optimal release time t_0^* satisfies

$$t_0^* = \arg\max_{t_0 \in [L(\gamma),\, T]} \Pi(t_0).$$

Although it is not possible to give a complete, explicit closed-form solution to this problem, we can (i) characterize informative bounds on the firm's optimal release time, (ii) identify conditions under which t_0^* is either at a bound or interior, and (iii) characterize the conditions t_0^* must satisfy when it is interior.[14] Choosing a release time at a bound can often still generate profits comparable to those obtained under the optimal interior choice for release time, which we will both analytically and numerically demonstrate. For convenience, we define $G_1(\lambda) \triangleq (a+b)/(b + ae^{(a+b)(T-\lambda)w})$, $G_2(\lambda) \triangleq (a+b)(T-\lambda)w/2$, and

$$\begin{aligned} g(\lambda) \triangleq\ & -aG_1^2 e^{(a+b)(T-\lambda)w+\lambda\theta_f} mpwY\alpha\theta_u - C_T e^{\lambda\theta_f} Y^2\alpha\theta_f\theta_u \\ & + \big(C_G + (C_{P,u} - C_{P,f})Y\alpha\theta_u\big) \times (\bar{B} - \bar{D})\theta_f \\ & + (C_G + C_{P,u}Y\alpha\theta_u) \\ & \times \frac{G_1^{m\alpha\theta_u/(bw)}(\bar{B} - \bar{D})e^{(a+b)(T+\lambda)w/2 + am(T-\lambda)\alpha\theta_u/b}}{ae^{(a+b)Tw} + be^{(a+b)\lambda w}} \\ & \times \big(-(a+b)\theta_f \cosh(G_2) \\ & \quad + (\theta_f(b-a) + 2am\alpha\theta_u)\sinh(G_2)\big). \end{aligned}$$

[12] Our regime is consistent with Ji et al. (2011) and Jiang et al. (2012), where identified flaws are assumed to be removed instantaneously.

[13] Because of the nature of (13), $\bar{c}$ and $\underline{\xi}$ are implicit bounds whose existence is rigorously demonstrated in the proofs. Because any closed-form representations of these bounds would be pages long and lack informativeness (and not be attainable in most cases), we numerically establish how wide the parameter region typically can be as we discuss results in the paper. This parameter regime will apply for all propositions.

[14] For both the release time and pricing optimization problems studied in this paper, the profit functions generally have a unique optimizer, although their shapes may not be concave. Our results apply for wide parameter regions as can also be numerically illustrated.

Proposition 1. *The optimal release time for a firm's software product satisfies* $t_0^* \le H(Y, \gamma, p)$ *such that*

(i) *if* $\theta_f \ge m\alpha\theta_u$, *then*

$$H(Y,\gamma,p) = \min\left\{T, \max\left\{L(\gamma), \frac{1}{\theta_f}\log\left(\frac{(\bar{B}-\bar{D})(C_G+(C_{P,u}-C_{P,f})Y\alpha\theta_u)}{C_T Y^2\alpha\theta_u}\right)\right\}\right\};$$

(ii) *if* $\theta_f < m\alpha\theta_u$, *then*

$$H(Y,\gamma,p) = \min\left\{T, \max\left\{L(\gamma), \frac{1}{\theta_f}\log\left(\frac{(\bar{B}-\bar{D})(C_G+(C_{P,u}-C_{P,f})Y\alpha\theta_u)}{C_T Y^2\alpha\theta_u}\right), T-\log\left(1+\frac{(a+b)\theta_f}{a(m\alpha\theta_u-\theta_f)}\right)\Big/((a+b)w)\right\}\right\},$$

and profits are decreasing in t_0 *over the interval* $(H(Y,\gamma,p), T]$. *Further, there exists* $K_\Psi > 0$ *such that* $|t_0^* - \Psi| < K_\Psi\delta$ *whenever* $g(L(\gamma)) > 0$, *where* Ψ *satisfies* $g(\Psi) = 0$.[15]

Proposition 1 establishes that a firm should never delay release beyond $H(Y, \gamma, p)$. Despite potentially inducing higher sales on some time intervals, a delayed release corresponds to a weaker *aggregate* installed base pointwise. Within the interval $(H(Y, \gamma, p), T]$, further delaying release will shrink profits because lost revenues and additional internal testing costs are not offset by the benefits over the remaining horizon associated with lower goodwill costs and bug processing costs. Hence, Proposition 1 asserts that a firm should release immediately if any constraining factors prevented release prior to $H(Y, \gamma, p)$. Software firms sometimes face such constraints when the adoption of their product is tied to the availability of specific hardware. For example, in the spring of 2006, Sony officially announced a delay in the release of its PlayStation 3 console until November due to unresolved issues with the production of Blu-ray components (Nagai 2006). In such a case, some firms who would have originally delayed the release of their titles to improve initial quality may have reduced incentives to delay under the new console launch date.

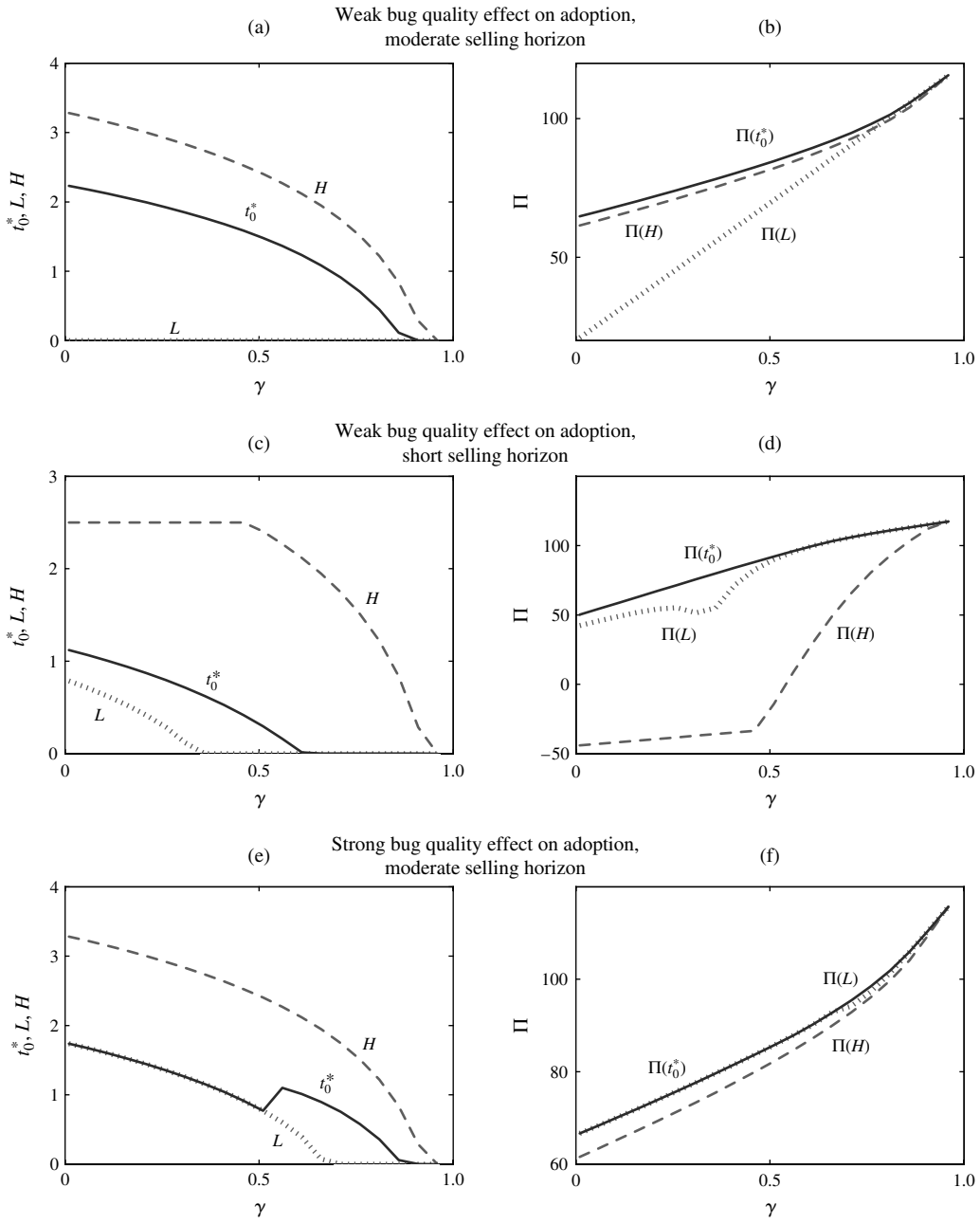
As a firm increases the maturity of its development processes, both bounds on the optimal release time have a tendency to shrink, which suggests that an earlier release may be preferable. Proposition 1 also describes the optimal release time as it moves into the interior and away from the bounds. All together, L, H, and Ψ characterize how the optimal release time is affected by the parameters of the model. In Figure 1, we illustrate how the optimal release time compares to the derived bounds while slightly deviating the parameter set for each panel. Panels (a), (c), and (e) all demonstrate that for sufficiently large γ, the bounds collapse, suggesting that the firm should release its product as soon as it can generate adoption. Said differently, firms that utilize mature development processes (e.g., those that have obtained a higher CMM level) should release their products to the market when they have a release candidate available. Although there are always some negative effects on adoption stemming from reduced quality due to bugs early in the product life cycle, at a high level of maturity, releasing at $L(\gamma)$ to boost adoption through network effects is more beneficial to the firm.

Panel (a) of Figure 1 shows that when the strength of the bug quality effect on adoption is relatively low ($c = 0.4$), L drops to zero, as is implied by (12) for all levels of software maturity. Even a firm whose software maturity is low can feasibly release immediately and still engender adoption, but it optimally chooses to delay release. In this case, the negative effect of low quality on adoption is quite limited, but goodwill costs are substantial because $N(t)$ will ramp up quickly when c is small (see (4) and (8)). In this panel, the optimal release time is decreasing as software maturity (γ) increases, and for sufficiently high maturity, the release time bounds collapse and the firm optimally releases immediately at time zero. Panel (b) demonstrates that the upper bound H performs well in comparison to the optimal interior release time, and profits under the lower bound L approach optimal profits as γ gets large.

Panels (c) and (d) of Figure 1 illustrate a case where the negative quality effect on adoption is still weak but a bit larger ($c = 3$). Additionally, the selling horizon is shortened from $T = 8$ to $T = 2.5$. In this case, there are two noteworthy differences. First, because the negative quality effect on adoption is slightly stronger, at the low range of software maturity, the lower bound L has now increased, moving away from zero. Second, the optimal release time occurs sooner, and the software is released at zero for a significant range of process maturities at the high end. Panel (d) illustrates the corresponding optimal profits, demonstrating that releasing as soon as a release candidate is available at L is nearly optimal. In the following proposition, we analytically examine what happens as

[15] The formal conditions of the focal regime are described in Lemma A1, which provides greater details on how δ should be taken. In particular, $c = \kappa_c\delta$ and $\xi = \kappa_\xi/\delta$, where κ_c, κ_ξ, $\delta > 0$.

Figure 1 Optimal Release Time, Release Time Bounds, and Profits as Affected by Software Maturity, the Impact Strength of Bug Quality on Adoption, and the Length of the Selling Horizon



Notes. For panels (a) and (b), $c = 0.4$ and $T = 8$. For panels (c) and (d), $c = 3$ and $T = 2.5$. For panels (e) and (f), $c = 6$ and $T = 8$. The other parameter values are $a = 4$, $b = 0.5$, $Y = 15$, $m = 30$, $\alpha = 0.25$, $\theta_f = 0.8$, $\theta_u = 0.015$, $C_T = 0.7$, $C_F = 0.7$, $C_{P,u} = 0.6$, $C_{P,f} = 0.5$, $C_G = 0.3$, $C_D = 0.5$, $\xi = 2.3$, $\rho = 0.1$, $\sigma(\gamma) = 2(1-\gamma)$, $p = 4$, and $w = 0.46$.

T becomes shorter, which can provide greater insight into the characteristics seen in panel (c).

Proposition 2. *There exist bounds $\bar{\theta}_u$, $\bar{T} > 0$ with $\bar{T}$ satisfying*

$$0 = -\frac{a(a+b)^2 e^{(a+b)\bar{T}w} mpwY}{(b+ae^{(a+b)\bar{T}w})^2} - C_T Y^2 \theta_f - (\bar{B}-\bar{D}) \times \left(C_{P,f} Y\theta_f - C_G m \times \left(1 - \frac{a\bar{T}\theta_f}{b} + \frac{\theta_f(\log(b+ae^{(a+b)\bar{T}w}) - \log(a+b))}{bw} - \frac{a+b}{b+ae^{(a+b)\bar{T}w}}\right)\right), \quad (14)$$

such that if $a > b$ *and* $\theta_u < \bar{\theta}_u$, *then*

(i) *if* $T < \bar{T}$, *then* $t_0^* = L$;

(ii) *if* $T > \bar{T}$, *then* $t_0^* > L$.

Furthermore, as either software functionality or maturity increases, a firm can release at L *over a wider range of selling horizons. Technically,* $\bar{T}$ *is increasing in* γ (*i.e., decreasing in* $\bar{B} - \bar{D}$) *and also increasing in* Y *whenever* $m'(Y)/m(Y) < 1/Y$.

Proposition 2 formally establishes that there exists a bound on the length of the selling horizon $\bar{T}$ below which a software firm should release at L. Above that bound, the optimal release time moves to the interior. A bound exists for a firm at any level of software maturity. The critical difference is how the bound is affected by software maturity. In Proposition 2, we analytically establish that $\bar{T}$ is increasing in software maturity, which is to say that a highly mature firm should release its software at L for a wider range of selling horizons. Said differently, $\bar{T}$ can be quite large for a firm with mature software development processes. Returning to panel (c) of Figure 1, we illustrate that firms with software maturities satisfying $\gamma \in [0.6, 1]$ already release their software at L when $T = 2.5$. As T decreases further, t_0^* will coincide with L for a greater range.

To contrast this result with what happens as c grows higher, in panel (e) of Figure 1, we use $c = 6$ and $T = 8$. Once the negative quality effects on adoption become more severe, even less mature software firms will find it optimal to release *earlier*, but the reasoning is different in this case. It is not the case that a lower maturity firm should release software products early in the absolute sense. Rather, it should release its product as soon as it feasibly can, because by the time it has a release candidate with sufficiently high quality to induce adoption, the amount of time remaining before product discontinuation has become limited. Panel (e) of Figure 1 illustrates how $L(\gamma)$ can decrease more sharply in software maturity. As γ becomes small, the remaining sales horizon is effectively limited and the firm should optimally release at L. As γ increases, $T - L(\gamma)$ increases and the firm's sales horizon becomes less constrained. In this case, an early release would involve goodwill and processing costs accumulating over a longer period of time for early adopters. Thus, the producer may prefer to delay release in order to improve quality first, which is depicted by t_0^* moving into the interior region between the bounds L and H. However, as γ continues to increase, such a firm's software product has inherently better quality, and the effects described in Proposition 2 dominate, indicating that the firm should release immediately at L.

Finally, Proposition 2 also establishes that $\bar{T}$ increases in Y under market conditions where the market potential is large and not too sensitive to changes in software functionality. Under these conditions and when the per-user contribution rate is low, an increase in functionality provides greater incentive to release earlier and harness customer error reporting. In particular, more bugs are introduced with greater functionality, and shorter sales horizons demand faster bug clearance.

4.1.2. Impact of User Contributions on Release Timing. In this section, we study the sensitivity of the optimal release time with respect to user contributions. First, we examine how an increase in the proportion of users who agree to participate in error reporting affects the optimal release time.

Proposition 3. *The optimal release time* t_0^* *is decreasing in* α *if* $Q_\alpha(\lambda) < 0$ *for all* $\lambda \in [0, \bar{\lambda}]$ *and increasing in* α *if* $Q_\alpha(\lambda) > 0$ *for all* $\lambda \in [0, \bar{\lambda}]$, *where*

$$
\begin{aligned}
&Q_\alpha(\lambda)\\
&\triangleq -bC_G\left(\frac{a+b}{G_1}\right)^{m\alpha\theta_u/(bw)}(ae^{(a+b)Tw}+be^{(a+b)\lambda w})w\theta_f\\
&\quad + e^{G_2+(a+b)w\lambda+am(T-\lambda)\alpha\theta_u/b}\times(a+b)^{m\alpha\theta_u/(bw)}\\
&\quad \times\big((a+b)\theta_f\cosh(G_2)[wbC_G - m\alpha\theta_u\\
&\qquad \times G_3(wa(T-\lambda)+\log G_1)]+\sinh(G_2)\\
&\qquad \times\big[w\big(-b^2C_G\theta_f + a^2m\alpha\theta_u(T-\lambda)G_3(2m\alpha\theta_u-\theta_f)\\
&\qquad\quad + ab(C_G\theta_f(1+m\alpha\theta_u(T-\lambda))\\
&\qquad\quad + C_{P,u}mY\alpha^2\theta_u^2(2+\theta_f(T-\lambda)))\\
&\qquad\quad + m\alpha\theta_u G_3\log(G_1)(\theta_f(b-a)+2am\alpha\theta_u)\big]\big), \quad (15)
\end{aligned}
$$

$G_3 \triangleq C_G + C_{P,u}Y\alpha\theta_u$, *and* $\bar{\lambda}$ *is given in the appendix. Further,*

(i) *there exists a bound* $\bar{v} > 0$ *such that if*

$$\frac{m\alpha\theta_u}{\theta_f} < \frac{(a+b)\cosh((a+b)Tw/2) - (b-a)\sinh((a+b)Tw/2)}{2a\sinh((a+b)Tw/2)}$$

and $C_G/C_{P,u} < \bar{v}$, *then* $Q_\alpha > 0$;

(ii) *there exists a bound* $\underline{v} > 0$ *such that if* $C_G/C_{P,u} > \underline{v}$ *is satisfied, then* $Q_\alpha < 0$.

Proposition 3 demonstrates that the net effect of the interaction between the optimal release time (t_0^*) and the level of error reporting (α) is critically determined by goodwill costs, the cost of processing user-generated error reports, error reporting rates of the firm and user base, bug fixing rates, functionality, and the various parameters describing the adoption curve. In our focal regime, how the release time is impacted by an increase in the error reporting fraction ultimately hinges on the sign of Q_α as defined in (15). For part (i), when the ratio between goodwill costs and the processing costs associated with user-generated error reports ($C_G/C_{P,u}$) is relatively low and

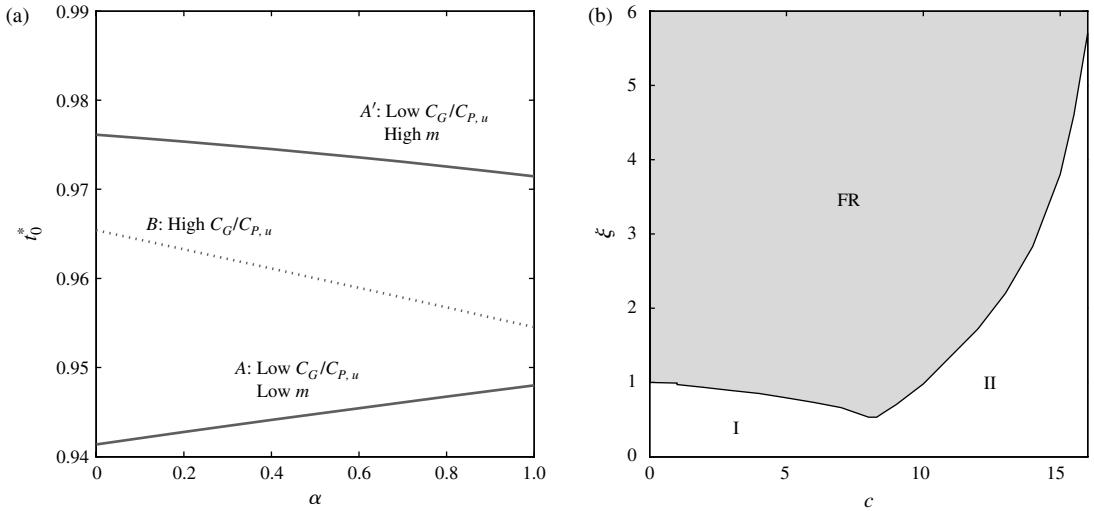
the strength of the error reporting rate from users compared with the firm is also low, then t_0^* increases in response to an increase in α. When goodwill costs are low, the firm has incentives to release its product early on. With an increase in the fraction of users contributing to error reporting, the firm can release even earlier to further harness these contributions, particularly in light of the small goodwill costs. On the other hand, if the cost of processing user reports is large in comparison to goodwill costs and the relative strength of total user error reporting rate relative to the firm's is small, then the firm also has incentives to incur further detection costs itself and release a more polished product later in the market. Part (i) of Proposition 3 establishes that the latter effect of this trade-off is stronger, and the firm may optimally prefer to shift its release time outward in such cases. In panel (a) of Figure 2, we provide a numerical illustration of this behavior with curve A.

In contrast, when the condition in part (i) of the proposition is violated and the relative potential of error reporting ($m\alpha\theta_u/\theta_f$) is high, the former effect tends to dominate, as is illustrated with curve A'. In this case, the inherent market potential is doubled in comparison to that for curve A, which increases the potential of user error reporting considerably. Similarly, part (ii) of Proposition 3 establishes that despite obvious drawbacks of releasing its software early when goodwill costs are high, a software firm will overall prefer to release earlier as the fraction of users participating in error reporting increases, provided that the cost of processing these user reports is not too substantial. In panel (a) of Figure 2, curve B demonstrates how t_0 decreases under these conditions. In this case, it is critical that the firm be able to remedy defects quickly in order to effectively offset the high goodwill costs while benefitting from the increased user error report contributions.

In panel (b) of Figure 2, we depict the extent of the bounds on the quality effect on adoption ($\bar{c}$) and the bug fixing rate ($\underline{\xi}$), which we discussed previously in §4.1 and in Footnote 13. As mentioned, the size of our focal regime (labeled FR in the figure) is extensive, and our results apply for a wide region of the parameter space. In particular, using the same parameters used to generate curve B, we illustrate the extent to which c and ξ can be adjusted while maintaining the same qualitative behavior (i.e., monotonicity) as seen in curve B. When this focal regime is violated, the optimal release time can exhibit varying behavior. For example, in region I, when the bug fixing rate is too slow, the firm has incentives to delay release rather than inducing greater detections that are not expeditiously resolved. For region II, as c becomes large, the software begins losing profitability.

Another dimension of user contribution to the debugging process is captured through the user error detection rate $\theta_u(Y)$, which in turn depends on the level of software functionality Y. Highlighting this interaction, we next explore how the level of software functionality affects the firm's release time decision. To simplify the problem and focus on the most relevant trade-offs, in the following we will hold the market potential constant at the level m; implicitly, we study a region where the market potential is not too elastic in functionality.

Figure 2 **How the Optimal Release Time Changes in the Proportion of Users Who Contribute Error Reports**



Notes. The common parameter values are $a = 4$, $b = 0.5$, $c = 10$, $T = 1.5$, $C_T = 0.25$, $C_F = 0.2$, $C_D = 0.2$, $C_{P,f} = 0.01$, $\bar{\theta}_f = 0.7$, $Y = 3.5$, $m(Y) = m_0(1 - e^{-0.42Y})$, $\theta_u(Y) = 0.1Y^{-0.7}$, $\sigma(\gamma) = 2(1 - \gamma)$, $\gamma = 0.75$, $p = 0.4$, $w(p) = 0.5e^{-0.02p}$, $\rho = 0.1$, and $\underline{\xi} = 2$. For curves A and A', the specific parameter values are $C_G = 1.8$ and $C_{P,u} = 1.2$, with $m_0 = 30$ in the former and $m_0 = 60$ in the latter. For curve B, the specific parameter values are $C_G = 2$, $C_{P,u} = 0.05$, and $m_0 = 30$. Panel (b) illustrates the extent of the bounds on the focal regime, denoted as FR, for curve B.

Proposition 4. *For fixed market potential, the optimal release time t_0^* is decreasing in Y if $Q_Y(\lambda) < 0$ for all $\lambda \in [0, \bar{\lambda}]$, and increasing in Y if $Q_Y(\lambda) > 0$ for all $\lambda \in [0, \bar{\lambda}]$, where*

$$Q_Y(\lambda) = \sigma(\gamma)(1-\rho)e^{-\lambda\theta_f} \times \left(K_G^Y C_G + K_{P,u}^Y C_{P,u} - \theta_f C_{P,f}\right) - \theta_f C_T, \quad (16)$$

and K_G^Y, $K_{P,u}^Y$, and $\bar{\lambda}$ are given in the appendix.

(i) *When the additional cost of processing user-reported bugs is low, then*

(a) *given any level of firm development maturity, there exists $\bar{\theta}_f > 0$ such that if $\theta_f < \bar{\theta}_f$ and $\theta_u'(Y) < 0$, then $Q_Y > 0$ and t_0^* is increasing in Y;*

(b) *if a firm's internal rate of bug detection (θ_f) is not too small and it has highly mature development processes, then $Q_Y < 0$ and t_0^* is decreasing in Y.*

(ii) *When a firm uses less mature development processes, the additional cost of processing user-reported bugs is high, and the goodwill costs are low, then*

(a) *if the impact of increased functionality on the user community's contribution to detection rate is negative (i.e., $Y\theta_u'(Y) + \theta_u(Y) < \bar{\mu} < 0$) and the firm has a relatively high internal detection rate, then $Q_Y < 0$ and t_0^* is decreasing in Y;*

(b) *if the impact is positive and the contributions are large (i.e., $\theta_u'(Y) > 0$ and $\theta_u(Y)$ is high), then $Q_Y > 0$ and t_0^* is increasing in Y.*

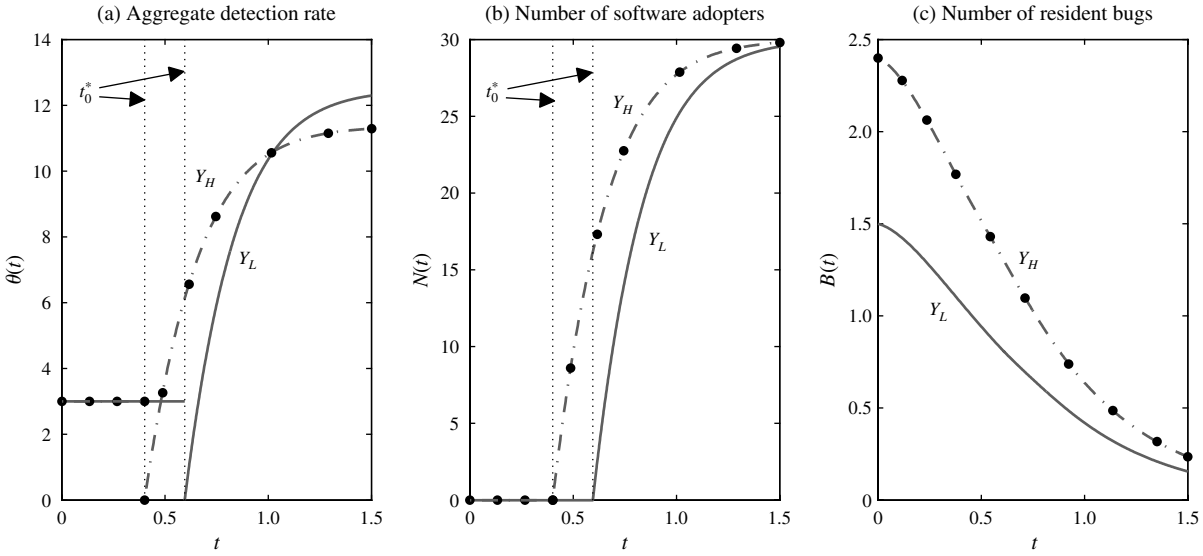
An important measure that helps determine the aggregate effect of a change in functionality on release time is the degree to which this change affects the *total* detection rate contributed per user across all modules; i.e., $\partial Y\theta_u(Y)/\partial Y = Y(\partial\theta_u(Y)/\partial Y) + \theta_u(Y)$. When the impact on user contribution is weak (i.e., $Y(\partial\theta_u(Y)/\partial Y) + \theta_u(Y)$ is small or negative), an increase in functionality leads to either a limited positive increase or a loss in the total user detection rate. There are multiple effects in play. On one hand, the firm has reduced incentives to release its product early to the market because of the lower benefit associated with user contributions. Instead, the firm can shift its release to a later time to avoid goodwill costs associated with early release. On the other hand, because there is a cost associated with processing user-reported defects, having fewer contributions from the user community may permit the firm to release earlier to engender faster adoption benefitting from network effects while incurring less additional processing costs.

Part i(a) of Proposition 4 establishes that when the cost of processing user-reported defects is small, then the optimal release time can be delayed with an increase in functionality. Referring to the tradeoffs identified above, because the user contributions are weak, the firm can benefit from pushing back the release time such that goodwill costs are not incurred while the firm continues testing on its own to improve the quality of its software by reducing defects. Moreover, even in cases when testing costs are large, having a lower internal rate of detection will limit the firm's exposure to these costs and permit a later release. Part ii(a) of Proposition 4 examines the impact of a loss in strength of user contributions as a result of increased functionality when the firm has lower software maturity and user error report processing costs are higher. Beyond a certain point, increased complexity may impact the usability of a system and reduce the chances of users spotting individual defects. Because of greater flaws due to lower maturity, a decrease in user reporting can permit earlier release without incurring as much total additional processing costs because of these reduced user contribution rates. In this case, releasing earlier can expedite adoption and be preferable as long as goodwill costs are limited in comparison to processing costs.

When the impact on user contribution is positive and contributions are large, an increase in functionality boosts the total detection rate stemming from users. In this case, there is again a distinct tradeoff. First, there are incentives to release the software earlier in order to benefit from stronger user contributions. On the other hand, the firm also wants to limit both goodwill costs stemming from early release and potential additional costs associated with the processing of user-reported defects. Part i(b) of Proposition 4 establishes that when firms have higher development maturity, and correspondingly fewer bugs in their products, they can release earlier to benefit from more user contributions as long as processing costs are not too high. High maturity and lower processing costs together limit the downside costs, making it optimal to decrease release time as a consequence to increased functionality. Notably, K_G^Y in (16) is negative when $\theta_u'(Y) > 0$, suggesting that stronger user contributions stemming from increases in functionality can extend this effect to a greater range of software maturity levels. On the other hand, part ii(b) of Proposition 4 formalizes the opposite effect: when a firm has lower software maturity, it may find it preferable to delay release to limit exposure to the additional processing and goodwill costs stemming from the higher bug density found in its product.

In Figure 3, we illustrate how the optimal release time responds to an increase in software functionality from $Y_L = 2.5$ to $Y_H = 4$. In this case, $C_{P,u}$ is relatively low and θ_f is at a higher level relative to per-user error reporting contributions; by part i(b) of Proposition 4, the optimal release time should decrease in response. In panel (a), we plot the overall detection rate $\theta(t)$ over time, which shows t_0^* dropping from 0.60 under Y_L down to 0.40 under Y_H. Panel (b)

Figure 3 Optimal Release Time as Influenced by Changes in Software Functionality



Notes. The common parameter values are $a = 10$, $b = 0.5$, $c = 5$, $T = 1.5$, $\alpha = 0.25$, $C_F = 0.7$, $C_D = 0.5$, $C_{P,f} = 0.01$, $C_{P,u} = 0.02$, $C_T = 1.5$, $C_G = 2$, $\theta_f = 3$, $m = 30$, $\theta_u(Y) = 2Y^{-0.2}$, $\sigma(\gamma) = 2(1-\gamma)$, $\gamma = 0.7$, $w(p) = 0.5e^{-0.02p}$, $p = 1.7$, $\rho = 0.1$, and $\xi = 2$. For the two curves depicting lower and higher levels of functionality, the parameter values are $Y_L = 2.5$ and $Y_H = 4$, respectively.

reflects the impact of this change on the adoption curve $N(t)$. Consistent with Lemma 1, the adoption curve under a lower t_0 and higher Y is greater everywhere, as can be seen in the figure. Panel (c) illustrates how bugs are resolved over time. Because of the product's higher level of functionality, under Y_H there are more defects that need to be detected and resolved. Releasing earlier allows the firm to better harness customer error reporting in order to improve software quality over the selling horizon.

Connecting our findings with what we observe in industry, we discuss a recent example. The Apple iOS 5 mobile operating system was released relatively quickly after its previous iteration and initially contained flaws and preconfigured settings that severely diminished handset battery performance (Albanesius 2011). Although Apple does not sell its mobile operating system separately to customers, the goodwill impact of poor quality can be high given its complementary handset and application (app) market businesses. Apple uses mature software development processes and takes a structured approach to quality assurance, using many channels to receive user feedback. Moreover, a great number of Apple iOS 5's incremental new features were aimed at enhancing user experience and facilitating the development of apps on its platform. Both of these classes of enhancements are likely to have generated a significant amount of attention and usage from their targeted communities, leading to potentially substantial feedback in case of software flaws. This setting fits well with part (ii) of Proposition 3 and part i(b) of Proposition 4, both suggesting that Apple had incentives to release earlier, which was confirmed ex post.

4.1.3. Beta Testing. In this section, we examine how beta testing interacts with the firm's release time decision. Because the focus of our paper centers on release timing and pricing in the context of user error reporting contributions, we abstract from initial software development decisions taking a software product as given (i.e., a release candidate is available at time zero). In the typical software release cycle, beta testing is a precursor to the release candidate stage, and therefore issues such as when to cease development (quality choice) and begin alpha/beta testing become more relevant. In that light, in this section, we do not intend to provide a comprehensive understanding of how to manage beta testing. Instead, we hope to provide some basic insights into how the existence and degree of beta testing might impact a firm's software release timing, which is a fundamental concern in our study.

We take a simple view of beta testing where the firm has the capability to harness a portion of the potential market $N_B \in (0, m)$ to serve as testers of the product during the interval $[0, t_0]$ before release. In effect, beta testing uses potential customers to boost the detection rate while not incurring a significant cost of quality (i.e., goodwill cost) because the product still has yet to be released. At the same time, beta testing may push up the cost of processing user-reported bugs by increasing the rate at which these detection reports arrive to the development team. In the following proposition, we explore how the extent of beta

testing affects the firm's incentives to either expedite or delay the release of its software.

Proposition 5. *Suppose both the internal and external bug detection rates and the strength of network effects are not too large. Then,* (i) *if a software firm has highly mature development processes and the cost of processing user-reported bugs is small, it will tend to optimally delay its product release as the size of the beta testing group increases;* (ii) *if the cost of processing user-reported bugs is large, a firm will instead tend to release earlier.*

Part (i) of Proposition 5 formalizes that when a firm with high process maturity produces a high-quality product and there are not many defects in the code, it should delay software release as the beta test size increases. When there are few defects in the code, the firm will not incur significant additional processing costs as a result of user-reported bugs. However, it incurs testing costs at each moment while searching for these defects. As the size of the beta test increases, the firm faces a trade-off: it can release earlier to reduce its own testing cost while harnessing a larger beta testing population for a shorter period of time or release later to utilize the larger beta testing population for a longer period of time, which improves quality and reduces goodwill costs but also leads to higher testing costs. For a high-maturity firm, the trade-off described above tilts toward the latter, and it prefers a delayed release strategy. For example, Blizzard Entertainment, a producer of very popular PC games, has long been known for the high quality of their products. Recently, they were in the beta testing phase of the third edition of one of their most successful role-playing games. During that phase, they announced that the beta test would be extended and the size of the beta testing group would also be increased, two moves that should be performed in lockstep for a high-maturity producer as suggested by part (i) of Proposition 5 (Hachman 2011).

On the other hand, when the processing of user error reports is more costly, part (ii) of Proposition 5 formally establishes that the firm can end beta testing and release its software earlier because (1) the greater-sized beta testing population improves the quality of the software faster, enabling earlier release; and (2) releasing the software temporarily relieves the incurrence of processing costs, and adoption ramps up, giving the firm an opportunity to fix previously discovered defects and reduce goodwill costs. This highlights the trade-off the firm faces between lower testing costs and processing costs associated with increased feedback from adopters; higher processing costs incentivize the firm to focus more on lowering testing costs and increasing revenues through an earlier release. Although firms of varying maturity are similarly affected when user error reporting processing costs are higher, a firm with lower maturity is more sensitive because its product is characterized by a much higher defect density. In particular, such a firm both needs more help from the user testers and has adoption that exhibits slower growth. Thus, the applicability of part (ii) of Proposition 5 occurs for a wider range of processing costs; i.e., $C_{P,u}$ can be much smaller for lower-maturity firms and still induce this net effect.

4.2. Optimal Pricing

In the previous section, we examined a software firm's optimal release timing and profitability when it utilizes release time as its primary lever to manage adoption and error reporting. Next, we explore a setting where a software firm also has some pricing power, optimally selecting a single price to charge throughout the selling horizon of the product. There are many instances where software firms prefer to set price at an optimal level and keep it to a large extent at that level for the entire selling horizon. For example, over the past several years, the AutoCAD suite and DivX video software bundle have been priced at $3,995 and $19.99, respectively, for a perpetual license, whereas WinEdt's noncommercial single-user license has been priced at a steady $40 over the last decade. In this section, our aim is to better understand what role each lever plays to manage adoption and harness the benefits of error reporting. We denote the optimal price chosen by the firm as p^*, which together with the optimal release time satisfy

$$(t_0^*, p^*) = \underset{t_0 \in [L(\gamma), T],\, p>0}{\arg\max}\ \Pi(t_0, p).$$

If error contributions from users are large and goodwill costs are low, then implications from Proposition 1 expand to firms of all maturity; i.e., they tend to release immediately. When a firm releases its software at the earliest time possible, i.e., $t_0^* = L(\gamma)$, it uses a two-pronged strategy. First, immediate release jump-starts adoption and permits error reporting feedback earlier in the software's life cycle. Second, the firm finds it profitable to price the software (and, implicitly, shape adoption) in a manner that can help prevent overexposure to lower initial quality. In this section we explore properties of the optimal price in such scenarios. First, we examine the link between price and the proportion of users α who report bugs.

Proposition 6. *Suppose that the cost of processing user error reports is not too high. Then,*

(i) p^* *decreases in* α *for software produced by a high-maturity firm when* $C_G/C_{P,u}$ *is high, and*

(ii) p^* *increases in* α *for software produced by a lower-maturity firm with low goodwill costs and intermediate user error detection rates* (*i.e.,* $C_G/[Y(C_T/((1-\rho)\bar{\sigma}) - (C_{P,u} - C_{P,f}))] < \alpha\theta_u(Y) < b/((a+b)mT)$).

For part (i) of Proposition 6, when software is produced by a highly mature firm, it has fewer bugs at the release candidate stage. Given that the firm is performing extensive testing across the entire codebase, if it can efficiently harness user contributions to help detect the bugs that remain, it has incentives to reduce its own internal testing. This strategy can be achieved by releasing early and pricing in a balanced way such that the installed base leads to a significant reduction in testing costs without severely impacting revenue. The higher the user error reporting rate, the lower the optimal price is, because a larger resulting installed base is an effective substitute to internal testing; detection costs are outsourced to the user base without compromising quality. This effect is strongest when goodwill costs are high because, in this case, the potential benefits of faster quality improvement by harnessing users are the largest. Second, the cost associated with processing user error reports should not be too high such that leveraging users remains economical.

Although our paper is normative in nature, we briefly present some suggestive evidence that our finding here can be supported. Over time, Microsoft developed and improved Windows Error Reporting to harness consumer bug detections toward improving quality (Glerum et al. 2009). Microsoft also released successive generations of products such as Office Professional (i.e., 2000, XP, 2003, 2007, and 2010). The suggested retail prices of these versions of Office Professional, as announced by Microsoft, were \$599, \$579, \$499, \$499, and \$499, respectively, and for the 2010 version, an alternative option to purchase the product key card instead was made available for \$349 (Microsoft 2001, 2007; Arar 2003; Meredith 2010). Given that Microsoft has high software maturity, part of its decreasing price trend over versions can be explained by the substantial increase in error reporting and its strategic outsourcing of its testing costs.

In contrast, for part (ii) of Proposition 6, when quality is lower (i.e., higher bug density), if the firm releases early, it has the potential to incur significant goodwill costs and processing costs associated with user reports. Clearly, a firm with low process maturity charges less than a firm with high process maturity because of its reduced-quality product. However, when the user error reporting rate increases, a lower-maturity firm prefers to restrict early adoption by raising its price slightly when goodwill costs are relatively limited in comparison to the processing costs. Here, the level of user error reporting should be within an intermediate range such that a change in the level has a stronger impact on the processing costs that are incurred. In particular, when the user error reporting rate is lower, an increase to the rate and its corresponding effect on processing costs can be mitigated with a small increase in price while not significantly impacting goodwill costs and revenues. Consistent with traditional monopolistic settings, an increase in the firm's costs per user is compensated through a higher price that allows the firm to retain high margins by serving in the beginning a reduced mass of adopters for whom its software is presumably most critical while inducing a delay for other adopters until quality further improves through bug detection and resolution.

Proposition 7. *For fixed market potential, suppose that the processing costs associated with user error reports are not too high.*

(i) *When the firm has highly mature development processes and goodwill costs are relatively high,*

(a) *if increased functionality decreases the user community's contribution to the detection rate (i.e., $\theta_u'(Y) < 0$), then p^* is increasing in Y;*

(b) *if it increases the rate (i.e., $\theta_u'(Y) > 0$), then p^* is decreasing in Y.*

(ii) *When the firm has less mature development processes and goodwill costs are small relative to the costs associated with processing user error reports (i.e., $C_G/C_{P,u}$ is relatively low),*

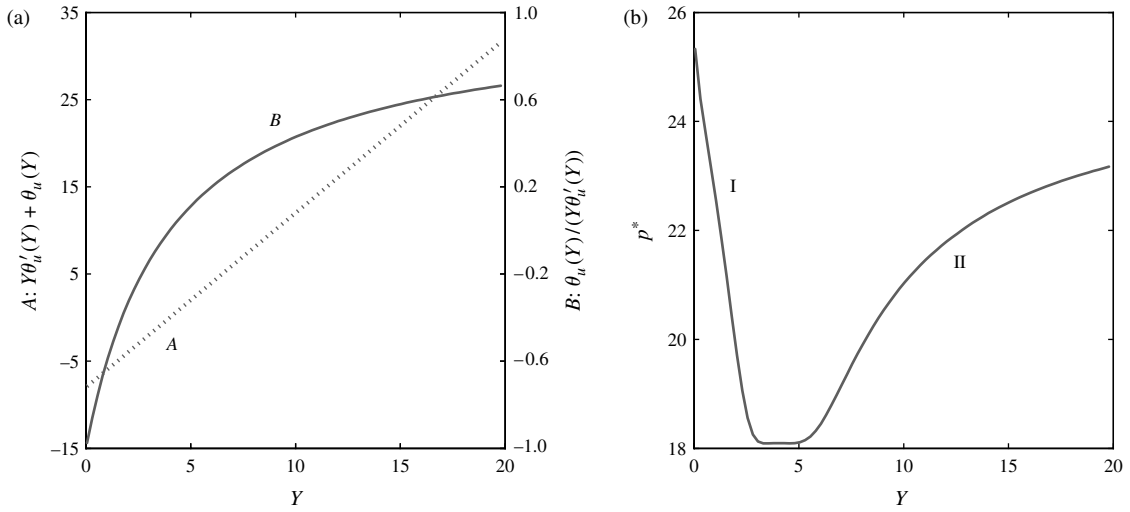
(a) *if the user detection rate is low and the impact of increased functionality on the total contribution is negative (i.e., $Y\theta_u'(Y) + \theta_u(Y) < \bar{\kappa} < 0$), then p^* is decreasing in Y;*

(b) *if the user detection rate is relatively strong compared with the marginal effect of increased functionality on the rate (i.e., $\theta_u(Y)/(Y\theta_u'(Y)) > \underline{\kappa} > 0$), then p^* is increasing in Y.*

Proposition 7 explores how pricing varies with functionality. When the firm has high maturity, as in part (i), an increase in functionality is associated with relatively fewer additional bugs. Above we found that when quality is high, reducing testing costs and leveraging a larger installed base tends to dominate increased processing costs. Nevertheless, when the impact of increased functionality on user detection rates is negative and tends to limit the firm's ability to leverage adopters, the increase in goodwill costs stemming from these additional flaws is the predominant factor. Thus, the firm instead prefers to price higher, serve a smaller installed base, and limit goodwill costs. Alternatively, if consumer usage rates increase considerably with greater functionality and lead to an increase in detection rates per bug, then the firm will find it profitable to leverage the testing efforts of users despite higher processing and goodwill costs. In this case, the firm marks down its price to boost adoption, as we establish in part i(b) of Proposition 7.

When process maturity is lower, the effect of how user contributions respond to a change in functionality has a much different impact on the firm's pricing. Because of the magnitude of bugs, any benefits from

Figure 4 Shape of Optimal Price Path in Software Functionality for a Firm with Lower Process Maturity



Notes. The parameter values are $a = 6$, $b = 0.5$, $c = 10$, $\sigma(\gamma) = 2(1-\gamma)$, $\gamma = 0.73$, $w(p) = e^{-0.1p}$, $T = 4$, $\theta_l = 0.25$, $\xi = 3$, $\alpha = 0.09$, $m = 10$, $C_T = 0.1$, $C_G = 0.1$, $C_D = 0.05$, $C_F = 0.05$, $C_{P,l} = 0.5$, $C_{P,u} = 1.0$, $\theta_u(Y) = Y - 8 + 16Y^{-1}$, and $Y \in [0.05, 20]$.

leveraging users toward detecting bugs tend to be dominated by losses associated with processing user reports and goodwill costs. An increase in functionality will exacerbate these costs. However, if such an increase inhibits user error detection (i.e., its net effect is negative, as seen in the left-hand portion of curve A in panel (a) of Figure 4), then the coupled reduction in processing costs can shift the balance so that the firm can benefit by marginally increasing the installed base. This scenario is illustrated in the left-hand portion (labeled I) of the p^* curve in panel (b) of Figure 4. Here, the firm slightly reduces its optimal price to stimulate adoption and generate more savings from outsourced detection, as is established in part ii(a) of Proposition 7. In contrast, when an increase in functionality strengthens user contributions to detection, processing costs then increase, and the firm has incentives to increase price, thus limiting exposure to both processing and goodwill costs. In panel (a), curve B becomes larger as Y increases and satisfies part ii(b) of Proposition 7 for sufficiently large Y. As a consequence, the optimal price increases in Y, which is illustrated in the right-hand portion of the price curve (labeled II) in panel (b).

5. Concluding Remarks

With the recent, rapid penetration of broadband Internet access, software firms and their customers now interact at a higher frequency and in more complex ways than before, especially after a product is released to the market. This modern structure enables firms to address software quality issues by leveraging both in-house developers and user/community resources. In this paper, we present a model of software adoption to develop an understanding of how firms, which vary in development characteristics such as software process maturity as well as in product characteristics including class and functionality, should optimally adapt their release timing and pricing in order to harness the potential of customer error reporting. We study this issue in-depth in both a setting where the firm has limited ability to control price and a setting where it can optimally set its price.

In the former setting, we characterize relevant bounds on a firm's software release time and relate these bounds to the firm's process maturity and its product's level of functionality. We then explore how consumer feedback affects adoption dynamics and, in turn, the firm's associated revenues and costs.

From our model and results, we generate several testable implications for future research. First, we find that both a shorter product life cycle and a higher software process maturity can induce a firm to release earlier in time. However, a firm possessing lower process maturity and offering a product with a longer life cycle has incentives to delay its release. Second, when the additional processing cost stemming from user bug reports is not too high in comparison to goodwill costs, a firm should release its product earlier when the proportion of users agreeing to participate in error reporting increases. Firms with higher processing costs relative to goodwill costs and limited user contributions should delay release instead. Third, we find that whether a firm delays release in response to added functionality critically depends on the extent to which detection contributions coming from the community adjust in response to this functionality. Importantly, the nature of the effect seems to be opposite depending on whether a firm has high or low process maturity. For example, if increased functionality tends

to dilute per-module detection contributions, then a highly mature firm should further delay release as it increases functionality, whereas a less mature firm has greater incentives to expedite release and harness these contributions earlier. Fourth, we find similar results related to beta testing. As the size of the beta test increases, firms with higher process maturity tend to delay. Finally, when a firm can set a single, optimal price, we find that comparative statics of the proportion of user participation in error reporting and software functionality on the firm's price align well with our comparative statics on release time (i.e., p^* moves in the same direction as t_0^*, for similar reasons).

In this paper, we construct a model while making some simplifying assumptions for tractability. First, for simplicity, we assume that debugging is perfect, which is to say that new bugs are not introduced into the code as part of the debugging process. Although the introduction of new bugs can certainly happen, it is generally true that software quality improves over time because of patching, even with this small risk. Our results in this paper would be quite robust to imperfect debugging, provided that debugging is mostly effective. Second, we take a simpler view that patches are released as soon as the vendor has a patch available; in reality, when patches are released (either willingly or by threat of disclosure) and when they are applied (if ever) by users is a complex topic in its own right and extensively studied in the vulnerability disclosure literature (see August and Tunca 2006, 2008, 2011; Cavusoglu et al. 2007, 2008; Arora et al. 2008). One possible future extension could be to explore this link between vulnerability disclosure and a vendor's product release time and pricing.

There are a number of directions to build on our model and results. One interesting extension is to examine bugs of varying degrees of severity. In this case, the firm would remediate the varying classes of bugs at different rates. In a preliminary numerical study, we found that the firm indeed spends more energy on fixing more severe bug classes, but it also concurrently delays release as the proportion of bugs in the severe class increases. Another direction to explore is studying software producers who choose to offer small functionality updates in addition to security patches over time. Although extensive functionality updates would generally be reserved for the next major release, one can explicitly study a firm's decision on how much functionality to build into a product by release, how much additional functionality to add to it later while already released, and when to release the next version; all of these decision problems can be grouped together as part of an integrated study on upgrade cycles. One limitation of our study is that we focus strictly on release time and price. More broadly, a firm may undertake strategies to adjust the level of user error reporting, such as temporarily throttling contributions from users when detected but unresolved bugs become backlogged and resources are limited. Developing an understanding of this broader decision problem is an interesting direction for future research.

Since we employ a diffusion model, demand is aggregated at the market level, and we do not lay out the decision-making process of individual consumers. That being said, a subsequent extension to our work can be to take a utility-based approach and incorporate customer error reporting, release timing, pricing, and adoption. Although making analytical progress with a continuous-time model in this type of approach would be difficult, such an extension can yield valuable additional insight into the way a software market is strategically grown. Finally, future studies might also examine competition and subscription-based revenue schemes.

Acknowledgments

The authors thank Lorin Hitt (the department editor), the associate editor, and the anonymous reviewers for their helpful suggestions throughout the review process. The authors also thank Özge Islegen, Sunil Kumar, David Lowsky, Noam Shamir, Hyoduk Shin, Seungjin Whang, Gad Allon, and Sandra Slaughter; the participants at the INFORMS 2011 Annual Meeting and the Workshop on Information Systems and Economics 2011; as well as the participants in research seminars at Stanford University, the University of North Carolina at Chapel Hill, the University of Michigan at Ann Arbor, London Business School, INSEAD, Singapore Management University, Hong Kong University of Science and Technology, Georgia Institute of Technology, the University of Texas at Dallas, the University of Southern California, the University of Minnesota, and Northwestern University for their discussions and comments on various drafts of this paper. This material is based on work supported by the National Science Foundation [Grant CNS-0954234].

Appendix. Proofs of Propositions

Before proceeding with the proofs of the proposition statements, we start by proving a few useful lemmas. First, we define the following quantities:

$$g_D(t) \triangleq \bar{B} - (\bar{B} - \bar{D})(b + ae^{(a+b)(t-t_0)w})^{-m\alpha\theta_u/(bw)} \times e^{-t_0\theta_f + (m\alpha\theta_u/(bw))(a(t-t_0)w + \log(a+b))}, \tag{17}$$

$$g_B(t) \triangleq \bar{B} - g_D(t), \tag{18}$$

$$g_N(t) \triangleq m\left(1 - \frac{a+b}{b + ae^{(a+b)(t-t_0)w}}\right). \tag{19}$$

Lemma A1. *Suppose* $c = \kappa_c\delta$ *and* $\xi = \kappa_\xi/\delta$, *where* κ_c, κ_ξ, $\delta > 0$. *Then, there exists* $\bar{\delta} > 0$ *and functions* $M_D(t)$, $M_B(t)$, $M_N(t) > 0$ *such that if* $\delta < \bar{\delta}$, *then* $g_D(t) - D(t) < \delta M_D(t)$, $B(t) - g_B(t) < \delta M_B(t)$, *and* $g_N(t) - N(t) < \delta M_N(t)$ *for* $t_0 < t \le T$.

Proof of Lemma A1. To begin, we examine an adapted dynamical system with state trajectories denoted by $D_a(t)$, $B_a(t)$, and $N_a(t)$. For this adapted system, bugs are resolved immediately upon detection, i.e., $B_a(t) = \bar{B} - D_a(t)$, and there are no negative quality effects on adoption such that $D_a(t)$ and $N_a(t)$ solve

$$\frac{\partial D_a(t)}{\partial t} = \alpha N_a(t)\theta_u(Y)(\bar{B} - D_a(t)),$$
$$\frac{\partial N_a(t)}{\partial t} = (m(Y) - N_a(t))\left(a + b\frac{N_a(t)}{m(Y)}\right)w(p).$$

It is straightforward to show that $D_a(t) = g_D(t)$, $B_a(t) = g_B(t)$, and $N_a(t) = g_N(t)$, as defined in Equations (17)–(19), solve this adapted dynamical system. Next, we show that the difference between this solution and the solution to the dynamical system in (13) is $O(\delta)$ as δ becomes small. First, we examine how the adoption process would behave if none of the bugs had been resolved yet to obtain a lower bound on the real adoption process $N(t)$. We denote this process as $\grave{N}(t)$, which satisfies (4) with $B(\cdot) = \bar{B}$ in that equation. Given t_0, we denote the fraction of the total market potential that adopted before t with $F(t) \triangleq \grave{N}(t)/m(Y)$. Then, if we define $f(t \mid t_0) \triangleq (\partial F/\partial t)(t \mid t_0)$, by (4), it follows that $f(t)/(1 - F(t)) = (a - c\bar{B}/Y + bF(t))w$, with $F(t_0) = 0$. This is a Riccati partial differential equation with solution $F(t) = 1 + 1/\nu(t)$, where $\nu(t) = e^{w(a-c\bar{B}/Y+b)t}(-e^{-w(a-c\bar{B}/Y+b)t_0} + wb\int_{t_0}^{t} e^{-w(a-c\bar{B}/Y+b)k}\,dk)$. Hence, we obtain $\grave{N}(t) = 0$ if $t < t_0$ and $\grave{N}(t) = mF(t)$ if $t \geq t_0$. In particular, for $t \geq t_0$, after simplifications,

$$\grave{N}(t) = m\left(1 - \frac{a - c\bar{B}/Y + b}{b + (a - c\bar{B}/Y)e^{(a-c\bar{B}/Y+b)(t-t_0)w}}\right). \tag{20}$$

By (20), as δ becomes small, $\grave{N}(t)$ satisfies

$$\grave{N}(t) = g_N(t) + \frac{\kappa_c \delta \bar{B} m(b - (b + a(a+b)(t-t_0)w)e^{(a+b)(t-t_0)w})}{(b + ae^{(a+b)(t-t_0)w})^2 Y} + O(\delta^2) \tag{21}$$

for $t \geq t_0$. Following a similar logic, we define $\acute{N}(t)$ as satisfying (4) with $B(\cdot) = 0$ to obtain an upper bound for $N(t)$. By (19) and (20), it follows that

$$\acute{N}(t) = g_N(t). \tag{22}$$

Suppose $\hat{B}(t) \geq \tilde{B}(t)$ for all $t \in [t_0, T]$. We will establish that $\hat{N}(t \mid \hat{B}(\cdot))$ and $\tilde{N}(t \mid \tilde{B}(\cdot))$ solving (4) satisfy $\hat{N}(t \mid \hat{B}(\cdot)) \leq \tilde{N}(t \mid \tilde{B}(\cdot))$. Defining $\Delta(t) \triangleq \tilde{N}(t \mid \tilde{B}(\cdot)) - \hat{N}(t \mid \hat{B}(\cdot))$, this inequality is equivalent to $\Delta(t) \geq 0$. Suppose toward contradiction that $\Delta(\tilde{t}) < 0$ for some $\tilde{t} \in (t_0, T]$. Because $\Delta(t_0) = 0$, $\Delta'(t_0) \geq 0$, and $\Delta(\cdot)$ is continuous, $\hat{t} \triangleq \sup\{t \in (t_0, \tilde{t}) \mid \Delta(t) = 0\}$ exists. However, because $\Delta'(\hat{t}) \geq 0$, $\hat{t}$ cannot be the supremum. Hence, there is a contradiction, and we conclude that $\Delta(t) \geq 0$, and thus $\hat{N}(t \mid \hat{B}(\cdot)) \leq \tilde{N}(t \mid \tilde{B}(\cdot))$ for all $t \in [t_0, T]$. It immediately follows that $N(t)$ solving (13) satisfies $\grave{N}(t) \leq N(t) \leq \acute{N}(t)$, and by (21) and (22), there also exists $M_N(t) > 0$ such that $(g_N(t) - N(t))/M_N(t) < \delta$ as δ becomes small.

By (1) and (2), $D(t)$ satisfies $\partial D(t)/\partial t = \alpha N(t)\theta_u \times (\bar{B} - D(t))$ for $t \in [t_0, T]$. Solving for $D(t)$, we obtain

$$D(t) = \bar{B} - (\bar{B} - D(t_0))e^{-\int_{t_0}^{t} \alpha\theta_u N(s)\,ds}, \tag{23}$$

from which we can explore how an arbitrary adoption process $\check{N}(t)$ affects detection $\check{D}(t)$ conditional on that adoption process. From this relationship, we can provide bounds on the actual detection process $D(t)$. By (23), it becomes clear that if $\check{N}_1(t) \geq \check{N}_2(t)$ for all $t \in [t_0, T]$, then $\check{D}(t \mid \check{N}_1(t)) \geq \check{D}(t \mid \check{N}_2(t))$. Because $\grave{N}(t) \leq N(t) \leq \acute{N}(t)$, it follows that $\check{D}(t \mid \grave{N}(t)) \leq D(t) \leq \check{D}(t \mid \acute{N}(t))$. By (20) and (23), we obtain

$$\check{D}(t \mid \grave{N}(t)) = \bar{B} - (\bar{B} - \bar{D})(\bar{B}c - (a+b)Y)^{m\alpha\theta_u/(bw)} \times \left(-bY + (\bar{B}c - aY)e^{(t-t_0)w((a+b)Y-\bar{B}c)/Y}\right) \times e^{-t_0\theta_f + m\alpha\theta_u(t-t_0)(aY-\bar{B}c)/(bY)}. \tag{24}$$

By (24), as δ becomes small, $\check{D}(t \mid \grave{N}(t))$ satisfies

$$\check{D}(t \mid \grave{N}(t)) = g_D(t) - \delta K_D(t) + O(\delta^2), \tag{25}$$

where

$$K_D(t) \triangleq \frac{\bar{B}\kappa_c m(\bar{B} - \bar{D})e^{-t_0\theta_f + am\alpha\theta_u(t-t_0)/b}}{(ae^{(a+b)tw} + be^{(a+b)t_0 w(p)})w} \times (Y(a+b))^{-1+m\alpha\theta_u/(bw)}\alpha\theta_u \times (Y(b + ae^{(a+b)(t-t_0)w}))^{-m\alpha\theta_u/(bw)} \times \left(-e^{(a+b)tw} + e^{(a+b)t_0 w}(1 + (a+b)(t-t_0)w)\right).$$

Similarly, for $\check{D}(t \mid \acute{N}(t))$, we obtain

$$\check{D}(t \mid \acute{N})) = g_D(t). \tag{26}$$

Thus, by (25) and (26), and because $\check{D}(t \mid \grave{N}(t)) \leq D(t) \leq \check{D}(t \mid \acute{N}(t))$, we conclude that there exists $M_D(t) > 0$ such that $(g_D(t) - D(t))/M_D(t) < \delta$ as δ becomes small.

Finally, solving (3), $B(t)$ satisfies

$$B(t) = e^{-\xi(t-t_0)}B(t_0) + e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(\bar{B} - D(s))\,ds; \tag{27}$$

hence, if $\check{D}_1(t) \geq \check{D}_2(t)$ for all $t \in [t_0, T]$, then $\check{B}(t \mid \check{D}_1(t)) \leq \check{B}(t \mid \check{D}_2(t))$. Because $\check{D}(t \mid \grave{N}(t)) \leq D(t) \leq \check{D}(t \mid \acute{N}(t))$, it follows that $\check{B}(t \mid \check{D}(t \mid \acute{N}(t))) \leq B(t) \leq \check{B}(t \mid \check{D}(t \mid \grave{N}(t)))$. By (26) and (27), we obtain

$$e^{-\xi(t-t_0)}B(t_0) + e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(\bar{B} - g_D(s))\,ds \leq B(t) \leq e^{-\xi(t-t_0)}B(t_0) + e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(\bar{B} - \check{D}(s \mid \grave{N}(s)))\,ds,$$

which upon subtraction can be rewritten as

$$0 \leq B(t) - e^{-\xi(t-t_0)}B(t_0) - e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(\bar{B} - g_D(s))\,ds \leq e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(g_D(s) - \check{D}(s \mid \grave{N}(s)))\,ds. \tag{28}$$

By (25) and the right-hand side of (28), there exists $\bar{K}_D > \sup_{t\in[t_0,T]} K_D(t)$ such that as δ becomes small,

$$e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(g_D(s) - \check{D}(s \mid \grave{N}(s)))\,ds \leq e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}\delta\bar{K}_D\,ds = \bar{K}_D\delta(1 - e^{-(\kappa_\xi/\delta)(t-t_0)}) \leq \bar{K}_D\delta. \tag{29}$$

Examining the middle expression in (28), and integrating by parts, we obtain

$$\begin{aligned} &B(t) - e^{-\xi(t-t_0)}B(t_0) - e^{-\xi t}\int_{t_0}^{t} \xi e^{\xi s}(\bar{B} - g_D(s))\,ds \\ &\quad = B(t) - \bar{B} + g_D(t) + (\bar{B} - B(t_0) - g_D(t_0))e^{-(\kappa_\xi/\delta)(t-t_0)} \\ &\qquad - \int_{t_0}^{t} e^{-(\kappa_\xi/\delta)(t-s)} g_D'(s)\,ds. \end{aligned} \tag{30}$$

By (28), (29), and (30), it follows that as δ becomes small, there exists $M_B(t)$ such that $B(t) - g_B(t) < \delta M_B(t)$, which completes the proof. □

Lemma A2. *$aw(T-t_0) + \log((a+b)/(b+ae^{(a+b)(T-t_0)w})) < 0 < (a+b)w(T-t_0) + \log((a+b)/(b+ae^{(a+b)(T-t_0)w}))$ for all $t_0 \in [0, T)$.*

Proof of Lemma A2. *Left-hand inequality*: Let $h(b) = aw(T-t_0) + \log(a+b) - \log(b+ae^{(a+b)(T-t_0)w})$. By differentiation, we obtain

$$h'(b) = \frac{1}{a+b} - w(T-t_0) - \frac{1 - bw(T-t_0)}{b + ae^{(a+b)(T-t_0)w}}.$$

Note that $h'(b) < 0$ if and only if $-1 + e^{(a+b)(T-t_0)w}(1 - (a+b)(T-t_0)w) < 0$. Defining $x = (a+b)(T-t_0)\cdot w > 0$, $h'(b) < 0 \Leftrightarrow -1 + e^x(1-x) < 0$. Defining $j(x) = -1 + e^x(1-x)$, we obtain $j'(x) = -xe^x < 0$ for all $x > 0$. Because $j(0) = 0$, it follows that $j(x) < 0$ for all $x > 0$; hence, $h'(b) < 0$. Similarly, $h(0) = 0$; therefore $h(b) < 0$ for all $b > 0$, which proves the result.

Right-hand inequality: We have

$$\frac{\partial(h(b) + bw(T-t_0))}{\partial w} = \frac{b(a+b)(T-t_0)}{b + ae^{(a+b)(T-t_0)w}} > 0.$$

Moreover, $(h(b) + bw(T-t_0)|_{w=0} = 0$. Thus, $h(b) + bw\cdot(T-t_0) > 0$. □

Proof of Lemma 1. (i) *Monotonicity of N with respect to t_0.* Let δ be chosen such that the conditions of Lemma A1 are satisfied. Then, by (19) and Lemma A1, it follows that $N(t) = m(1 - (a+b)/(b+ae^{(a+b)(t-t_0)w})) + \delta\hat{g}_N(t) + O(\delta^2)$. By Lemma A1, $\hat{g}_N(t)$ (and corresponding terms $\hat{g}_D(t)$ and $\hat{g}_B(t)$) can be shown to satisfy

$$\begin{aligned} \frac{\partial \hat{g}_N(t)}{\partial t} &= -\frac{\kappa_c w(\bar{B} - g_D(t))(m - g_N(t))}{Y} \\ &\quad + \frac{w\hat{g}_N(t)(m(b-a) - 2bg_N(t))}{m}, \\ \frac{\partial \hat{g}_D(t)}{\partial t} &= -\alpha\theta_u(\hat{g}_D(t)g_N(t) + \hat{g}_N(t)(g_D(t) - \bar{B})), \\ \hat{g}_B(t) &= -\frac{g_B'(t) + \kappa_\xi \hat{g}_D(t)}{\kappa_\xi}, \end{aligned} \tag{31}$$

by taking a Taylor's series expansion of (13) and subsequently equating terms, where $\hat{g}_N(t_0) = 0$ and $\hat{g}_D(t_0) = 0$. System (31) shows how c and ξ affect $D(t)$, $B(t)$, and $N(t)$ through κ_c and κ_ξ. Differentiating $N(t)$, we obtain

$$\begin{aligned} \frac{\partial N}{\partial t_0}(t \mid t_0, \cdot) &= -\frac{amw(p)(a+b)^2 e^{(a+b)(t-t_0)w}}{(b + ae^{(a+b)(t-t_0)w})^2} \\ &\quad + \delta\frac{\partial \hat{g}_N}{\partial t_0}(t \mid t_0, \cdot) + O(\delta^2) < 0 \end{aligned}$$

for small enough δ.

Monotonicity of N with respect to p. Let us consider (0<) $p_1 < p_2$. Then we have $w(p_1) > w(p_2)$. At t_0, we have $N(t_0 \mid t_0, p_1, \cdot) = N(t_0 \mid t_0, p_2, \cdot) = 0$ and $\partial N/\partial t(t_0 \mid t_0,\ p_1, \cdot) > \partial N/\partial t(t_0 \mid t_0, p_2, \cdot)$. Therefore, in the vicinity of t_0, $N(t \mid t_0, p_1, \cdot) > N(t \mid t_0, p_2, \cdot)$. Defining $\Delta_{N,p}(t \mid p_1,\ p_2) \stackrel{\Delta}{=} N(t \mid t_0, p_1, \cdot) - N(t \mid t_0, p_2, \cdot)$, we have $\Delta_{N,p}'(t_0 \mid p_1, p_2) > 0$ and $\Delta_{N,p}(t_0 \mid p_1, p_2) = 0$.

Suppose that at some point $\infty > t_1 > t_0$, the curves cross each other. Let $t_1 = \inf\{t \mid t > t_0 \text{ and } \Delta_{N,p}(t \mid p_1, p_2) = 0\}$. Then, $\Delta_{N,p}(t) > 0$ for $t \in (t_0, t_1)$. Since after release all detections come from users and one adoption path strictly dominates another at all points, obviously more bugs get detected under p_1 than p_2, and thus, more bugs are also getting fixed (since bugs are getting fixed at a constant rate out of the pool of known but still resident bugs); i.e., $B(t \mid p_1) \le B(t \mid p_2)$. Consequently, it is easy to see that $\Delta_{N,p}'(t_1 \mid p_1, p_2) > 0$. Thus, if the curves cross at t_1, then it must be the case that there exists ϵ such that $\Delta_{N,p}(t \mid p_1, p_2) < 0$ for $t \in (t_1 - \epsilon, t_1)$. This is a contradiction. Thus, $N(t \mid t_0, p_1, \cdot) > N(t \mid t_0, p_2, \cdot)$ for all $t > t_0$.

(ii) There are two cases. First, suppose $t = t_0 + K_t\delta$, where $K_t > 0$. By Lemma A1 and (31), differentiating $N(t)$, we obtain $\partial N(t)/\partial Y = awK_t m'(Y)\delta + O(\delta^2)$. Hence, $\partial N(t)/\partial Y > 0$ for sufficiently small δ. On the other hand, suppose $t > t_0$, and t does not converge to t_0 as δ becomes small. In this case, we similarly obtain $\partial N(t)/\partial Y = m'(Y)(1 - (a+b)/(b+ae^{(a+b)(t-t_0)w})) + \delta(\partial\hat{g}_N(t)/Y) + O(\delta^2)$, yielding the same conclusion. Thus, there exists $\bar{\delta}$ such that if $\delta < \bar{\delta}$, $N(t)$ is increasing in Y. □

Proof of Proposition 1. Substituting (10) and (11) into (9) yields

$$\begin{aligned} &\Pi(t_0, p) \\ &= pN(T) - C_{P,f}(\bar{B} - \bar{D})(1 - e^{-\theta_f t_0}) \\ &\quad - \frac{(C_F + \theta_f C_D)(((\bar{B}\theta_f - \bar{D}\xi)/\xi)(1 - e^{-t_0\xi}) - (\bar{B} - \bar{D})(1 - e^{-t_0\theta_f}))}{\theta_f - \xi} \\ &\quad - C_T\theta_f Y t_0 - \int_{t_0}^{T} (C_F + \alpha\theta_u N(t) C_D)(D(t) - \bar{B} + B(t))\,dt \\ &\quad - \int_{t_0}^{T} N(t)\left(\frac{C_G B(t)}{Y} + C_{P,u}\alpha\theta_u(\bar{B} - D(t))\right)dt. \end{aligned} \tag{32}$$

Differentiating (32), we compute the derivative as

$$\begin{aligned} &\frac{\partial \Pi(t_0, \cdot)}{\partial t_0} \\ &= p\frac{\partial N}{\partial t_0}(t \mid t_0, \cdot)\Big|_{t=T} - \frac{C_{P,f}\theta_f(\bar{B} - \bar{D})}{e^{\theta_f t_0}} \\ &\quad - \frac{(C_F + \theta_f C_D)e^{-t_0(\theta_f + \xi)}(-(\bar{B} - \bar{D})e^{t_0\xi}\theta_f + e^{t_0\theta_f}(\bar{B}\theta_f - \bar{D}\xi))}{\theta_f - \xi} \\ &\quad - C_T\theta_f Y + \int_{t_0}^{T} \frac{\partial N(t \mid t_0)}{\partial t_0}\Big(-C_{P,u}\alpha\theta_u(\bar{B} - D(t)) - \alpha\theta_u C_D(D(t) \\ &\qquad - \bar{B} + B(t)) - \frac{C_G B(t)}{Y}\Big)\,dt \\ &\quad - \int_{t_0}^{T} \frac{\partial D(t \mid t_0)}{\partial t_0}(C_F + \alpha\theta_u N(t)(C_D - C_{P,u}))\,dt \\ &\quad - \int_{t_0}^{T} \frac{\partial B(t \mid t_0)}{\partial t_0}\left(C_F + \alpha\theta_u N(t) C_D + \frac{C_G}{Y}N(t)\right)dt. \end{aligned} \tag{33}$$

Setting $c = \kappa_c\delta$ and $\xi = \kappa_\xi/\delta$ as in the conditions of Lemma A1 and substituting the result from Lemma A1 into (33), it follows that for sufficiently small δ,

$$\begin{aligned}\frac{\partial\Pi(t_0,\cdot)}{\partial t_0} &= \frac{1}{Y\alpha\theta_u}\times\Big[-aA_1^2e^{2A_2+t_0\theta_f}m(Y)pwY\alpha\theta_u - C_Te^{t_0\theta_f}Y^2\alpha\theta_f\theta_u \\ &\quad + (\bar B-\bar D)\theta_f(C_G+(C_{P,u}-C_{P,f})Y\alpha\theta_u) \\ &\quad + \frac{A_1^{1+(m\alpha\theta_u)/(bw)}(\bar B-\bar D)(C_G+C_{P,u}Y\alpha\theta_u)e^{A_2+(am(T-t_0)\alpha\theta_u)/b}}{a+b} \\ &\quad \times(-(a+b)\theta_f\cosh(A_2)+(-(a-b)\theta_f+2am\alpha\theta_u) \\ &\quad \times\sinh(A_2))\Big]+O(\delta), \qquad (34)\end{aligned}$$

where $A_1 = G_1(t_0)$, $A_2 = G_2(t_0)$. First, note that $C_Te^{t_0\theta_f}\cdot Y^2\alpha\theta_f\theta_u > (\bar B-\bar D)\theta_f(C_G+(C_{P,u}-C_{P,f})Y\alpha\theta_u)$ is satisfied if and only if $t_0 > (1/\theta_f)\log(((\bar B-\bar D)(C_G+(C_{P,u}-C_{P,f})\cdot Y\alpha\theta_u))/(C_TY^2\alpha\theta_u))$. Similarly, $-(a+b)\theta_f\cosh(A_2)+(-(a-b)\theta_f+2am\alpha\theta_u)\times\sinh(A_2)<0$ if and only if $\theta_f\ge m\alpha\theta_u$ or both $\theta_f < m\alpha\theta_u$ and

$$T-\frac{\log(1+(a+b)\theta_f/(a(m\alpha\theta_u-\theta_f)))}{(a+b)w} < t_0.$$

Thus, when $t_0 > (1/\theta_f)\log(((\bar B-\bar D)(C_G+(C_{P,u}-C_{P,f})\cdot Y\alpha\theta_u))/(C_TY^2\alpha\theta_u))$ and either $\theta_f\ge m\alpha\theta_u$ or both $\theta_f < m\alpha\theta_u$ and

$$T-\frac{\log(1+(a+b)\theta_f/(a(m\alpha\theta_u-\theta_f)))}{(a+b)w} < t_0$$

are satisfied, by (34), it follows that $\partial\Pi(t_0,\cdot)/\partial t_0 < 0$. This, together with constraint $t_0\le T$ and the definition of $H(Y,\gamma,p)$ presented in the statement of the proposition, completes the proofs of parts (i) and (ii). Second, evaluating (34) at $t_0 = T$, we obtain $(\partial\Pi(t_0,\cdot)/\partial t_0)|_{t_0=T} = -ampw - C_TY\theta_f - (\bar B-\bar D)C_{P,f}\theta_fe^{-T\theta_f}+O(\delta)<0$, from which it follows that there exists $T_H < T$ such that $t_0^* < T_H$. Therefore, by (34) and the definition of $g(\cdot)$ given in the proposition statement, $t_0^* = \Psi + O(\delta)$ follows immediately. □

Proof of Proposition 2. In addition to the conditions of Lemma A1, let $\theta_u = \kappa_u\delta$. Then, for sufficiently small δ, by Lemma A1 and (33) in the proof of Proposition 1, we obtain

$$\frac{\partial\Pi(t_0,\cdot)}{\partial t_0} = \frac{e^{-t_0\theta_f}}{Y}(q_0(T)+q_1(T))+O(\delta), \qquad (35)$$

where

$$\begin{aligned}q_0(T) &\triangleq -e^{t_0\theta_f}(ampwYA_1^2e^{2A_2}-C_TY^2\theta_f),\\ q_1(T) &\triangleq (\bar B-\bar D)\Big(-C_{P,f}Y\theta_f \\ &\quad + C_Gm\Big(1-A_1-\frac{a(T-t_0)\theta_f}{b}-\frac{\theta_f}{bw}\log(A_1)\Big)\Big).\end{aligned}$$

Differentiating $q_0(T)$ and $q_1(T)$, we obtain $q_0'(T) = aA_1^3\cdot e^{2A_2+t_0\theta_f}mpw^2Y(ae^{2A_2}-b)$ and $q_1'(T) = (\bar B-\bar D)aC_GmA_1\cdot((\theta_f(e^{2A_2}-1))/(a+b)+e^{2A_2}A_1w(a+b))$. Define $q(T)\triangleq q_0(T)+q_1(T)$. Because $a>b$, $q_0'(T)>0$ and $q_1'(T)>0$ are satisfied; hence, $q(\cdot)$ is increasing in T. Noting that $q(T)|_{T=t_0} = -C_{P,f}Y\theta_f(\bar B-\bar D)-Y(ampw+C_TY\theta_f)e^{t_0\theta_f}<0$ and, by Lemma A2, $\lim_{T\to\infty}q(T)>0$, $\tilde T(t_0) = \{T\colon q(T) = 0\mid t_0\}$ exists and is unique. Therefore, $\partial^2\Pi(t_0,\cdot)/\partial t_0\partial T>0$; hence, $\tilde T(\cdot)$ is increasing in t_0. Defining $\bar T = \tilde T(0)$, which is characterized in (14), if $T<\bar T$, then $q(T)<0$ for all $t_0\in[0,T]$. Therefore, by (35), $\partial\Pi(t_0,\cdot)/\partial t_0<0$ for all $t_0\in[0,T]$ for sufficiently small δ, and hence $t_0^* = L$, which proves part (i). For part (ii), suppose $T>\bar T$. Then, when $t_0 = 0$, $q(T)>0$; hence, $t_0^*>L$ for sufficiently small δ. By (14),

$$C_{P,f}Y\theta_f - C_Gm\Big(1-\frac{a\bar T\theta_f}{b}+\frac{\theta_f(\log(b+ae^{(a+b)\bar Tw})-\log(a+b))}{bw} - \frac{a+b}{b+ae^{(a+b)\bar Tw}}\Big)<0.$$

Because of this fact and $q'(T)>0$, it immediately follows that as $(\bar B-\bar D)$ decreases, $\bar T$ increases. Because $\bar B-\bar D = (1-\rho)\sigma(\gamma)Y$ and $\sigma'(\gamma)<0$, the same result holds as γ increases. Finally, taking an implicit derivative of (14) and substituting from (14) as well, we obtain

$$\begin{aligned}\frac{d\bar T}{dY} &= \big(\theta_f(C_T+C_{P,f}(1-\rho)\sigma(\gamma))(b+aJ)^3(m(Y)-Ym'(Y))\big) \\ &\quad\times\big(am(Y)^2((a+b)^3pw^2J(aJ-b)+C_G\sigma(\gamma)(1-\rho)(b+aJ) \\ &\quad\times(\theta_f(b+aJ)(J-1)+wJ(a+b)^2))\big)^{-1}, \qquad (36)\end{aligned}$$

where $J = e^{(a+b)Tw}$. Because $J>1$ and $a>b$, by (36), we obtain $d\bar T/dY>0$, which completes the proof. □

Proof of Proposition 3. Using T_H, whose existence is proven at the end of the proof of Proposition 1, let $\bar\lambda\triangleq\min(T_H, H(Y,\gamma,p))$. By Lemma A1 and following the proof of Proposition 1, for sufficiently small δ, we can differentiate (33) to obtain

$$\begin{aligned}\frac{\partial^2\Pi(t_0,\cdot)}{\partial t_0\partial\alpha} &= \frac{(b+ae^{2A_2})^{-m\alpha\theta_u/(bw)}(\bar B-\bar D)e^{-t_0\theta_f}}{bwY\alpha^2\theta_u(ae^{(a+b)Tw}+be^{(a+b)t_0w})} \\ &\quad\times Q_\alpha(t_0)+O(\delta), \qquad (37)\end{aligned}$$

where Q_α is characterized in (15), with λ taking the place of t_0 for clarity in the proposition statement. Depending on the sign of Q_α, we can use the increasing differences property to infer the monotonicity of t_0^* with respect to α. Note that Q_α is linear in both C_G and $C_{P,u}$ and hence can be written as

$$Q_\alpha = K_G^\alpha C_G + K_{P,u}^\alpha C_{P,u}, \qquad (38)$$

where

$$\begin{aligned}K_G^\alpha &\triangleq \theta_f\frac{e^{(a+b)wt_0}(a+b)^{1+m\alpha\theta_u/(bw)}}{A_1}\big(-bwA_1^{-m\alpha\theta_u/(bw)} \\ &\quad + e^{am(T-t_0)\alpha\theta_u/b}\big(bw-m\alpha\theta_u(aw(T-t_0)+\log(A_1))\big)\big) \\ &\quad + 2am^2\alpha^2\theta_u^2\sinh(A_2)e^{A_2+(a+b)wt_0+(am(T-t_0)\alpha\theta_u)/b} \\ &\quad\times(a+b)^{m\alpha\theta_u/(bw)}(aw(T-t_0)+\log(A_1)), \qquad (39)\\ K_{P,u}^\alpha &\triangleq (aw(T-t_0)+\log(A_1))\times\big(-(a+b)\theta_f\cosh(A_2) \\ &\quad + (\theta_f(b-a)+2am\alpha\theta_u)\sinh(A_2)\big)+2baw\sinh(A_2).\end{aligned}$$

By the definition of A_1 and Lemma A2, it follows that $aw(T-t_0)+\log(A_1)<0$. Therefore, if $-(a+b)\theta_f\cosh(A_2)+(\theta_f(b-a)+2am\alpha\theta_u)\sinh(A_2)<0$, then $K_{P,u}^\alpha>0$. This

inequality can be rewritten as

$$\frac{m\alpha\theta_u}{\theta_f} < \frac{(a+b)\cosh(A_2)-(b-a)\sinh(A_2)}{2a\sinh(A_2)},$$

whose right-hand side is increasing in t_0, from which it follows that a sufficient upper bound on $(m\alpha\theta_u)/\theta_f$ to ensure $K^\alpha_{P,u}>0$ is as stated in part (i) of this proposition. In this case, by (38) and since $K^\alpha_{P,u}>0$, $Q_\alpha>0$ is satisfied whenever $K^\alpha_G\geq 0$. If $K^\alpha_G<0$, then $Q_\alpha>0$ provided that $C_G/C_{P,u} < \inf\{-K^\alpha_{P,u}/K^\alpha_G\colon t_0\in[0,H(Y,\gamma,p)]\}$. This completes the proof of part (i).

For part (ii), by (39), K^α_G is linear in θ_f and can be expressed as $K^\alpha_G = K^\alpha_{G,0}+K^\alpha_{G,1}\theta_f$ where $K^\alpha_{G,0}$ and $K^\alpha_{G,1}$ are given in (39). By Lemma A2, $K^\alpha_{G,0}<0$. Examining $K^\alpha_{G,1}$, we have $K^\alpha_{G,1}<0$ if and only if $0 > e^{(am(T-t_0)\alpha\theta_u)/b}r(t_0)$, where $r(t)\triangleq bw - m\alpha\theta_u(aw(T-t)+\log(A_1)) - bw(A_1e^{aw(T-t)})^{-(m\alpha\theta_u)/(bw)}$. Moreover,

$$r'(t_0) = \frac{abwm\alpha\theta_uA_1(-1+e^{2A_2})}{a+b}(-1+(A_1e^{aw(T-t_0)})^{-m\alpha\theta_u/(bw)}).$$

From Lemma A2, it immediately follows that $(A_1e^{aw(T-t_0)})^{-m\alpha\theta_u/(bw)}>1$; hence, $r'(t_0)>0$. Since $r(T)=0$, we have $r(t_0)<0$ for all $t_0\in[0,T)$; hence, $K^\alpha_{G,1}<0$. Because $K^\alpha_{G,1}<0$ and $K^\alpha_{G,0}<0$, it follows that $K^\alpha_G<0$, and hence, by (38), $Q_\alpha<0$ provided $C_G/C_{P,u} > \sup\{-K^\alpha_{P,u}/K^\alpha_G\colon t_0\in[0,H(Y,\gamma,p)]\}$. This completes the proof. □

Proof of Proposition 4. Using T_H, whose existence is proven at the end of the proof of Proposition 1, let $\bar\lambda\triangleq\min(T_H,H(Y,\gamma,p))$. By Lemma A1 and following the proof of Proposition 1, we can differentiate (33) to obtain

$$\frac{\partial^2\Pi(t_0,\cdot)}{\partial t_0\partial Y} = Q_Y(t_0)+O(\delta) \tag{40}$$

for sufficiently small δ, where

$$Q_Y(t_0)\triangleq\sigma(\gamma)(1-\rho)e^{-t_0\theta_f}(K^Y_GC_G+K^Y_{P,u}C_{P,u}-\theta_fC_{P,f})-\theta_fC_T,$$

and

$$K^Y_G \triangleq \frac{\theta'_u(Y)}{\theta_u(Y)^2}\times\big(K^Y_{G,0}+K^Y_{G,1}\times\big((ae^{2A_2}+b)\theta_f - am\alpha\theta_u(Y)(e^{2A_2}-1)\big)\theta_u(Y)\big),$$

$$K^Y_{P,u}\triangleq K^Y_{P,u,0}+K^Y_{P,u,1}\times Y\theta'_u(Y)\left(-\theta_f+\frac{am\alpha A_1(e^{2A_2}-1)}{a+b}\times\theta_u(Y)\right)+K^Y_{P,u,2}\times\big(\theta_u(Y)+Y\theta'_u(Y)\big),$$

with $K^Y_{G,0}\triangleq\theta_fZ_2/\alpha$, $K^Y_{G,1}\triangleq -mZ_1A_1Z_3/((a+b)bw)$, $K^Y_{P,u,0}\triangleq-\theta_fZ_2$, $K^Y_{P,u,1}\triangleq Z_3m\alpha Z_1/(bw)$, $K^Y_{P,u,2}\triangleq am\alpha\cdot A_1Z_3(e^{2A_2}-1)/(a+b)$, $Z_1\triangleq a(T-t_0)w+\log(A_1)$, $Z_2\triangleq-1+Z_3$, and $Z_3\triangleq A_1^{m\alpha\theta_u(Y)/(bw)}e^{am(T-t_0)\alpha\theta_u(Y)/b}$. It can be shown that $K^Y_{G,0}<0$, $K^Y_{G,1}>0$, $K^Y_{P,u,0}>0$, $K^Y_{P,u,1}<0$, $K^Y_{P,u,2}>0$, $Z_1<0$, $Z_2<0$, and $Z_3\in(0,1)$.

In proving the results, we use Topkis's monotonicity theorem. When δ is sufficiently low, if Q_Y is positive everywhere, then t_0^* is increasing in Y. Otherwise, if Q_Y is negative everywhere, then t_0^* is decreasing in Y.

(i(a)) For any level of maturity γ, if θ_f and $C_{P,u}$ are small enough, the monotonicity depends on the sign of K^Y_G, which can be rewritten as $K^Y_G=\theta'_u(Y)((Z_2/\alpha+K^Y_{G,1}\cdot(ae^{2A_2}+b)\theta_u(Y))\theta_f/\theta_u(Y)^2 - am\alpha K^Y_{G,1}(e^{2A_2}-1))$. Define $Z_4\triangleq A_1^{m\alpha/(bw)}e^{(am(T-t_0)\alpha)/b}\in(0,1)$. Note that $Z_3=Z_4^{\theta_u(Y)}$. Let $\varrho(\theta)\triangleq-1/\alpha+Z_4^\theta(1/\alpha-\theta Z_1m/(bw))$. Then, $\varrho(\theta_u(Y))=Z_2/\alpha+K^Y_{G,1}(ae^{2A_2}+b)\theta_u(Y)$. Note that $\varrho'(\theta)=-Z_4^\theta(mZ_1)^2\alpha\theta/(bw)^2<0$ for any $\theta>0$. Moreover, $\varrho(0)=0$. Consequently, for any $\theta_u(Y)>0$, $Z_2/\alpha+K^Y_{G,1}(ae^{2A_2}+b)\cdot\theta_u(Y)<0$. Thus, K^Y_G (and, hence, Q_Y) has the sign opposite to that of $\theta'_u(Y)$. In particular, when $\theta'_u(Y)<0$, then t_0^* is increasing in Y.

(i(b)) If θ_f is not too small, and the maturity of the firm is high, then $\sigma(\gamma)$ is small, and $-\theta_fC_T$ is the dominating term in Q_Y. Thus, $Q_Y<0$, and t_0^* is decreasing in Y.

(ii(a)) When $C_{P,u}$ is high and C_G and maturity are low, then the monotonicity depends on the sign of $K^Y_{P,u}$, which can be rewritten as

$$K^Y_{P,u}=\left(1-Z_3\left(1+\frac{Z_1m\alpha\theta'_u(Y)Y}{bw}\right)\right)\theta_f+\frac{am\alpha A_1(e^{2A_2}-1)}{a+b}\times Z_3Y\theta'_u(Y)\theta_u(Y)\left(\frac{mZ_1}{bw}+\frac{1}{\theta_u(Y)}+\frac{1}{Y\theta'_u(Y)}\right).$$

Suppose that $\theta'_u(Y)<0$ and $|Y\theta'_u(Y)|$ is high enough. Given that $\theta_u(Y)$ is upper bounded, this condition can be simply expressed as $Y\theta'_u(Y)+\theta_u(Y)<\bar\mu<0$ with $\bar\mu$ high enough. Given that $Z_3\in(0,1)$ depends directly only on $\theta_u(Y)$ and not on Y and $\theta'_u(Y)$, it follows that the coefficient of θ_f in $K^Y_{P,u}$, i.e., $1-Z_3(1+Z_1m\alpha\theta'_u(Y)Y/(bw))$, can become negative for high enough $|\theta'_u(Y)Y|$ given that θ_u is upper bounded. In such regions, if θ_f is large enough, then $Q_Y<0$, and thus, t_0^* is decreasing in Y.

(ii(b)) If $\theta_u(Y)$ is very high, then since $Z_4\in(0,1)$, $Z_3=Z_4^{\theta_u(Y)}$ and $Z_3\theta_u(Y)$ become very small. In this case, $K^Y_{P,u}$ will be close to $\theta_f>0$. In such a scenario, t_0^* is increasing in Y. □

Proof of Proposition 5. Adapting (10) and (11) to include the contribution to the detection rate from beta testing, we obtain $D(t)=\bar B+(\bar D-\bar B)\times e^{-(\theta_f+\alpha\theta_uN_B)t}$ and

$$B(t)=\frac{\xi(\bar B-\bar D)e^{-(\theta_f+\alpha\theta_uN_B)t}+(\bar D\xi-\bar B(\theta_f+\alpha\theta_uN_B))e^{-\xi t}}{\xi-(\theta_f+\alpha\theta_uN_B)}$$

over $t\in[0,t_0]$. Accounting for user participation during beta testing prior to release, the processing cost and fixing cost become

$$PC=\left(\frac{\theta_f}{\theta_f+\alpha\theta_uN_B}\times C_{P,f}+\frac{\alpha\theta_uN_B}{\theta_f+\alpha\theta_uN_B}\times C_{P,u}\right)(\bar B-\bar D)\times(1-e^{-(\theta_f+\alpha\theta_uN_B)t_0})+C_{P,u}\alpha\theta_u\int_{t_0}^T N(t)(\bar B-D(t))\,dt,$$

$$FC=\frac{(C_F+(\theta_f+\alpha\theta_uN_B)C_D)}{(\theta_f+\alpha\theta_uN_B)-\xi}\left(\frac{\bar B\theta_f-\bar D\xi}{\xi}(1-e^{-t_0\xi})-(\bar B-\bar D)(1-e^{-t_0(\theta_f+\alpha\theta_uN_B)})\right)+\int_{t_0}^T(C_F+\alpha\theta_uN(t)C_D)\times(D(t)-\bar B+B(t))\,dt.$$

Substituting the adjusted expressions for $B(t)$, $D(t)$, PC, and FC into (9) yields

$$\begin{aligned}\Pi(t_0,p) &= pN(T\,|\,\cdot) - C_T\theta_f Y t_0 - \left(\frac{\theta_f C_{P,f}}{\theta_f+\alpha\theta_u N_B} + \frac{\alpha\theta_u N_B C_{P,u}}{\theta_f+\alpha\theta_u N_B}\right)\\ &\quad\times(\bar B-\bar D)(1-e^{-(\theta_f+\alpha\theta_u N_B)t_0}) - \Bigg((C_F+(\theta_f+\alpha\theta_u N_B)C_D)\\ &\quad\times\left(\frac{\bar B\theta_f-\bar D\xi}{\xi}(1-e^{-t_0\xi}) - (\bar B-\bar D)(1-e^{-t_0(\theta_f+\alpha\theta_u N_B)})\right)\Bigg)\\ &\quad\times\left((\theta_f+\alpha\theta_u N_B)-\xi\right)^{-1} - C_{P,u}\alpha\theta_u\int_{t_0}^{T} N(t)(\bar B-D(t))\,dt\\ &\quad-\int_{t_0}^{T}(C_F+\alpha\theta_u N(t)C_D)(D(t)-\bar B+B(t))\,dt\\ &\quad-\frac{C_G}{Y}\int_{t_0}^{T} N(t)B(t)\,dt, \end{aligned}\tag{41}$$

where $D(t)$, $B(t)$, and $N(t)$ solve the dynamical system in (13) over $t\in[t_0,T]$, with adjusted initial conditions

$$D(t_0) = \bar B + \frac{\bar D-\bar B}{e^{(\theta_f+\alpha\theta_u N_B)t_0}},$$

$$B(t_0) = \frac{\xi(\bar B-\bar D)e^{-(\theta_f+\alpha\theta_u N_B)t_0} + (\bar D\xi - \bar B(\theta_f+\alpha\theta_u N_B))e^{-\xi t_0}}{\xi-(\theta_f+\alpha\theta_u N_B)},$$

and $N(t_0)=0$. Differentiating (41), we obtain

$$\begin{aligned}&\frac{\partial\Pi(t_0,\cdot)}{\partial t_0}\\ &= p\frac{\partial N}{\partial t_0}(t\,|\,t_0,\cdot)\Big|_{t=T} - C_T\theta_f Y - (\theta_f C_{P,f}+\alpha\theta_u N_B C_{P,u})\\ &\quad\times(\bar B-\bar D)e^{-(\theta_f+\alpha\theta_u N_B)t_0} - ((C_F+(\theta_f+\alpha\theta_u N_B)C_D)\\ &\quad\times\left(-(\bar B-\bar D)e^{t_0\xi}(\theta_f+\alpha\theta_u N_B) + e^{t_0(\theta_f+\alpha\theta_u N_B)}\right.\\ &\quad\left.\times(\bar B(\theta_f+\alpha\theta_u N_B)-\bar D\xi)\right)\\ &\quad\times\left((\theta_f+\alpha\theta_u N_B-\xi)e^{t_0(\theta_f+\alpha\theta_u N_B+\xi)}\right)^{-1}\\ &\quad+\int_{t_0}^{T}\frac{\partial N(t\,|\,t_0)}{\partial t_0}\left(-C_{P,u}\alpha\theta_u(\bar B-D(t))\right.\\ &\qquad\left.-\alpha\theta_u C_D(D(t)-\bar B+B(t)) - \frac{C_G B(t)}{Y}\right)dt\\ &\quad-\int_{t_0}^{T}\frac{\partial D(t\,|\,t_0)}{\partial t_0}(C_F+\alpha\theta_u N(t)(C_D-C_{P,u}))\,dt\\ &\quad-\int_{t_0}^{T}\frac{\partial B(t\,|\,t_0)}{\partial t_0}\left(C_F+\alpha\theta_u N(t)C_D+\frac{C_G}{Y}N(t)\right)dt. \end{aligned}\tag{42}$$

For part (i), we demonstrate the result for a parameter region where θ_f, θ_u, $C_{P,u}$, and $C_{P,f}$ are small but relatively large in comparison with c and $1/\xi$; i.e., both the negative impact of bugs on adoption and delays in fixing these bugs are negligible relative to the other parameters in question. Thus, defining $\delta=\nu^2$, setting $\theta_f=\kappa_f\nu$, $\theta_u=\kappa_u\nu$, $C_{P,u}=\kappa_{P,u}\nu$, and $C_{P,f}=\kappa_{P,f}\nu$ and taking $c=\kappa_c\delta$ and $\xi=\kappa_\xi/\delta$ as in the conditions of Lemma A1, we can substitute an analog of the result from Lemma A1—the solution of the adapted dynamical system—into (33), such that for sufficiently small ν,

$$\begin{aligned}&\frac{\partial\Pi(t_0,\cdot)}{\partial t_0}\\ &= \frac{1}{\kappa_u Y\alpha}\left(-aA_1^2e^{2A_2}\kappa_u mpwY\alpha + C_G(\bar B-\bar D)(\kappa_f+\kappa_u\alpha N_B)\right.\\ &\qquad-\frac{C_G(\bar B-\bar D)A_1e^{A_2}}{a+b}((a+b)(\kappa_f+\kappa_u\alpha N_B)\cosh(A_2)\\ &\qquad\left.+((a-b)(\kappa_f+\kappa_u\alpha N_B)-2am\alpha\kappa_u)\sinh(A_2))\right)\\ &\qquad+O(\nu). \end{aligned}\tag{43}$$

Suppose t_0^* is in the interior. Then, by (42) and the implicit function theorem, we obtain

$$\begin{aligned}\frac{dt_0^*}{dN_B} &= -\frac{C_G(\bar B-\bar D)\kappa_u\alpha^2(ae^{(a+b)Tw}+be^{(a+b)t_0w})^3\nu}{baw(a+b)^2e^{(a+b)(T+t_0)w}}\times\frac{M_1}{M_2}\\ &\quad+O(\nu^2), \end{aligned}\tag{44}$$

where

$$\begin{aligned}M_1 &\triangleq a(-T+t_0(2-A_1e^{2A_2})) - \log(A_1)/w,\\ M_2 &\triangleq \alpha\big(-ae^{(a+b)Tw}(C_G(\bar B-\bar D)+(a+b)pwY)\\ &\qquad+be^{(a+b)t_0w}((a+b)pwY-C_G(\bar B-\bar D))\big).\end{aligned}$$

Let

$$\begin{aligned}\Delta^{N_B} \triangleq \Big\{&t_0>0:\ -aA_1^2e^{2A_2}\kappa_u mpwY\alpha\\ &+\frac{C_G(\bar B-\bar D)A_1(e^{2A_2}-1)\kappa_u m\alpha}{a+b}=0\Big\}.\end{aligned}$$

By (43), when t_0^* moves into the interior, it must satisfy $t_0^*=\Delta^{N_B}+O(\nu)$, from which it follows that there exists $\bar\omega>0$ such that if $\bar B-\bar D<\bar\omega$, then t_0^* is $O(\nu)$. In this case, $M_1=(-aTw-\log(a+b)+\log(b+ae^{(a+b)Tw}))/2+O(\nu)$; hence, by Lemma A2, $M_1>0$ for sufficiently small ν. Similarly, $M_2=-(a(a+b)^2e^{(a+b)Tw}\kappa_u mw)/((b+ae^{(a+b)Tw})^3)\times(C_G(\bar B-\bar D)(b+ae^{(a+b)Tw})+(a+b)pwY\times(ae^{(a+b)Tw}-b))+O(\nu)$. Thus, if network effects are limited, i.e., $b<a$, then $M_2<0$ for sufficiently small ν, which by (44), implies $dt_0^*/dN_B>0$. On the other hand, if $t_0^*=0$ and does not satisfy the first-order condition, then it is unchanged for a small increase in N_B. Therefore, $dt_0^*/dN_B\geq 0$, which proves part (i).

For part (ii), differentiating (42) with respect to N_B, by setting $c=\kappa_c\delta$ and $\xi=\kappa_\xi/\delta$ as in the conditions of Lemma A1 and making similar substitutions, we obtain

$$\begin{aligned}\frac{\partial^2\Pi(t_0,\cdot)}{\partial t_0\partial N_B} &= \frac{e^{-t_0(\theta_f+\alpha\theta_u N_B)}}{Y}\\ &\quad\times(K_G^{N_B}C_G+K_{P,u}^{N_B}C_{P,u}+C_{P,f}t_0Y\alpha\theta_f\theta_u)+O(\delta), \end{aligned}\tag{45}$$

where $K_G^{N_B}\triangleq 1-t_0\alpha\theta_u N_B+\kappa^{N_B}$, $K_{P,u}^{N_B}\triangleq Y\alpha\theta_u\kappa^{N_B}$, and

$$\begin{aligned}\kappa^{N_B} &\triangleq -(A_1e^{aw(T-t_0)})^{m\alpha\theta_u/(bw)} - t_0\theta_f\\ &\quad+t_0A_1^{m\alpha\theta_u/(bw)}\times\frac{(e^{(a+b)w(T+t_0)/2+am(T-t_0)\alpha\theta_u/b})}{ae^{(a+b)Tw}+be^{(a+b)t_0w}}\\ &\quad\times((a+b)(\theta_f+\alpha\theta_u N_B)\cosh(A_2)\\ &\quad+((a-b)(\theta_f+\alpha\theta_u N_B)-2am\alpha\theta_u)\sinh(A_2)).\end{aligned}$$

As θ_u gets small, κ^{N_B} approaches -1 as θ_u gets small; hence, by (45), for large enough $C_{P,u}$, $\partial^2\Pi(t_0,\cdot)/\partial t_0\partial N_B < 0$, and the result follows. □

Lemma A3. *The optimal release time satisfies* $t_0^* = L(\gamma)$ *when either* (i) $C_G < \alpha\theta_u Y(C_T/((1-\rho)\bar{\sigma}) - (C_{P,u} - C_{P,f}))$ *or* (ii) $C_G > \alpha\theta_u Y(C_T/((1-\rho)\bar{\sigma}) - (C_{P,u} - C_{P,f})e^{-\theta L(\gamma)})$ *and*

$$\sigma(\gamma) < \frac{\alpha\theta_u Y C_T}{(1-\rho)(C_G + \alpha\theta_u Y(C_{P,u} - C_{P,f}))}.$$

Proof. (i) If $C_G < \alpha\theta_u(Y)Y(C_T/((1-\rho)\bar{\sigma}) - (C_{P,u} - C_{P,f}))$, since $\sigma(\gamma) \le \bar{\sigma}$ for all γ, then it follows that for all p and all γ, we have $\log(((\bar{B} - \bar{D})(C_G + (C_{P,u} - C_{P,f}) \cdot Y\alpha\theta_u))/(C_T Y^2 \alpha\theta_u)) < 0$. Thus, from Proposition 1 we have $t_0^*(p^*) = L(\gamma)$.

(ii) Note that for any fixed p, if $\sigma(\gamma) < \alpha\theta_u C_T Y/((1-\rho)(C_G + (C_{P,u} - C_{P,f})Y\alpha\theta_u))$, then it immediately follows that $\log(((\bar{B} - \bar{D})(C_G + (C_{P,u} - C_{P,f})Y\alpha\theta_u))/(C_T Y^2\alpha\theta_u)) < 0$ and thus, from Proposition 1, we see that $t_0^*(p) = L(\gamma)$. □

Proof of Proposition 6. Conditions in cases (i) and (ii) place the price optimization in the context of conditions in Lemma A3, in which case $t_0^* = L(\gamma) = 0$ (under a low δ regime). Differentiating (32) with respect to p and α, we obtain

$$\frac{\partial^2\Pi(p,\cdot)}{\partial p\partial\alpha} = R_\alpha(p,\cdot) + O(\delta),$$

where

$$R_\alpha(p,\cdot) = -w'(p) \times \frac{(1-\rho)\sigma(\gamma)Z_3 m\theta_u}{b^2(ae^{2A_2}+b)w^3(p)} \times ((ae^{2A_2}+b)Z_1 + abTw(p)(e^{2A_2}-1)) \times V_\alpha,$$

where $V_\alpha \triangleq m(C_G + C_{P,u}Y\alpha\theta_u)Z_1 + bC_{P,u}Yw(p)$. We know that $w'(p) < 0$ for all $p > 0$. Also, $((1-\rho)\sigma(\gamma)Z_3 m\theta_u)/(b^2(ae^{2A_2}+b)w^3(p)) > 0$. Moreover, it can be shown that $(ae^{2A_2}+b)Z_1 + abTw(p)(e^{2A_2}-1) > 0$ if and only if $2aA_2e^{2A_2}/(ae^{2A_2}+b) > \log((ae^{2A_2}+b)/(a+b))$. Consider the function $\Lambda(x) = axe^x/(ae^x+b) - \log((ae^x+b)/(a+b))$. Then, we have $\Lambda'(x) > 0$ for all $x > 0$, with $\Lambda(0) = 0$. Thus, $\Lambda(x) > 0$ for all $x > 0$. Hence, $\Lambda(2A_2) = 2aA_2e^{2A_2}/(ae^{2A_2}+b) > \log((ae^{2A_2}+b)/(a+b)) > 0$, and thus, $(ae^{2A_2}+b)Z_1 + abTw(p)(e^{2A_2}-1) > 0$. Thus, the sign of $R_\alpha(p,\cdot)$ is given by the sign of V_α.

There exists $\bar{p}$ (independent of costs and functionality) such that $p^* < \bar{p}$ because $\lim_{p\to\infty}\pi(p) = 0$, and we focus on regimes where $\pi^* \ge \pi_0 > 0$. Given that $w''(p) \ge 0$, it follows that $w'(p) \le w'(\bar{p}) < 0$ and $w(p) > w(\bar{p}) > 0$ for all $p \in [0,\bar{p})$. When $C_G/C_{P,u}$ is high or Y is low, then, since $Z_1 < 0$, it follows that $R_\alpha(p,\cdot) < 0$. If $C_G/C_{P,u}$ is low, the sign of $R_\alpha(p,\cdot)$ is given by the coefficient of $C_{P,u}$ in V_α, which is $X^\alpha_{P,u} = Y(\alpha\theta_u m Z_1 + bw(p))$. Define $\chi(x) \triangleq \alpha\theta_u m(aTx + \log(a+b) - \log(b + ae^{(a+b)Tx})) + bx$. Then $X^\alpha_{P,u} = Y\chi(w(p))$. We have $\partial\chi/\partial x = b - a(a+b)e^{(a+b)Tx}mT\alpha\theta_u/(b+ae^{(a+b)Tx})$. Note that $(a+b)e^{(a+b)Tx}/(b+ae^{(a+b)Tx}) \in (1,(a+b)/a)$ for all $x > 0$. Thus, $\partial\chi/\partial x \in (b - (a+b)mT\alpha\theta_u, b - amT\alpha\theta_u)$. If $b/((a+b)mT) > \alpha\theta_u$, i.e., user contribution to debugging is low, then χ is increasing, and thus $X^\alpha_{P,u} > 0$ for any p which leads to $R_\alpha(p,\cdot) > 0$. Since $p < \bar{p}$, for sufficiently small δ, $R_\alpha(p,\cdot)$ will dominate the $O(\delta)$ term everywhere in the interval $[0,\bar{p}]$. Both parts (i) and (ii) follow immediately via Topkis's monotonicity theorem. □

Proof of Proposition 7. Conditions in cases (i) and (ii) place the price optimization in the context of conditions in Lemma A3, in which case $t_0^* = L(\gamma) = 0$ (under a low δ regime). Thus, we only have a one-variable optimization, i.e., with respect to p. Differentiating (32) with respect to p and Y, we obtain

$$\frac{\partial^2\Pi(p,\cdot)}{\partial p\partial Y} = R_Y(p,\cdot) + O(\delta),$$

where

$$R_Y(p,\cdot) = -w'(p) \times \frac{m\alpha(1-\rho)\sigma(\gamma)Z_3A_1}{b^2(a+b)w^3(p)} \times \left((ae^{2A_2}+b)Z_1 + abw(p)T(e^{2A_2}-1)\right) \times V_Y,$$

where $V_Y \triangleq bC_{P,u}w(p)(\theta_u(Y) + Y\theta_u'(Y)) + mZ_1\theta_u'(Y)(C_G + C_{P,u}Y\alpha\theta_u(Y))$. Note that $w'(p) < 0$ and $(m\alpha(1-\rho)\sigma(\gamma)\cdot Z_3A_1)/(b^2(a+b)w^3(p)) > 0$. From the proof of Proposition 6, we also know that $(ae^{2A_2}+b)Z_1 + abw(p)T(e^{2A_2}-1) > 0$. Thus, the sign of $R_Y(p,\cdot)$ is the same as the sign of V_Y. By similar arguments as in Proposition 6, if δ is sufficiently small, $R_Y(p,\cdot)$ dominates the $O(\delta)$ term everywhere on $[0,\bar{p}]$.

Cases i(a) and i(b): In these cases, the monotonicity is determined by the sign of the coefficient of C_G in V_Y. Since $Z_1 < 0$, it follows that the coefficient of C_G has a sign opposite to $\theta_u'(Y)$. The results follow directly.

Case ii(a): In this case, the monotonicity is determined by the sign of the coefficient of $C_{P,u}$ in V_Y, which is equal to $bw(p)(\theta_u(Y) + Y\theta_u'(Y)) + mZ_1\alpha\theta_u'(Y)Y\theta_u(Y)$. From Lemma A2 we have $-bw(p)T < Z_1 < 0$. Then, since $\theta_u'(Y) < 0$ in this case,

$$bw(p)(\theta_u(Y) + Y\theta_u'(Y)) + mZ_1\alpha\theta_u'(Y)Y\theta_u(Y) < bw(p)\left[\theta_u(Y) + Y\theta_u'(Y)(1 - mT\alpha\theta_u(Y))\right].$$

If $\theta_u < 1/(mT\alpha)$ and $|Y\theta_u'|$ is large, then $\theta_u(Y) + Y\theta_u'(Y)(1 - mT\alpha\theta_u(Y)) < 0$. In this case, $V_Y < 0$, and p^* is decreasing in Y.

Case ii(b): If $\theta_u'(Y) < 0$ but $\theta_u(Y) + Y\theta_u'(Y) > 0$, it immediately follows that the coefficient of $C_{P,u}$ in V_Y is positive. On the other hand, if $\theta_u'(Y) > 0$, given that from Lemma A2 we have $-bw(p)T < Z_1 < 0$, then

$$bw(p)(\theta_u(Y) + Y\theta_u'(Y)) + mZ_1\alpha\theta_u'(Y)Y\theta_u(Y) > bw(p)\left(\theta_u(Y) + Y\theta_u'(Y)(1 - mT\alpha\theta_u(Y))\right).$$

If $\theta_u(Y)/Y\theta_u'(Y) > |1 - mT\alpha\bar{\theta}_u|$, then $V_Y > 0$, and, consequently, p^* is increasing in Y. □

References

Albanesius C (2011) Apple confirms iOS 5 battery bug, promises fix. *PC Magazine* (November 2), http://www.pcmag.com/article2/0,2817,2395790,00.asp.

Arar Y (2003) Microsoft reveals Office 2003 prices, release. *PCWorld* (August 19), http://www.pcworld.com/article/112077/article.html.

Arora A, Caulkins JP, Telang R (2006) Research note—Sell first, fix later: Impact of patching on software quality. *Management Sci.* 52(3):465–471.

Arora A, Telang R, Xu H (2008) Optimal policy for software vulnerability disclosure. *Management Sci.* 54(4):642–656.

August T, Tunca TI (2006) Network software security and user incentives. *Management Sci.* 52(11):1703–1720.

August T, Tunca TI (2008) Let the pirates patch? An economic analysis of software security patch restrictions. *Inform. Systems Res.* 19(1):48–70.

August T, Tunca TI (2011) Who should be responsible for software security? A comparative analysis of liability policies in network environments. *Management Sci.* 57(5):934–959.

Babcock C (2008) Open source code contains security holes. *InformationWeek* (January 8), http://www.informationweek.com/open-source-code-contains-security-holes/205600229.

Ballmer S (2002) Connecting with customers. *Microsoft Executive E-mail* (October 2), http://www.microsoft.com/mscorp/execmail/2002/10-02customers.mspx.

Bass MF (1969) A new product growth model for consumer durables. *Management Sci.* 15(5):215–227.

Bass MF, Krishnan TV, Jain DC (1994) Why the Bass model fits without decision variables. *Marketing Sci.* 13(3):203–223.

Cavalcanti YC, de Almeida ES, da Cunha CEA, Lucrédio D, de Lemos Meira SR (2010) An initial study on the bug report duplication problem. *Proc. 14th Eur. Conf. Software Maintenance Reengineering* (IEEE Computer Society, Washington, DC), 264–267.

Cavusoglu H, Cavusoglu H, Raghunathan S (2007) Efficiency of vulnerability disclosure mechanisms to disseminate vulnerability knowledge. *IEEE Trans. Software Engrg.* 33(3):171–185.

Cavusoglu H, Cavusoglu H, Zhang J (2008) Security patch management: Share the burden or share the damage? *Management Sci.* 54(4):657–670.

Cisco (2011) Global Internet traffic projected to quadruple by 2015. Press release, June 1, http://newsroom.cisco.com/press-release-content?type=webcontent&articleId=324003.

Cohen MA, Eliashberg J, Ho T-H (1996) New product development: The performance and time-to-market tradeoff. *Management Sci.* 42(2):173–186.

Cusumano MA (2004) Who is liable for bugs and security flaws in software? *Comm. ACM* 47(3):25–27.

Davis N, Mullaney J (2003) The team software process^{SM} (TSP^{SM}) in practice: A summary of recent results. Technical Report CMU/SEI-2003-TR-014, Software Engineering Institute, Carnegie Mellon University, Pittsburgh. http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA418430.

Dou Y, Niculescu MF, Wu DJ (2013) Engineering optimal network effects via social media features and seeding in markets for digital goods and services. *Inform. Systems Res.* 24(1):164–185.

Ehrlich W, Prasanna B, Stampfel J, Wu J (1993) Determining the cost of a stop-test decision. *IEEE Software* 10(2):33–42.

Giger E, Pinzger M, Gall H (2010) Predicting the fix time of bugs. *Proc. 2nd Internat. Workshop Recommendation Systems Software Engrg.* (ACM, New York), 52–56.

Glerum K, Kinshumann K, Greenberg S, Aul G, Orgovan V, Nichols G, Grant D, Loihle G, Hunt G (2009) Debugging in the (very) large: Ten years of implementation and experience. *Proc. ACM SIGOPS 22nd Sympos. Operating Systems Principles (SOSP '09)* (ACM, New York), 103–116.

Goel AL, Okumoto K (1979) Time-dependent error-detection rate model for software reliability and other performance measures. *IEEE Trans. Reliability* R-28(3):206–211.

Hachman M (2011) Blizzard pushes out Diablo III release date until 2012. *PCMag.com* (September 23), http://www.pcmag.com/article2/0,2817,2393507,00.asp.

Harter DE, Slaughter SA (2003) Quality improvement and infrastructure activity costs in software development: A longitudinal analysis. *Management Sci.* 49(6):784–800.

Harter DE, Kemerer CF, Slaughter SA (2012) Does software process improvement reduce the severity of defects? A longitudinal field study. *IEEE Trans. Software Engrg.* 38(4):810–827.

Harter DE, Krishnan MS, Slaughter SA (2000) Effects of process maturity on quality, cycle time, and effort in software product development. *Management Sci.* 46(4):451–466.

Henry P (2007) Protecting the enterprise in a Vista environment. *SC Magazine* (*Australia*) (March 7), http://www.scmagazine.com.au/Feature/74901,protecting-the-enterprise-in-a-vista-environment.aspx.

Hooimeijer P, Weimer W (2007) Modeling bug report quality. *Proc. 22nd IEEE/ACM Internat. Conf. Automated Software Engrg.* (ACM, New York), 34–43.

Internet Systems Consortium (2012) Internet host count history. Accessed September 2012, https://www.isc.org/solutions/survey/history.

Internet World Stats (2012) World Internet usage and population statistics. Accessed September 2012, http://www.internetworldstats.com/stats.htm.

Ji Y, Mookerjee VS, Sethi SP (2005) Optimal software development: A control theoretic approach. *Inform. Systems Res.* 16(3):292–306.

Ji Y, Kumar S, Mookerjee VS, Sethi SP, Yeh D (2011) Optimal enhancement and lifetime of software systems: A control theoretic analysis. *Production Oper. Management* 20(6):889–904.

Jiang Z, Sarkar S, Jacob VS (2012) Postrelease testing and software release policy for enterprise-level systems. *Inform. Systems Res.* 23(3, Part 1):635–657.

Kalish S (1983) Monopolist pricing with dynamic demand and production cost. *Marketing Sci.* 2(2):135–159.

Kalish S, Lilien GL (1986) A market entry timing model for new technologies. *Management Sci.* 32(2):194–205.

Keizer G (2007) Vista's biggest problem remains Windows XP, survey says. *Computerworld* (November 14), http://www.computerworld.com/s/article/9046942/Vista_s_biggest_problem_remains_Windows_XP_survey_says.

Keizer G (2011) Mozilla gets tough on Firefox memory leaks. *Computerworld* (June 13), http://www.computerworld.com/s/article/9217580/Mozilla_gets_tough_on_Firefox_memory_leaks.

Kennealy E (2000) The byte stops here: Duty and liability for negligent Internet security. *Comput. Security J.* 16(4):1–26.

Kimura M, Toyota T, Yamada S (1999) Economic analysis of software release problems with warranty cost and reliability requirement. *Reliability Engrg. System Safety* 66(1):49–55.

Koch HS, Kubat P (1983) Optimal release time of computer software. *IEEE Trans. Software Engrg.* 9(3):323–327.

Krishnan MS, Kriebel CH, Kekre S, Mukhopadhyay T (2000) An empirical analysis of productivity and quality in software products. *Management Sci.* 46(6):745–759.

Krishnan TV, Bass FM, Jain DC (1999) Optimal pricing strategy for new products. *Management Sci.* 45(12):1650–1663.

Manes S (2007) Dim Vista. *Forbes* (February 26), http://www.forbes.com/forbes/2007/0226/050.html.

Markoff J (2006) A challenge for exterminators. *New York Times* (October 9), http://www.nytimes.com/2006/10/09/technology/09vista.html.

McDaid K, Wilson SP (2001) Deciding how long to test software. *Statistician* 50(2):117–134.

Meredith L (2010) Pricing details for Microsoft Office 2010 revealed. *TechNewsDaily* (May 17), http://technewsdaily.com/481-pricing-details-for-microsoft-office-2010-revealed.html.

Michaels P (2008) Faulty software can lead to astronomic costs. *Computer Weekly* (June), http://www.computerweekly.com/Articles/2009/01/13/230950/Faulty-software-can-lead-to-astronomic-costs.htm.

Microsoft (2001) Microsoft rolls out first individual trial program for Microsoft Office XP. Press release (April 2), Microsoft, Redmond, WA.

Microsoft (2007) Microsoft Office system pricing, fact sheet, January 2007. Microsoft, Redmond, WA.

M Squared Technologies (2010) RSM metrics of popular software programs. Accessed May 2010, http://msquaredtechnologies.com/m2rsm/.

Muthitacharoen A, Saeed KA (2009) Examining user involvement in continuous software development (a case of error reporting system). *Comm. ACM* 52(9):113–117.

Nagai M (2006) Sony delays PS3 launch to November. *CNET News* (March 15), http://news.cnet.com/Sony-delays-PS3-launch-to-November/2100-1043_3-6049479.html.

Niculescu MF, Shin H, Whang S (2012) Underlying consumer heterogeneity in markets for subscription-based IT services with network effects. *Inform. Systems Res.* 23(4):1322–1341.

Okumoto K, Goel AL (1980) Optimal release time for software system based on reliability and cost criteria. *J. Systems Software* 1(4):315–318.

O'Neill S (2008) Why I'm skipping Windows Vista: IT speaks out. *PCWorld* (November 4), http://pcworld.com/article/153295/skipping_windows_vista.html.

Organisation for Economic Co-operation and Development (2012) OECD broadband portal—Historical penetration rates, G7. http://www.oecd.org/sti/broadband/oecdbroadbandportal.htm.

Oswald E (2005) Businesses slow to adopt XP SP2. *BetaNews* (April 4), http://betanews.com/2005/04/04/businesses-slow-to-adopt-xp-sp2.

Otto PN (2009) Reasonableness meets requirements: Regulating security and privacy in software. *Duke Law J.* 59(2):309–342.

Paulk MC, Curtis B, Chrissis MB, Weber CV (1993) Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Petreley N (1998) The official terminology guide to commercial software development. *InfoWorld* 20(10):140.

Pham H, Zhang X (1999) Software release policies with gain in reliability justifying the costs. *Ann. Software Engrg.* 8(1–4):147–166.

Ransom D (2010) Six industries poised to grow. *Wall Street Journal* (April 27), http://online.wsj.com/article/SB10001424052748704471204575210023871636874.html.

Raymond ES (1999) *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, 1st ed. (O'Reilly & Associates, Sebastopol, CA).

Robinson B, Lakhani C (1975) Dynamic price models for new-product planning. *Management Sci.* 21(10):1113–1122.

Rooney P (2004) WinXP SP2 adoption slow in enterprise, picks up in SMB. *CRN* (November 4), http://www.crn.com/news/security/51202859/winxp-sp2-adoption-slow-in-enterprise-picks-up-in-smb.htm

RTI (2002) The economic impacts of inadequate infrastructure for software testing. Planning Report 02-3 prepared by Research Triangle Institute for the National Institute of Standards and Technology, Research Triangle Park, NC.

Sethi SP, Bass FM (2003) Optimal pricing in a hazard rate model of demand. *Optimal Control Appl. Methods* 24(4):183–196.

Shantikumar JG, Tufekci S (1983) Application of a software reliability model to decide software release time. *Microelectronics Reliability* 23(1):41–59.

Siemens Information Systems Ltd. (2003) Reduced defect density an average of 71 percent in three technical areas. On average of 46 percent of that reduction occurred as the organization moved from SW-CMM maturity level 5 towards CMMI maturity level 5. Software Engineering Institute, Carnegie Mellon University, Pittsburgh. Accessed September 2011, http://seir.sei.cmu.edu/cmmiresearch/results/state_31.html.

Software Engineering Institute (2006) CMMI for development, version 1.2. Technical Report CMU/SEI-2006-TR-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh.

Sutter JD (2010) Why Internet connections are fastest in South Korea. *CNN.com* (March 31), http://www.cnn.com/2010/TECH/03/31/broadband.south.korea/index.html.

Thompson L (2010) Socorro: Mozilla's crash reporting system. *Mozilla Webdev Development* (blog), May 19, http://blog.mozilla.com/webdev/2010/05/19/socorro-mozilla-crash-reports/.

Tryhorn C (2009) Digital Britain: "Broadband in every house by 2012." *Guardian* (*UK*) (January 29), http://www.guardian.co.uk/technology/2009/jan/29/digital-britain-broadband-houses-2012.

Weszka J (2003) Transition from SW-CMM to CMMI: The benefits continue! Presentation at the CMMI Technology Conference, November 19, Denver.

Wheeler DA (2001) More than a gigabuck: Estimating GNU/Linux's size. Accessed August 2010, http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html.

Whittaker JA (2001) Software's invisible users. *IEEE Software* 18(3):84–88.

Yamada S, Osaki S (1987) Optimal software release policies with simultaneous cost and reliability requirements. *Eur. J. Oper. Res.* 31(1):46–51.

Zhang J, Seidmann A (2010) Perpetual versus subscription licensing under quality uncertainty and network externality effects. *J. Management Inform. Systems* 27(1):39–68.

Zheng S (2002) Dynamic release policies for software systems with a reliability constraint. *IIE Trans.* 34(3):253–262.

Zimmermann T, Premraj R, Bettenburg N, Just S, Schröter A, Weiss C (2010) What makes a good bug report? *IEEE Trans. Software Engrg.* 36(5):618–643.