Vertically Differentiated Markets with Costly Under-provisioning: The Case of Versioning in Software Industry

Abstract

Versioning literature recommends that a software firm should always sell only one version of a product with the highest quality unless other factors such as piracy, network externality, or concave cost of producing quality are present. However, software firms universally adopt versioning strategies that are invariant across different market settings. To bridge the gap between theory and practice, this research proposes a vertical differentiated consumer utility model that captures consumer heterogeneity in taste for functionality and preferred level of functionality. While consumers experience disutility if the level of functionality is lower than their preferred level, they do not derive any additional utility if the level of functionality is higher than their preferred level. To that extent, under-provisioning of functionality is costly. This article shows that in a monopolistic market, versioning strategy is always optimal compared with strategies to sell one version to all consumers or only to high type users. Counterintuitively, an increase in the high (low) type users’ preferred level of functionality negatively (positively) impacts high type users’ surplus. As the preferred level of functionality of high type users increases, functionality level of the high version increases, and the impact on functionality level of the low version depends on the proportion of high type users in the market. On the other hand, as the preferred level of functionality of low type users increases, functionality level of high version remains the same and the impact on functionality level of low version is ambiguous.

Keywords: Vertical differentiation, utility function, versioning, costly under-provisioning, preferred level of functionality
1. Introduction

Versioning\(^1\) is ubiquitous in information goods in general and in software in particular. Many software firms first develop a flagship product with high level functionality and then create different versions by strategically disabling some of the functionality of the flagship product (Ghose & Sundarajan 2005; Gershoff et al. 2011). These versions are created such that higher versions have additional functionality on top of that of the lower versions. For example, Microsoft offers their operating system Windows 7 in six versions. The flagship version Ultimate has all the functionality of the lower version Enterprise and has some additional functionality. Enterprise version has all the functionality of the lower version Home and some additional functionality. Similarly, Intuit, which specializes in the tax-related software TurboTax, offers five versions of its federal and state tax filing software wherein each lower version has some functionality stripped off from the higher version. The flagship version Business has all the forms and functionality required by mid-size businesses and the lowest version Free Edition has just the simplest tax return (1040EZ).

Versioning is a product-price strategy in which a firm offers different versions of software at different prices such that different “types” of users self-select the version-price pair that is “targeted” to them (Mussa & Rosen 1978; Varian 1998; Shapiro & Varian 1998). Versioning leads to second-degree price discrimination (Bhargava & Choudhary 2001) and information goods in general and software in particular are easily amenable to it because of two unique characteristics. First, the upfront fixed development cost of the first copy of software is substantial while the production cost of the subsequent copies (the marginal cost of an additional copy) is virtually zero (Shapiro & Varian 1998). Second, software firms increasingly are adopting modularity design, which makes it easier for them to disable some functionality of the flagship version to create lower quality versions targeted to different market segments at almost negligible cost (Ghose & Sundrarajan 2005; Wei & Nault 2011). For example, all

\(^{1}\) Software versioning also refers to the practice of assigning unique version numbers to the unique states of the computer wherein these numbers are assigned in increasing order and correspond to new developments in the software (http://en.wikipedia.org/wiki/Software_versioning). This research refers to versioning as in Shapiro and Varian (1998), wherein firms “create different versions of the same core of information, based on different buyers’ needs.”
versions of Windows 7 have the same installation media and only the license determines the level of functionality to be installed onto users’ computers (Darryl 2010).

Product and pricing strategies for information goods such as software are of central interest to the academic community, and the extant literature can be broadly classified into two streams. The first stream of literature concerns the effect of cost structure of producing quality on a firm’s product and pricing strategy in a market where users have vertical differentiation in taste for quality (Bhargava & Choudhary 2001, 2008; Jones & Mendelson 2011). This literature maintains that versioning is optimal only when either marginal cost of producing quality is concave or under certain type of consumer heterogeneity in valuation. While Jones and Mendelson (2011) posited that under convex development cost “a monopoly will offer only a single product,” Bhargava and Choudhary (2001) found that versioning is profitable “when highest quality product has the best benefit-to-cost ratio,” and Bhargava and Choudhary (2008) recommended that “versioning is optimal when the optimal market share of the lower-quality version, offered alone, is greater than the optimal market share of the high-quality version, offered alone.”

The second stream of literature has studied optimality of versioning strategy in the presence of market characteristics, such as presence of piracy (Wu et al. 2003; Chellappa & Shivendu 2003, 2005; Chen & Wu 2008), presence of network externality (Jing 2000; Cheng & Tang 2010), existence of outside option (Chen & Seshadri 2007), multidimensional users’ heterogeneity (Lacourbe, et al. 2009; Wei & Nault 2011), and presence of multiple information goods providers (Wei & Nault 2006). Table 1 provides a summary of academic recommendations for versioning of information goods. Versioning also has been studied in economics literature and the recommendation has been that versioning is never optimal if the marginal cost of producing different versions is the same (Anderson & Dana 2009; Salant 1989), which is the case for information goods with zero marginal cost.
Software has negligible marginal cost of additional copies; firms remove functionality from the flagship product to create lower functionality versions at almost zero cost. Therefore, cost structure of producing functionality (equivalent of quality) in software does not appear to be a determinant of software firms’ versioning strategies. Further, market conditions do not appear to influence software firms’ versioning strategies per-se. Versions of software that are present in countries of high average income levels and low piracy rates are also available in countries of low average income levels and high piracy rates. For instance, Microsoft has a worldwide product strategy for Windows operating systems and Windows 7 Professional has the same functionality in the US and Chinese markets. Versioning strategies

<table>
<thead>
<tr>
<th>Paper</th>
<th>Model Specification</th>
<th>Market Characteristics</th>
<th>Academic finding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bhargava &amp; Choudhary (2001)</td>
<td>Vertical</td>
<td>Monopoly</td>
<td>Versioning is not optimal for information goods</td>
</tr>
<tr>
<td>Wu et al. (2003)</td>
<td>Vertical</td>
<td>Monopoly; Piracy</td>
<td>Versioning is profitable instrument to fight piracy</td>
</tr>
<tr>
<td>Jing (2000)</td>
<td>Vertical</td>
<td>Monopoly; Network externality</td>
<td>Versioning is optimal in the presence of network externality</td>
</tr>
<tr>
<td>Chellappa &amp; Shivendu (2005)</td>
<td>Vertical</td>
<td>Monopoly; Piracy</td>
<td>Versioning is profitable under narrowly defined condition for information goods</td>
</tr>
<tr>
<td>Ghose &amp; Sundararajan (2005)</td>
<td>Vertical</td>
<td>Monopoly</td>
<td>Increase in total number of versions leads to increase in quality difference between the highest and lowest versions</td>
</tr>
<tr>
<td>Chen &amp; Seshadri (2007)</td>
<td>Vertical</td>
<td>Monopoly; Outside option</td>
<td>Versioning is not optimal when high type users have higher utility of outside option than low type users.</td>
</tr>
<tr>
<td>Wei &amp; Nault (2006)</td>
<td>Vertical</td>
<td>Duopoly</td>
<td>Versioning can deter entry</td>
</tr>
<tr>
<td>Bhargava &amp; Choudhary (2008)</td>
<td>Vertical</td>
<td>Monopoly</td>
<td>Versioning is optimal if relative valuations and cost for low and high quality versions decrease when taste for quality decreases.</td>
</tr>
<tr>
<td>Lacourbe, et al. (2009)</td>
<td>Horizontal and vertical</td>
<td>Monopoly</td>
<td>Versioning is optimal when horizontal effect is stronger</td>
</tr>
<tr>
<td>Cheng &amp; Tang (2010)</td>
<td>Vertical</td>
<td>Monopoly; Network externality</td>
<td>Two versions are preferred to offering free trial when network intensity is not strong.</td>
</tr>
<tr>
<td>Wei &amp; Nault (2011)</td>
<td>Horizontal and vertical</td>
<td>Monopoly</td>
<td>Versioning is optimal if shared characteristics are not too valuable relative to mutually exclusive characteristics or if consumer valuation of the highest group is not too much higher than the next lowest group.</td>
</tr>
<tr>
<td>Jones &amp; Mendelson (2011)</td>
<td>Vertical</td>
<td>Comparing information goods and industrial goods</td>
<td>Versioning is not optimal for a monopolist as long as cost of producing quality is convex</td>
</tr>
</tbody>
</table>
of software firms also do not appear to be driven by the strength of network externality as firms have similar versioning strategies across software products with high to low network externality strength. For example, users of a particular electronic tax filing software do not necessarily benefit from an increase in the installed base. Nevertheless, TurboTax is available in five versions with increasingly more complex capabilities. Therefore, a significant gap is evident between the business practices adopted in the software industry across different markets and the previous academic findings and recommendations regarding the optimality of versioning strategy.

This paper seeks to bridge the gap between the academic recommendations and ubiquitous business practices of versioning by taking a fresh look at the formulation of consumer utility function for software, rather than focusing on cost structure and market characteristics, which has been the approach in the extant literature. Software consists of functionality based on a set of features that enable users to perform certain tasks. For example, Microsoft product development teams begin the process of new software development by first creating a “vision statement” that provides a list of “user activities that need to be supported by the product features” (Cusumano & Selby 1997). Software users derive utility by using different functionality to perform different tasks (Wilde & Scully 1995). Software quality consists of multidimensional attributes like reliability, correctness, and usability (Kitchenham & Pfleeger 1996). Thus, when a firm adopts a versioning strategy for software, very often all the versions have the same level of reliability, correctness, and usability, and the only differentiating attribute among the different versions is functionality. To that extent, users perceive software to have higher quality when it has a higher level of functionality. For instance, Windows 7 Professional is perceived to have higher quality than Windows 7 Home because the Professional version includes some additional functionality to that available in the Home version. Therefore, in this conceptualization, software quality is equivalent to the set of functionality as in Nault and Wei (2011), and the software firm has only one product characteristic to determine—the level of functionality.

In this model, users are heterogeneous not only in their taste for functionality but are also in their preferred level of functionality, as in Raghunathan (2000). Further, users are indifferent to enhanced
functionality beyond their preferred level, as in Ghose and Sundararajan (2005) wherein users derived maximum utility when the quality offered by the monopoly firm matched their marginal valuation for quality, and utility remained constant for any quality level greater than marginal valuation for quality. The conceptualization of consumer utility from software in this research has some similarity to the vertical and horizontal differentiation model of Lacourbe et al. (2009), in which the firm makes two product characteristic decisions: the level of quality and design taste of the product. Since users derive utility from software by using functionality to accomplish certain tasks, they experience disutility if the level of functionality is lower than their preferred level because they are unable to perform the desired tasks. For example, when corporate users use Windows 7 Professional instead of Window 7 Enterprise, they are unable to perform certain tasks because the Professional version does not have their preferred level of functionality, and as a result of this under-provisioning of functionality, the users experience disutility. On the other hand, users who neither need nor use functionality beyond their preferred level are indifferent to a higher level of functionality than their preferred level provides. For instance, a user who works in a corporate environment needs printer connectivity in multiple networks. She may feel inconvenienced when using an operating system that does not save her default printer setting for all networks she works with, as she would have to reset the printer setting each time she connects to a different network. Similarly, she may not use UNIX applications’ functionality and has no utility from functionality supporting UNIX applications.

The key finding in this research is that a versioning strategy is always optimal for a software monopolist as long as users are heterogeneous in their sensitivity to under-provisioning of functionality and are indifferent to over-provisioning of functionality after a certain preferred level. This finding is new and explains the universal practice of versioning adopted by software firms across different markets and software products. We posit that the driver of versioning strategy across software firms is the way users derive utility from different levels of functionality. Any particular cost structure of producing functionality and market characteristics may impact a firm’s versioning strategy, but may not be its determinant as has been the recommendations of extant literature. As software of different types becomes
an integral part of users’ business and personal activities, preferred level of functionality across different user types is likely to increase in the future. This paper shows that the optimal functionality of the high and the low versions are impacted differently as the preferred level of functionality increases. Surprisingly, when the proportion of high type users is large, the optimal functionality of the low version decreases as the low type users prefer more functionality. This is so because the firm economizes on information rent to be paid to the high type users by further distorting the functionality level of the low version.

Our contributions are threefold: First, we propose a novel conceptualization of consumer utility that captures consumer heterogeneities in marginal valuation for functionality and preferred level of functionality. By adopting this unique way of formulating utility, we were able to analyze general cases in which the two characteristics of users (marginal valuation for functionality and preferred level of functionality) are independent, whereas previous studies either have considered only one characteristic or have considered the two characteristics as identical. Second, we explain the realistic dynamics of the market for information goods and show why a versioning strategy is always optimal in the context of software when these two consumer characteristics are positively correlated. We recommend that firms take both dimensions of consumer heterogeneity into consideration in designing the optimal product functionality and pricing strategy. Third, we capture the effect of consumer disutility caused by under-provisioning of functionality to users when a software firm adopts a versioning strategy. This conceptualization sheds light on conditions under which a software firm provides low type users a product with less functionality than they prefer, even if the software firm has zero cost of providing them their preferred level of functionality.

We use this framework to study the impact of changes in consumers’ preferred level of functionality on the optimal functionality level of the high and the low versions in a versioning strategy. Counter to intuition, this analysis shows that when the preferred level of functionality of high or low type users increases, distortion of the low version (the difference between the preferred level of functionality of low type users and the functionality of the low version) depends on the distribution of users’
characteristics, no matter whether the software firm reduces, increases, or keeps the same degree of functionality. Further, the high type users get lower surplus when they have a higher preferred level of functionality or higher surplus when the low type users have a higher preferred level of functionality. In addition, the optimal functionality of the high version increases as the high type users’ preferred level of functionality increases, but is invariant of the low type users’ preferred level of functionality. The low type users’ surplus is invariant of either type of user’s preferred level of functionality.

The remainder of this paper is organized as follows: In §2, we outline our conceptualization of consumer utility function and a monopolist software company’s product strategies. In §3, we discuss a software firm’s three product strategies and show that a versioning strategy is optimal compared with the other two strategies of selling one version. In §4, we discuss the impact of changes in users’ preferred level of functionality on the software firm’s optimal versioning strategy. In §5 we analyze some extensions to our model when some of the assumptions are relaxed. We conclude our analysis and provide managerial implications in §6.

2. Model

In this study, we consider a class of software whose functionality consumers use to accomplish certain tasks and whose versions differ only in the level of functionality. Users derive utility from using a set of functionality of the software (Kekre et al. 1995) and recognize functionality as a measure of quality (Raghunathan 2000; Wei & Nault 2011). We assume that users have full information about the software’s level of functionality. A high version has higher level of functionality as compared with a low version, and without loss of generality, we assume that the high version has all the functionality that the software firm has developed and the low version has a subset of functionality of the high version.

Users are heterogeneous on two dimensions. The first dimension is taste parameter for functionality, \( \theta \), wherein \( \alpha \) proportion of users have \( \theta_H \) taste parameter and \( 1 - \alpha \) proportion of users have \( \theta_L \) taste where \( \theta_H > \theta_L \). We refer to users with taste parameter \( \theta_H \) (\( \theta_L \)) as the high type (the low type). The market consists of both types of users, that is, \( \alpha \in (0,1) \). The second dimension of consumer
heterogeneity is preferred level of functionality, $x$. A user perceives the software as providing a set of features, or functionality, which meets her specific needs (Wilde & Scully 1995); the preferred level of functionality in our conceptualization is akin to the most preferred quality level for heterogonous software users in Raghunathan (2000). In this conceptualization, users evaluate software in a task-oriented context akin to Garvin’s (1984) perspective of “user view” of quality and corresponding to the ISO definition of quality, “the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs” (ISO 1994). High type users ($\theta_H$) have higher needs to be satisfied by the software and prefer more functionality. Users are heterogeneous in preferred level of functionality, wherein high type users (low type users) have a preferred functionality level $x_H$ ($x_L$), where $x_H > x_L$. Taste parameters for functionality, $\theta$, and preferred level of functionality, $x$, are the users’ private information and the software firm knows only the distribution.

A user derives utility from using the software in four distinct parts. First, utility is derived by using the basic functionality that is present in all versions, and all users get utility $k$ from basic functionality (Shivendu & Zhang 2012). For example, simple text typing/editing is the basic functionality of Microsoft Word and is included in all versions. Second, utility is derived from using the functionality of software and is given by $\theta f(q)$, where $\theta$ is user taste for functionality, $q$ is the software’s level of functionality, and $f(q)$ is a function that maps software functionality to user utility. We assume that $f(q)$ is continuous and differentiable for all $q \in (0, x_i)$, $f(0) = 0$, and $\theta f(q)\big|_{q=x_i} = \theta f(x_i)$ for all $q > x_i$, where $i \in (H,L)$. This implies that a user does not derive any additional utility if software has more functionality than her preferred level of functionality. We assume that the users’ utility increases at a nonincreasing rate as functionality level of the software increases: $\frac{\partial f(q)}{\partial q} > 0$ and $\frac{\partial^2 f(q)}{\partial q^2} \leq 0$, for $\forall q \leq x_i$, where $i \in (H,L)$. This implies that functionality has positive but diminishing returns. Some studies have employed linear utility function, (Mussa & Rosen 1978; Moorthy & Png 1992, Chellappa & Shivendu
that is \( f(q) = aq \), which is a special case of our second part of utility function without upper bound.

The third part of utility depends on how much lower the functionality offered by the firm is compared with the user’s preferred level. In our conceptualization, the user suffers a disutility if he employs software with less functionality than his preferred level. A consumer’s disutility for under-provisioning of functionality is captured by a function \( g(x, q) \), for \( \forall q \leq x \), where \( i \in (H, L) \). This disutility is akin to a one-sided transportation cost in a horizontal differentiation model that depends on the distance between the functionality of software and the user’s preferred level. We assume that the disutility function \( g(x, q) \) is continuous and differentiable within \((0, x)\), \( i \in (H, L) \). The user experiences a decrease in disutility from under-provisioning of functionality at a nondecreasing rate as the functionality level \( q \) of the product increases:

\[
\frac{\partial g(x, q)}{\partial q} < 0 \text{ and } \frac{\partial^2 g(x, q)}{\partial q^2} \geq 0, \quad \forall q \leq x, \quad i \in (H, L).
\]

On the other hand, the user experiences an increase in disutility at an nondecreasing rate as the consumer’s preferred level of functionality \( x \) increases:

\[
\frac{\partial g(x, q)}{\partial x} > 0 \text{ and } \frac{\partial^2 g(x, q)}{\partial x^2} \geq 0, \quad \forall q \leq x, \quad i \in (H, L).
\]

Further, since disutility from under-provisioning of functionality increases as the difference between the functionality level and the users’ preferred level of functionality increases, the cross derivative of the users’ disutility is negative,

\[
\frac{\partial^2 g(x, q)}{\partial x \partial q} < 0, \quad \forall q \leq x, \quad i \in (H, L).
\]

A consumer experiences zero disutility when the functionality level of the software is the same as his/her preferred level, that is, \( g(x, q) \big|_{q=x} = 0, \quad i \in (H, L) \).

The fourth part of utility is characterized when \( q > x \), where \( i \in (H, L) \), wherein the user’s utility remains constant. Note that this characterization implies that the user does not get any utility from having more functionality than he needs and over-provisioning does not create any disutility for the user. Therefore, a user derives maximum utility when the level of functionality exactly matches his/her preferred level.
preferred level $q = x_i, i \in \{H, L\}$, and utility remains constant even if the software firm provides more functionality than the preferred level. Now, consumer utility can be written as:

$$U_i(\theta_i, x_i, q) = \begin{cases} 
\kappa + \theta_i f(q) - g(x_i, q), & \text{if } q < x_i \\
\kappa + \theta_i f(x_i), & \text{if } q \geq x_i 
\end{cases}, i \in \{H, L\}$$  \hspace{1cm} (1)

Note the following three key features of this utility function (Figure 1): (1) utility is monotonically increasing at a decreasing rate in functionality level for high as well as low type users up to their preferred levels of functionality respectively; (2) users’ utility remains constant even if the quality of software increases beyond their preferred levels; and (3) there is an “indifferent quality level” $q'$ at which both types of users derive the same utility.\(^2\)

![Figure 1: Consumer utility when $x_H > x_L$.](image)

Following the literature (Wei & Nault 2006; Chellappa & Shivendu 2005), the development cost of the first copy of the software, $c(q)$, is convex in the functionality level of the highest version of the software. The convex cost of producing functionality is justified because incorporating additional functionality in software requires more design and testing effort at an increasing rate (Chen & Seshadri 2007). In other words, to develop a higher level of functionality is increasingly costly, $\frac{\partial c(q)}{\partial q} > 0$ and $\frac{\partial^2 c(q)}{\partial q^2} > 0$. Further, we assume $c(q)$ is continuous and differentiable for $\forall q$. The firm develops the high version of the software with the largest set of functionality and removes some functionality to make the

\(^2\) Please see online supplement for the proof of existence of “indifferent functionality level” $q'$.  


low version without incurring any additional cost. Further, there is no variable cost of providing additional copies of the high or low version and all upfront R&D investment is sunk.

2.1 Firm’s product-price strategies

The firm has three options regarding product-price strategy. Option A is to sell one version to high type users only. The firm’s profit function is:

$$\pi_H(p_H, q_H) = D_H(p_H, q_H)p_H - c(q_H).$$

(2)

where $D_H(p_H, q_H)$ is demand from high type users. The firm’s optimization problem is:

$$\max \pi_H(p_H, q_H) = \alpha p_H - c(q_H) \quad \text{s.t.}$$

(3)

$$\text{IR:} \quad U_H(\theta_H, x_H, q_H) \geq p_H$$

Option B is to sell one version to all users. The firm’s profit function is:

$$\pi_A(p_A, q_A) = (D_H(p_A, q_A) + D_L(p_A, q_A))p_A - c(q_A).$$

(4)

The firm’s optimization problem is:

$$\max \pi_A(p_A, q_A) = p_A - c(q_A) \quad \text{s.t.}$$

(5)

$$\text{IR:} \quad U_L(\theta_L, x_L, q_A) \geq p_A$$

Option C is to adopt a versioning strategy in which two functionality-price pairs are offered and users self-select. The firm’s profit function is:

$$\pi_V(p_{H}, p_{L}, q_{H}, q_{L}) = D_H(p_{H}, q_{H})p_H + D_L(p_{L}, q_{L})p_L - c(q_{H}).$$

(6)

And the firm’s optimization problem is:

$$\max \pi_V(p_{H}, p_{L}, q_{H}, q_{L}) = \alpha p_H + (1-\alpha)p_L - c(q_{H}) \quad \text{s.t.}$$

(7)

$$\text{IR (L):} \quad U_L(\theta_L, x_L, q_L) \geq p_L$$

$$\text{IR (H):} \quad U_H(\theta_H, x_H, q_H) \geq p_H$$

$$\text{IC (L):} \quad U_L(\theta_L, x_L, q_L) - p_L \geq U_L(\theta_L, x_L, q_H) - p_H$$

$$\text{IC (H):} \quad U_H(\theta_H, x_H, q_H) - p_H \geq U_H(\theta_H, x_H, q_L) - p_L$$

A summary of the notations is provided in Appendix A. We make the following two additional assumptions:
A1. The development cost for functionality is relatively small; therefore, it is always optimal for a firm to produce at least the low type users’ preferred level of functionality, that is, 
\[ \min(\theta_H f'(x_L), \alpha(\theta_H f'(x_L) - g'(x_H, q = x_L))) \geq c'(x_L). \] 
In other words, this assumption implies that irrespective of the firm’s adopted product strategy, the rate of increase of benefit to users from functionality is no less than the rate of increase of cost for functionality at the low type users’ preferred level of functionality. This implies that the optimal highest functionality \( q' \) offered by the firm is never less than \( x_L \), that is, \( q' \geq x_L \).

A2. The high type users’ preferred level of functionality is not too large compared with the low type users’ preferred level of functionality. The difference in utility between high and low type users is greater than the disutility for under-provisioning of functionality of the high type users when the functionality is set at the low type users’ preferred level, that is, \( f(x_L)(\theta_H - \theta_L) > g(x_H, q = x_L) \). This assumption implies that the indifferent functionality level is \( q' < x_L \).

3. Versioning Strategy

The monopolist software firm has three product-price options, as given in §2.1. Before we examine the firm’s profits under different options and optimal product-price strategy, we first consider some characteristics of the user utility function and optimal functionality level under the three options. Since the high type users have a higher preferred level of functionality \( x_H > x_L \), they are more sensitive to under-provisioning of functionality and their utility from software decreases at a faster rate compared with low type users when functionality level decreases.

**LEMMA 1:** When users experience disutility from under-provisioning of functionality, there exists an indifferent functionality level \( q' \) such that (i) \( U_H(\theta_H, x_H, q') = U_L(\theta_L, x_L, q') \), and (ii) \( q' > 0 \).

*For all proofs, see Appendix B (online supplement).*

When the software has just the basic functionality, \( q = 0 \), the high and the low type users derive the same utility \( k \), that is, \( \theta_H f(0) = \theta_L f(0) = 0 \), but the high type users incur greater disutility than the low
type users because they have higher preferred level of functionality than the low type users, 
\( g(x_H, 0) > g(x_L, 0) \). Thus, the high type users have lower total utility than the low type users at the basic level of functionality. When the level of functionality is at the high type users’ preferred level of functionality, \( q = x_H \), the high type users have higher total utility than the low type users because they derive higher utility than the low type users, \( \theta_H f(x_H) > \theta_L f(x_H) \), and both types incur zero disutility, 
\( g(x_H, q = x_H) = g(x_L, q = x_H) = 0 \). Thus, there exists a functionality level, that is, \( q' > 0 \), at which both user types have the same utility from using the software as utility is monotonically nondecreasing in level of functionality for both types. We refer to this level of functionality as the indifferent level of functionality \( q' \) (Figure 1). Since the high type users’ preferred level of functionality is higher than the low type users’, at any functionality level lower than \( q' \), the high type users derive higher utility but incur more disutility than the low type users, such that total utility of high types is lower than that of low types. Referring to the example of multi-printer connectivity, both Windows Professional and Ultimate have the functionality of saving the user’s default printer setting for different networks while Windows Home does not. The corporate user may have more valuation for functionality provided by Windows Home than a user who uses it only at home and connects to one printer. However, the corporate user may have a lower total utility for Windows Home than the home user if she incurs too much inconvenience for resetting the printer setting many times a day.

Before deciding the functionality-price menu under the three product strategies, the firm determines the highest level of functionality to be produced under each strategy. In our model, the utility derived by high type users is constant \( k + \theta_H f(x_H) \) if the software’s functionality level is higher than the high type user’s preferred level, that is, \( q > x_H \). This implies that an increase in the level of functionality beyond the preferred level \( x_H \) does not lead to any increase in the high type user’s valuation. Nonetheless, the firm incurs a development cost for the increase of the functionality. Therefore, the firm will never set the level of functionality of the high version more than the preferred level of the high type user’s because that would lead to some increase in cost without any increase in revenue.
LEMMA 2: When users experience disutility from under-provisioning of functionality, the software firm’s optimal functionality offerings under different product strategies are as follows: (A) When the firm adopts the product strategy of selling only to high type users, the optimal level of functionality \( q_{ih}^* \) solves the first-order condition given by \( \alpha \partial q H f'(q_{ih}^*) - \alpha g'(x_H, q_{ih}^*) - c'(q_{ih}^*) = 0 \). Optimal level of functionality is always bound such that \( x_L < q_{ih}^* \leq x_H \). The optimal functionality level increases within that bound as the proportion of high type users \( \alpha \) increases. (B) When the firm adopts the product strategy of selling to all users, then the firm produces the low type users’ preferred functionality level as the optimal level of functionality, \( q_{ih}^* = x_L \). (C) When the firm adopts a versioning strategy, then the highest level of functionality produced is the same as in (A), \( q_{ih}^* = q_{ih}^{*+} \).

When the software firm adopts the product strategy of selling only to high type users, the optimal functionality is set at the level at which the marginal cost of functionality equals the marginal benefit of functionality for the high type users, weighted by the proportion of high type users. As the proportion of high type users increases, the firm has an incentive to increase the level of functionality of software because the increase in revenue outweighs the increase in development cost of more functionality. When the firm adopts the product strategy of selling to all users, the optimal functionality level is set at \( q_{ih}^* = x_L \). The economic intuition is that the marginal benefit of functionality is greater than the marginal cost at any level of functionality that is lower than the low type users’ preferred level \( x_L \) (Assumption 1). Since consumer valuation remains constant when \( q > x_L \), the firm has no gain in producing functionality greater than the low type users’ preferred level of functionality.

When the software firm adopts a versioning strategy, the optimal functionality level of the high version is the same as the optimal functionality level given in Lemma 2 (A). The economic intuition behind this result is as follows: The optimal level of functionality is determined by the tradeoff between the marginal benefit of functionality and the marginal cost of functionality and at the optimal functionality level, marginal cost equals marginal benefit. This tradeoff is the same under both options, selling only to high type users or adopting a versioning strategy.
PROPOSITION 1: When users experience disutility from under-provisioning of functionality, under a versioning strategy, the software firm’s optimal level of functionality of the low version is within the range of \([q', x_l]\), that is, \(q' \leq q_{vl}^* \leq x_l\).

Under a versioning strategy, the optimal functionality level of the low version lies within the range of \([q', x_l]\), that is, \(q' \leq q_{vl}^* \leq x_l\). The intuition is easy to see. By our assumption A2 \((q' < x_l)\), the information rent paid to the high type users is zero at the indifferent functionality level \(q'\) because at that level both types of users have the same utility. By producing a level of functionality lower than \(q'\), the firm incurs loss in revenue from the low type users and does not gain from the high type users by saving information rent, which is zero at \(q'\). Therefore, we have \(q' \leq q_{vl}^*\). Similarly, for any functionality level, \(q > x_l\), the firm must provide more information rent to the high type users, but gains nothing from the low type users. Therefore, we have \(q_{vl}^* \leq x_l\).

LEMMA 3: When users experience disutility from under-provisioning of functionality, then under a versioning strategy, the optimal functionality level of the low version depends on the proportion of high type users in the market. (A) When \(\alpha \in (0, \alpha_2]\) where \(\alpha_2 = \frac{\theta_l f'(x_l)}{\theta_h f'(x_l) - g'(x_h, x_l)}\), the optimal functionality level of the low version is \(q_{vl}^{\alpha^*} = x_l\). (B) When \(\alpha \in [\alpha_1, 1)\) where \(\alpha_1 = \frac{\theta_l f'(q') - g'(x_h, q')}{\theta_h f'(q') - g'(x_h, q')}\), the optimal functionality level of the low version is \(q_{vl}^{\alpha^*} = q'\). (C) When \(\alpha \in (\alpha_2, \alpha_1)\), the optimal functionality level of the low version is bound within \([q', x_l]\), such that \(q_{vl}^{\alpha^*}\) solves the first-order condition given by \((\theta_l f'(q_{vl}^{\alpha^*}) - g'(x_l, q_{vl}^{\alpha^*})) - \alpha(\theta_h f'(q_{vl}^{\alpha^*}) - g'(x_h, q_{vl}^{\alpha^*})) = 0\).
Figure 2: Optimal functionality level of the low version as the proportion of high type users changes

Under a versioning strategy, the optimal functionality level of the low version, as characterized in Lemma 3, depends on two critical proportions of high type users in the market, namely the upper threshold proportion $\alpha_1$ and the lower threshold proportion $\alpha_2$, as shown in Figure 2. When under-provisioning of functionality leads to disutility, the optimal functionality level of the low version is at the preferred level of low type users $x_L$ as long as there are relatively fewer high type users in the market, that is, $\alpha \in (0, \alpha_2)$. The economic intuition is that in this range, the marginal benefit of functionality for the low type users’ valuation is greater than the marginal effect on information rent to be paid to the high type users to maintain incentive compatibility. Therefore, it is optimal for the firm to offer the low version software at the highest possible functionality level, $x_L$. When the proportion of high type users in the market is moderate, $\alpha \in (\alpha_2, \alpha_1)$, then firms makes a tradeoff between the marginal revenue from the low type users and the marginal information rent to the high type users and reduces functionality of the low version from the upper bound $x_L$ as the proportion of high type users increases (Figure 2). The economic intuition is easy to see: As the proportion of high type users increases, functionality of the low version is distorted downwards to save on information rent. When the proportion of high type users is relatively large in the market, $\alpha \in [\alpha_1, 1)$, the firm is better off setting functionality level for the low version at the indifferent functionality level $q'$ where information rent to high type users is zero.

Now we focus on optimal profits under the three options. Note that Option A (selling the high version only to high types) covers only the high type consumers, while Option B (selling to all) and Option C (versioning strategy) cover the entire market. With a versioning strategy, the entire market is
covered, but the firm’s ability to charge a high price for the high version is limited because of the potential of cannibalization between versions (Raghunathan 2000; Belleflamme 2002). In the presence of the low version, the high type users pay less for the high version as they get information rent, but the low type users, who previously did not find the high version attractive, buy the low version. The firm provides information rent to the high type users by reducing the price of the high version so that the high type users receive the same net surplus from buying either of the versions, and therefore, do not switch from buying the high version to the low version. The information rent increases as the difference in functionality of the two versions decreases. As the level of functionality offered to low type users increases (up to $x_L$), revenue from the low type users increases, but revenue from the high type users decreases because the firm pays more information rent to the high type users. The firm determines an optimal level of functionality of the low version (Lemma 3) and the price of the high version by balancing these two opposite effects.

**PROPOSITION 2:** When users experience disutility from under-provisioning of functionality, versioning strategy dominates the product strategies of selling only to the high type users or selling to all users.

The economic intuition of Proposition 2 is as follows: Compared with the product strategy of selling to all users, a versioning strategy with no functionality distortion of the low version has the same revenue from low type users, but higher revenue from high type users, after adjusting for higher development cost of high level of functionality ($q_{VH L} > x_L$). Therefore, a versioning strategy dominates when selling to all users in the range of $\alpha \in (0, \alpha_x]$. While the optimal revenue for a strategy of selling to all users is constant, the optimal revenue for a versioning strategy increases as the proportion of high type users increases in the market. Thus, when the proportion of high users is relatively higher, $\alpha \in [\alpha_x, 1)$, then also a versioning strategy will be the dominant strategy. Therefore, a versioning strategy always dominates a product strategy of selling to all users.
Figure 3: Profits under three product strategies with disutility from under-provisioning of functionality

Since the firm provides some information rent to the high type users to incentivize them to refrain from switching to the low version, the high type users are charged a lower price under a versioning strategy than under a product strategy of selling only to high type users, although the optimal functionality level of the high version is the same under both product strategies, that is, \( q_H^* = q_H^{pe} \). Consequently, the optimal revenue from the high type users under a versioning strategy is lower than under a strategy of selling only to high type users. Nevertheless, the gain in revenue from low type users more than compensates for this loss of revenue from high type users, when the proportion of high type users is relatively low, \( \alpha \in (0, \alpha_1) \). On the other hand, when the proportion of high type users is relatively high, \( \alpha \in [\alpha_i, 1) \), the firm sets the optimal functionality level of the low version at the indifferent level of functionality, \( q' \), and pays no information rent to the high type users. This implies that when \( \alpha \in [\alpha_i, 1) \), the price of the high version is same as under the strategy of selling only to high type users. Therefore, the firm generates the same revenue from high type users and some additional revenue from low type users under a versioning strategy compared with a strategy of selling only to high type users. Therefore, revenue is strictly higher under a versioning strategy compared with a product strategy of selling only to high type users, a result that holds even when the common utility of basic functionality between the two versions is zero, \( k = 0 \). Figure 3 graphically shows profits under the three product-price strategies and the dominance of a versioning strategy.
4. Impact of changes in users’ preferred level of functionality

Disutility from under-provisioning of functionality has significant implications in determining the optimal functionality levels for both types of users. In this section we study the impact of changes in preferred functionality levels of high type users ($x_H$) and low type users ($x_L$) on the level of functionality of high version ($q_{HV}^*$), the level of functionality of low version ($q_{VL}^*$), the upper threshold proportion ($\alpha_1$) and the lower threshold proportion ($\alpha_2$).

**LEMMA 4:** As the preferred level of functionality of high type users $x_H$ increases, (i) the indifferent functionality level $q'$ increases, and (ii) the lower threshold proportion $\alpha_2$ and the upper threshold proportion $\alpha_1$ decrease.

As the high type users’ preferred functionality level increases, these users become more sensitive to under-provisioning as their disutility from under-provisioning increases. This implies that high type users are less attracted to the low version, which has two effects. First, the indifferent functionality level, $q'$, at which both types have the same total utility from software, increases. Second, the firm increases the level of functionality of the low version, which leads to a smaller functionality distortion of the low version to increase revenue from low type users.

However, as high type users’ preferred level of functionality increases, they become more sensitive to changes in the functionality level of the low version. As a result, the firm starts reducing the low version’s level of functionality from the preferred functionality level of low type users, $x_L$, with at a smaller threshold proportion of high type users ($\alpha_2$) and at a faster rate as the high type users preferred level of functionality, $x_H$, increases. Combining these two factors—an increase in the indifferent functionality level ($q'$) and an increase in sensitivity of information rent to high type users to functionality of the low version—the firm stops reducing functionality of the low version at a smaller threshold proportion of high type users ($\alpha_1$).
**Proposition 3:** As the preferred level of functionality of high type users $x_{H}$ increases, (i) the optimal functionality distortion of the low version $\left( x_{L} - q^p_{L} \right)$ (a) does not change if the proportion of high type users is small, (b) increases if proportion of high type users is moderate, and (c) decreases if the proportion of high type users is large, (ii) the optimal level of functionality of the high version $q^p_{H}$ increases, and (iii) consumer surplus of high type users decreases and low type users remain the same.

Combining the changes on $\alpha_2$ and $\alpha_4$, the overall effect is that the condition on the proportion of high type users in the market for the firm’s providing some functionality distortion is less restrictive, but the changes in magnitude of functionality distortion is mixed. When there are few high type users (small $\alpha$), the low type users continue to receive the same level of functionality $x_L$ (no functionality distortion) even when the high type users’ preferred level of functionality increases. This occurs simply because there are not enough high type users to make any impact on the overall strategy. When there are a relatively moderate number of high type users, the low type users who originally received zero functionality distortion or some functionality distortion receive more distortion as the high type users’ preferred level of functionality $x_H$ increases. The economic intuition is as follows: Since high type users are now more sensitive to changes in functionality level of the low version, the firm gains more revenue by reducing information rent to the high type users by distorting functionality of the low version than losing revenue from the low type users by lowering the functionality level of the low version. When the proportion of high type users is relatively high (large $\alpha$), low type users get the low version at a indifferent functionality level $q'$. As $x_H$ increases, the indifferent quality level increases, and therefore, when $\alpha$ is high, the low type users get less functionality distortion. This is because the firm is better off producing the low version at a higher functionality level without paying any rent to high type users.
In Figure 4, we illustrate the impact of changes in preferred functionality level of the high type users \((x_H)\) on the lower and the upper threshold proportions \((\alpha_z \text{ and } \alpha_i)\), the optimal functionality level of the low version \((q_{LV}^*)\), and information rent. The marginal effect on information rent to high type users increases, when the preferred functionality of high type users increases from \(x_{HA}\) to \(x_{HC}\). The band of the optimal level of functionality of the low version narrows, as the upper limit \((x_L)\) is unchanged, but the lower limit, the indifferent functionality level, increases from \(q'_A\) to \(q'_C\). Both the upper and the lower threshold proportions of high type users decrease from \([\alpha_{zA}, \alpha_{iA}]\) to \([\alpha_{zC}, \alpha_{iC}]\). When high type users’ preferred level of functionality increases from \(x_{HA}\) to \(x_{HB}\), the low type users get less functionality distortion if \(\alpha_{iHB} < \alpha < 1\), more functionality distortion if \(\alpha_{zB} < \alpha < \alpha_{iHB}\), and no distortion if \(0 < \alpha < \alpha_{zB}\).

On the other hand, an increase in high type users’ preferred level of functionality has a positive impact on the high version’s optimal level of functionality. The economic intuition is easy to see: As \(x_H\) increases, the marginal effect of functionality on high type users’ valuation increases, though the marginal effect on development cost remains the same. Therefore, the firm has an incentive to provide the high version of a higher level of functionality. On the other hand, surprisingly, the information rent to high type users decreases when the high type users’ preferred level of functionality increases. In Figure 4, the information rent decreases as the high type users preferred level of functionality increases from \(x_{HA}\) to
As high type users’ preferred level of functionality increases, these users become more sensitive to under-provisioning of functionality and less inclined to switch to the low version. And the firm provides less information rent to high type users to incentivize them to refrain from switching to the low version. It is counter to intuition that high type users would get lower information rent because one may think that the high type users would be better off when they prefer a higher level of functionality. But when the high type users’ preferred level of functionality increases, they get a higher functionality level, the range of proportion of high type users in the market when they get information rent is narrower, and the amount of information rent decreases. Therefore, the high type users’ consumer surplus declines.

**Lemma 5:** As the preferred level of functionality of low type users $x_L$ increases, (i) the indifferent functionality level $q'$ decreases, and (ii) the lower threshold proportion $\alpha_2$ and the upper threshold proportion $\alpha_1$ increase.

Changes in the low type users’ preferred level of functionality has the opposite impact on the optimal level of functionality of the low version as compared with the changes in the preferred level of functionality of high type users. The indifferent level of functionality $q'$ decreases as the low type users incur more disutility from under-provisioning of functionality when their preferred level of functionality increases. As a result, the range of the optimal level of functionality of the low version, $[q', x_L]$, widens. Moreover, since the low type users are more sensitive to under-provisioning of functionality, the marginal effect of functionality on revenue from the low type users increases as the low type users’ preferred level of functionality increases. Therefore, the two threshold proportions of the high type users in the market increase when $x_L$ increases.

**Proposition 4:** As the preferred level of functionality of low type users $x_L$ increases, (i) the optimal functionality distortion of the low version $(x_L - q''_L)$ (a) increases if proportion of high type users is large, and (b) decreases if proportion of high type users is moderate, and (c) does not change if proportion of
high type users is small, (ii) there is no impact on the optimal level of functionality of high version \( q_{H}^{*} \), and (iii) consumer surplus of high type users increases.

The firm’s decision whether to reduce or increase functionality distortion when the low type users’ preferred level of functionality increases depends on the distribution of high type users in the market in exactly the opposite way as an increase in the high type users’ preferred level of functionality, discussed in Proposition 3. In Figure 5, we illustrate the impact of preferred functionality level of low type users on the optimal functionality level of the low version. When the low type users’ preferred functionality increases from \( x_{LA} \) to \( x_{LB} \) or from \( x_{LB} \) to \( x_{LC} \), the range of the optimal functionality level of the low version widens from \([q_{LA}^{*}, x_{LA}] \) to \([q_{LB}^{*}, x_{LB}] \) and from \([q_{LB}^{*}, x_{LB}] \) to \([q_{LC}^{*}, x_{LC}] \), and the two threshold proportions of high type users in the population shift from \([\alpha_{LA}, \alpha_{LA}] \) to \([\alpha_{LB}, \alpha_{LB}] \) and from \([\alpha_{LB}, \alpha_{LB}] \) to \([\alpha_{LC}, \alpha_{LC}] \). When the low type users’ preferred level of functionality increases from \( x_{LA} \) to \( x_{LB} \), the low type users receive no distortion when \( \alpha \in (0, \alpha_{LA}) \), less distortion when \( \alpha \in (\alpha_{LA}, \alpha_{LB}) \), or more distortion when \( \alpha \in (\alpha_{LB}, 1) \).

![Figure 5](image)

Figure 5: Optimal functionality level of the low version and information rent to a high type user as preferred functionality level for low type users increases when \( f(q) = q \), \( g(x,q) = (x-q)^2 \), \( c(q) = q^2 \)

The preferred level of functionality of low type users has no impact on the optimal level of functionality of the high version because the firm sets the functionality of the high version at a level where the marginal cost of functionality equals the marginal benefit of functionality to the high type users,
weighted by the proportion of the high type users and independent of the characteristics of the low type users. Nevertheless, the information rent to high type users is impacted. In Figure 5, within the range of \( \alpha \in (0, \alpha_{\text{ULC}}) \), the firm pays more information rent to each individual high type user and increases the functionality level of the low version as the low type users’ preferred functionality level increases. The information rent that an individual high type user gets is minimized to zero when the optimal functionality level of the low version is at \( q' \) when \( \alpha \in (\alpha_1, 1) \). And the upper threshold proportion of high type users, \( \alpha_1 \), increases as the preferred functionality level of the low type users increases. Therefore, it is counterintuitive that when the low type users prefer a higher level of functionality, the high type users are better off as they get more information rent for the high version with the same level of functionality. The intuition is as follows: The firm has an incentive to raise the functionality level of the low version since the low type users are more sensitive to under-provisioning of functionality as they prefer a higher level of functionality. The firm has to raise information rent to the high type users to keep them from buying the low version with the improved level of functionality. Nevertheless, the firm gains more in the revenue from the low type users than it loses in information rent to the high type users. Further, there is no change in the low type users’ surplus as they get no information rent. To conclude, as the low type users’ preferred level of functionality increases, the consumer surplus of low type consumers remains the same, the price of the high version decreases, and the consumer surplus of high type users increases.

To summarize, as the high type users prefer a higher level of functionality, (i) the condition on functionality distortion of the low version is less restrictive \( (1-\alpha_z \text{ increases}) \), (ii) functionality distortion of the low version depends on the distribution of high type users in the market, (iii) the optimal functionality level of the high version \( (q_{\text{HN}}^*) \) is greater, (iv) the condition for an individual high type user’s getting some information rent is more restrictive \( (\alpha_1 \text{ increases}) \), and (v) the amount of information rent is smaller. These findings imply that when the high type users prefer a higher functionality level, the firm would increase, decrease, or maintain the magnitude of functionality distortion to the low type users depending upon the proportion of high type users in the market, and the firm may start distorting the
functionality of the low version at a smaller proportion of high type users in the population. The high type users receive more functionality but get a lower or zero information rent. On the other hand, the impact of an increase in low type users’ preferred level of functionality has the opposite effects, except it has no impact on the optimal level of functionality of the high version.

5. **Some Extensions**

In §3 and §4, we discussed the product and pricing strategy of a software firm when users suffer disutility from under-provisioning of functionality. We also assumed that the cost of developing functionality and utility to users from functionality is such that the software firm always produces software with higher functionality than the low type users’ preferred level of functionality, \( q' \geq x_L \), (assumption A1), and that the high type users’ preferred level of functionality is not too large compared with the low type users’ such that the indifferent functionality level is lower than the low type users’ preferred level of functionality, \( q' < x_L \), (assumption A2). In this section, we analyze the optimality of a versioning strategy when users suffer no disutility from under-provisioning, when the cost of developing functionality is relatively high (A1 is relaxed), or the preferred level of functionality of the low type users is smaller than the indifferent functionality level (A2 is relaxed).

5.1 **No consumer disutility from under-provisioning of functionality**

We consider the case in which users have no disutility from under-provisioning of functionality, that is, \( g(x_i, q) = 0, \forall q \) for \( i \in \{H, L\} \). When the firm adopts the product strategy of selling only to high type users, the optimal functionality is lower than when users experience disutility from under-provisioning of functionality. The firm will still set the level of functionality of the high version at the level at which the marginal cost of functionality equals the marginal benefit to the high type users, weighted by the proportion of high type users. But the firm has less incentive to provide a higher level of functionality to the high type users because they are less sensitive to under-provisioning of functionality. When the firm adopts a product strategy of selling to all users, the optimal functionality level is never higher than the preferred level of low type users \( x_L \) for the same economic intuition as in §3.
Under a versioning strategy, the firm sets the optimal functionality level of the high version at the same level as when it sells only to high type users. The firm chooses the optimal level of functionality of the low version by trading off the decrease in revenue from having to pay information rent to the high type users in the presence of low version and the increase in revenue from the low type users as the full market is covered. When there are relatively more high type users in the market, \( \alpha > \frac{\theta_l}{\theta_h} \), the firm lowers information rent to the high type by setting the level of functionality of the low version at the basic level where the rent to the high type users is minimized. When there are relatively more low type users in the market, \( \alpha < \frac{\theta_l}{\theta_h} \), the firm has an incentive to increase the revenue from the low type users, and therefore, sets the level of functionality of the low version at the preferred functionality level of the low type users \( x_L \) to extract the maximum revenue from that segment. The firm receives the same profit by setting the level of functionality of the low version at the basic level or at the preferred functionality level of the low type, \( x_L \), when \( \alpha = \frac{\theta_l}{\theta_h} \).

**PROPOSITION 5:** When users experience no disutility from under-provisioning of functionality, a versioning strategy dominates the other two product strategies—selling only to the high type users or selling to all users.

When users do not experience disutility from under-provisioning of functionality, a versioning strategy always dominates the product strategy of selling to all users (Option B). The intuition is the same as in §3 in which consumers experience disutility for under-provisioning of functionality. When the proportion of high type users is relatively low \( \left( \alpha < \frac{\theta_l}{\theta_h} \right) \), a versioning strategy generates the same revenue from low type users, because there is no distortion in the functionality level of the low version, but more revenue from high type users compared with a strategy of selling to all users. And when there is a relatively high proportion of high type users \( \left( \alpha > \frac{\theta_l}{\theta_h} \right) \), the firm’s gain in revenue from the high type
users is more than the loss in revenue from the low type users (as the functionality level of the low version is distorted downwards) under a versioning strategy than under a strategy of selling to all users.

Moreover, when users experience no disutility of under-provisioning, a versioning strategy weakly dominates a product strategy of selling only to high type users (Option A). Under a strategy of selling only to high type users, the firm sets the optimal level of functionality at \( q_{H}^{*} \) and charges a price equal to valuation of high type users for \( q_{H}^{*} \), that is, \( k + \theta_{H} f(q_{H}^{*}) \). When there are relatively more high type users in the market, \( \alpha > \frac{\theta_{L}}{\theta_{H}} \), the firm sets the optimal level of functionality of the high version at \( q_{H}^{*} \) and the optimal level of functionality of the low version is set at the basic level. Thus, the firm is able to charge the same price to high type consumers and generate the same profit under both strategies. When there are more low type users in the market, \( \alpha < \frac{\theta_{L}}{\theta_{H}} \), the firm sets the optimal level of functionality of the high version at \( q_{H}^{*} \) and the optimal level of functionality of the low version at \( x_{L} \). Because the gain in revenue from the low type users is more than the loss in revenue from payment of information rent to the high type users, the firm’s profit is higher under a versioning strategy compared with a product strategy of selling only to high type users.

Figure 6: Profits under three product strategies when there is no disutility from under-provisioning
The firm’s optimal product strategy balances the effect of the low versions’ level of functionality on the information rent paid to high type users, and the revenue from the low type users by covering the entire market. Profits under the three product strategies are shown in Figure 6. The optimal profit under a versioning strategy and the optimal profit under a strategy of selling only to high type users increase but the optimal profit under a strategy of selling to all users is constant as the proportion of high type users $\alpha$ increases. As the proportion of the high type users in the market increases, the firm reduces the level of functionality of the low version. When the proportion of high type users is above the critical proportion, $\frac{\theta_L}{\theta_H}$, the low version has just the basic functionality. Given any proportion of high type users in the market, a versioning strategy dominates the other two product strategies as the profit line $\pi_v^*$ is above $\pi_{hv}$ and $\pi_A^*$ for all $\alpha$.

**COROLLARY 1:** When users do not experience disutility from under-provisioning of functionality and there is no basic functionality across versions, a versioning strategy dominates the product strategies of selling one version to high type users only or to all users when the proportion of the high type users is small $\left(\alpha < \frac{\theta_L}{\theta_H}\right)$; while a versioning strategy weakly dominates the other two product strategies when the proportion of the high type users is large $\left(\alpha \geq \frac{\theta_L}{\theta_H}\right)$.

When there is no common utility for the basic functionality across both the versions, $k=0$, a versioning strategy still weakly dominates the other two product strategies. When the proportion of high type users is large, the firm sets the functionality level of the low version at zero to minimize information rent to high type users. In that case, a versioning strategy is effectively the same as a product strategy of selling one version at $q_H^*$ to high type users only, and a versioning strategy weakly dominates the other two product strategies.

**5.2 Cost of developing functionality is relatively high**
Next, we relax the assumption A1 that \( \min(\theta_f f'(x_L), \alpha(\theta_H f'(x_L) - g'(x_L))) \geq c'(x_L) \). It implies that the optimal level of functionality under a product strategy of selling to high type users only or to all users may be less than \( x_L \). The optimal functionality of the high version under a versioning strategy may not be greater than the indifferent functionality level \( q' \). Therefore, there are four possibilities: a) \( q_{\text{HH}}^O < q' \), b) \( q_{\text{HH}}^O > q_{\text{HL}}^O > q' \), c) \( q_{\text{HH}}^O > q > q_{\text{HL}}^O \), and d) \( q_{\text{HH}}^O = q' \).

![Figure 7: Consumer utility of high and low type users when \( q_{\text{HH}}^O < q' \)](image)

When the optimal functionality is less than the indifferent functionality level, that is, \( q_{\text{HH}}^O < q' \) (possibility “a” shown in Figure 7), low type users receive higher utility from \( q_{\text{HH}}^O \) level of functionality than high type users. In this case, high type users get the low version software and low type users the high version. When the high version’s optimal functionality is greater than the indifferent functionality level, \( q_{\text{HH}}^O > q_{\text{HL}}^O > q' \) (possibility “b”), the results in §3 hold. When the optimal \( q_{\text{HH}}^O > q > q_{\text{HL}}^O \) (possibility “c”) is true, the firm will produce the high version at \( q_{\text{HH}}^O \) and the low version at \( q' \), and versioning still dominates the other two product strategies. When the optimal functionality is same as the indifferent functionality level, \( q_{\text{HH}}^O = q' \) (possibility “d”), the optimal strategy for the firm will be to produce one version of the functionality level and sell to all users, \( q_{\text{A}}^O = q' \).

### 5.3 High type users’ preferred level of functionality is relatively too large

30
We know from Lemma 4 that when high type users’ preferred level of functionality increases, the indifferent level of functionality, \( q' \), increases. When \( x_H \) is too large compared with \( x_L \), the indifferent functionality level, \( q' \), may be higher than the low type users’ preferred level of functionality, \( x_L \). In this subsection we relax assumption A2, that is \( f(x_L)(\theta_H - \theta_L) > g(x_H, q = x_L) \), and consider the case in which \( q' \geq x_L \).

![Diagram of consumer utility of high and low type users](image)

**Figure 8: Consumer utility of high and low type users when \( x_L < q' < q_{H_H}^{\beta} \)**

When the indifferent functionality level is equal to the low type users’ preferred level of functionality, \( q' = x_L \), the results in §3 hold and the optimal functionality level of the low version is \( x_L \). When the indifferent functionality level is greater than the low type users’ preferred level of functionality, \( q' > x_L \), three outcomes are possible: 1) \( q_{H_H}^{\beta} > q' \), 2) \( q_{H_H}^{\beta} < q' \), or 3) \( q_{H_H}^{\beta} = q' \). As shown in Figure 8, when the optimal functionality is greater than the indifferent functionality level (outcome 1), the firm is indifferent from producing the low version at a functionality level between \( x_L \) and \( q' \) because there is no change in the firm’s profit with any change in functionality of the low version when it is between \( x_L \) and \( q' \). Nevertheless, the firm is more inclined to set the optimal functionality of the low version at \( x_L \) because the high type users are less likely to buy the low version at a functionality level \( x_L \). For outcomes 2 and 3, the analyses are identical to possibilities “a” and “d” in §5.2, when the optimal functionality level of the high version is less than or equal to the indifferent functionality level. The optimal strategy for the firm is to produce one version at the indifferent functionality level and sell to all users.

31
6. Discussion

In this paper we examined the optimality of a versioning strategy for a profit maximizing monopolist software firm by developing a novel consumer utility model that captures consumer heterogeneity in two dimensions, namely marginal valuation for functionality and preferred level of functionality. In this stylized model, two effects are associated with the consumer preferred level of functionality: (a) users experience constant utility even if the functionality of software increases beyond their preferred level, and (b) they experience disutility if the functionality of software is less than their preferred level. We showed that a versioning strategy is always optimal compared with product strategies of selling one version to the high type users only or to all users in the market. This result is consistent with the software industry’s ubiquitous practice of versioning, but is at variance from recommendations from the literature shown in Table 1. The economic intuition for this finding is as follows: When the firm adopts a strategy of selling to all users, the profit is limited by the low type users’ preferred level of functionality as users derive no additional utility from functionality beyond their preferred level. When the firm adopts a strategy of selling a high version to the high type users only, profit is limited because only part of the market is covered. The high type users incur more disutility from under-provisioning of functionality than the low type users as the preferred level of functionality of high type users is higher. This implies that the firm needs to pay lower information rent to the high type users in the presence of the low version. A versioning strategy dominates a strategy of selling a high version only to high type users because of savings in information rent. Moreover, under some conditions, the firm pays no information rent and can even achieve first-degree price discrimination whenever the optimal functionality level of the low version equals the “indifferent functionality level.” The firm always produces the high version at the optimal level of functionality, although the low version’s functionality level is distorted downwards if the proportion of high type users is relatively high as a result of information rent and efficiency tradeoffs.

We also examined the impact of increases in the preferred functionality level of both types of users on the optimal functionality level and price under a versioning strategy. When the high type users’
preferred level of functionality increases, the high type consumers are more sensitive to under-provisioning of functionality and less inclined to buying the low version. This makes it possible for the firm to offer higher level of functionality, but pay lower information rent to the high type users. This implies that though the functionality level and profit of the firm increases, consumer surplus decreases.

The impact of increase in the high type users preferred level of functionality on the optimal functionality level of the low version is ambiguous and is determined by the distribution of users’ taste for functionality. On the other hand, when the low type users’ preferred level of functionality increases, they are more sensitive to under-provisioning of functionality. Since any increase in the level of functionality of the low version implies that the high type users get more information rent, the distribution of users’ taste for functionality in the market determines whether the low type users receive an increase, a decrease, or no change in level of functionality of the low version. Any increase in the preferred functionality level of the low type users has no impact on the optimal functionality level of the high version.

Extant literature in information systems has focused on either cost structure of developing quality or market characteristics like the presence of piracy or network externality to study the efficacy of a versioning strategy. In the absence of specific market characteristics, the recommendation has been that it is suboptimal to offer versions of information goods like software where the development cost is convex and the marginal cost of producing additional copies is negligible (Bhargava & Choudhary 2001, 2008; Jones & Mendelson 2011). In this research, we reexamined the consumer utility function and posited that users derive utility from the level of functionality in software in a task-oriented setting. This conceptualization of utility function for software users is unique and also allows us to study software firm’s product-pricing strategies in which under-provisioning is costly and over-provisioning is useless.

Our user utility model borrows different parts from literature - utility from basic functionality (Shivendu & Zhang 2012), utility from using functionality (Bhargava & Choudhary 2001; Moorthy & Png 1992), preferred level of functionality and costly under-provisioning (Ranghunathan 2000), and useless over-provisioning (Ghose & Sundararajan 2005) - and integrates to provide an overall mapping from the level of functionality to user utility. This conceptualization of consumer utility is one of the key theoretical
contributions of this research and may be used to study the impact of cost structure and other product and market characteristics on the versioning strategy.

To distinguish our conceptualization of utility from other models of vertical differentiation, we refer back to the multi-printers connectivity example discussed in §1 and §3. A corporate user whose job requires printer connectivity in multiple networks would have the same valuation for Windows Professional and Ultimate, which both have the functionality of remembering the default printer setting for different networks, but Ultimate offers more functionality than she needs—support for UNIX applications. Moreover, users are dissatisfied when software does not deliver the functionality that they need to perform tasks. The corporate user will be dissatisfied with Windows Home, which retains printer settings only for one network, because she would have to reset the default printer setting each time she connects her PC to a different network. However, a home user who only connects to the network at home will find the functionality of remembering the default printer setting for different networks useless. In this instance, the corporate user is more dissatisfied than the home user by Windows Home. This disutility from under-provisioning of functionality effectively allows the firm to separate the users who have different needs for software functionality with lower information rent, which makes a versioning strategy more profitable compared with a single version strategy. Our analytical framework explains business practices widely adopted by general-purpose software firms like Microsoft and Intuit to offer inferior products by strategically removing features from the flagship product.

This research provides key insights to managers to help them to develop optimal product-pricing strategies for information goods like software and information services from which users derive utility in a task-oriented context. Even though the firm can provide a low version with low type users’ preferred level of functionality, it is often beneficial to distort the level of functionality of low version downwards to save on information rent. This research provides guidelines for adjusting functionality-price decisions as the high or the low type users’ preferred levels of functionality increase over time. More specifically, when there is a shift in users’ preferred levels of functionality, firms may need to adjust the degree of functionality distortion of the low version to avoid either any loss of revenue from low type users or any
overpayment of information rent to high type users. Managers also need to be aware that the optimal degree of functionality distortion in the low version depends on the proportion of high type users in the market, and the firm may accordingly adjust the low version’s functionality level in different markets.

We made some assumptions regarding the users’ preferred level of functionality and cost of development to build the framework for analysis. Later in §5, we showed that a versioning strategy often dominates other product strategies even when these assumptions are relaxed. We identified the setting in which a single version to all users may dominate—when the cost of producing quality is too high and the preferred level of functionality for high type users is too large compared with low type users. We assumed the users had full information about the level of functionality in both the high and low versions, though in reality, users often may be uncertain about the true level of functionality and a potential future research project may be to allow uncertainty in the utility model. We assumed that over-provisioning is useless, it also may be costly when users have some resource constraints, as in Chellappa and Mehra (2011), or are inconvenienced by too much functionality. Future research may examine versioning strategies under costly over-provisioning, which may shed some light on theoretical understanding of the widespread creation of bloatware.

References


Belleflamme, P. “Pricing information goods in the presence of copying,” mimeo, Queen Mary University of London, 2002.


Chellappa, R. K., and Mehra, A. “Versioning 2.0: A product line and pricing model for information goods under usage constraints and with R&D costs,” Working paper, presented at Theory of Economics of

---

3 Software bloat is a process whereby successive versions of a computer program include an increasing proportion of unnecessary features that the end users do not access, or that generally use more system resources than necessary, while offering little or no benefit to users (http://en.wikipedia.org/wiki/Bloatware).


Appendix A: Summary of Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_H$</td>
<td>Marginal valuation for functionality of high type users</td>
</tr>
<tr>
<td>$\theta_L$</td>
<td>Marginal valuation for functionality of low type users</td>
</tr>
<tr>
<td>$x_H$</td>
<td>High type users preferred level of functionality</td>
</tr>
<tr>
<td>$x_L$</td>
<td>Low type users preferred level of functionality</td>
</tr>
<tr>
<td>$q_H$</td>
<td>Functionality under a strategy of selling to high type users only</td>
</tr>
<tr>
<td>$q_A$</td>
<td>Functionality under a strategy of selling to all users</td>
</tr>
<tr>
<td>$q_{VL}$</td>
<td>Functionality of low version under a versioning strategy</td>
</tr>
<tr>
<td>$q_{vH}$</td>
<td>Functionality of high version under a versioning strategy</td>
</tr>
<tr>
<td>$k$</td>
<td>Utility derived from the basic functionality of the software</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Proportion of high type users in the market</td>
</tr>
<tr>
<td>$D$</td>
<td>Supper Script: consumers experience disutility for under-provisioning of functionality</td>
</tr>
<tr>
<td>$p_A$</td>
<td>Price under a product strategy of selling to all users</td>
</tr>
<tr>
<td>$p_H$</td>
<td>Price under a product strategy of selling to the high type users only</td>
</tr>
<tr>
<td>$p_{vH}$</td>
<td>Price of the high version under a versioning strategy</td>
</tr>
<tr>
<td>$p_{vL}$</td>
<td>Price of the low version under a versioning strategy</td>
</tr>
<tr>
<td>$\pi_A$</td>
<td>Profit under a product strategy of selling to all users</td>
</tr>
<tr>
<td>$\pi_H$</td>
<td>Profit under a product strategy of selling to the high type users only</td>
</tr>
<tr>
<td>$\pi_v$</td>
<td>Profit under a versioning strategy</td>
</tr>
<tr>
<td>$q'$</td>
<td>Functionality level where both types of users have the same valuation</td>
</tr>
<tr>
<td>$\alpha_2$, $\alpha_1$</td>
<td>Critical values of proportion of the high type consumer in the market where the firm starts and stops distorting functionality of the low version respectively</td>
</tr>
</tbody>
</table>

Appendix B: Proofs of Lemmas and Propositions (online supplement)