Generational Evolution in Digital Platforms:
An Empirical Investigation of the Windows Desktop Operating System

Ramnath K. Chellappa

Goizueta Business School, Emory University

Jonathan Gomez

Marshall School of Business, University of Southern California

Anand Swaminathan

Goizueta Business School, Emory University

Last update: December 2, 2024

ABSTRACT

Despite relying on complementary development for most of their capabilities, digital platforms must undergo periodic changes to introduce new functionalities. However, the process of evolving across platform generations is made difficult by the continued presence of prior generations. This paper provides a holistic understanding of this process and highlights strategies for promoting generational platform evolution. Employing a semi-structural approach to examine the evolution of Microsoft's Windows operating system, we unravel the factors affecting the success of Window's generational evolution. We do this by developing a generalizable framework which incorporates strategic behavior from users and developers deciding to adopt and abandon different platform generations. Our results highlight the importance of first-party development as a commitment mechanism to promote the platform's generational evolution.

Keywords: platform evolution, platform strategy, platform competition, first-party development, planned obsolescence, operating systems

INTRODUCTION

Digital platforms have been a topic of interest in economics (e.g., Katz and Shapiro 1994; Caillaud and Jullien 2003; Armstrong 2006) and management (e.g., Boudreau 2010; Kretschmer and Claussen 2016; Chellappa and Mukherjee 2021) due to the fact that they derive most of their value from network effects (Parker and Van Alstyne 2005). Research on digital platforms assume that they are a "structurally stable" set of components on which external complementors can develop additional capabilities (Gawer 2014). However, the nature of complements is largely dependent on the capabilities offered by the platform (Baldwin and Clark 2006). Accordingly, a platform owner may find it necessary to periodically update the platform to a later generation which introduces new functionality and additional avenues for complementary development (Cennamo 2018). However, there is limited research examining how a platform can evolve across generations.

The strategic development of new generations of a product have been studied extensively as planned obsolescence (e.g., Swan 1970; Bulow 1986; Waldman 1996). A mainstay of this literature is that a producer must credibly commit to the longevity of each product generation in order to entice widespread adoption (Waldman 1993). This problem is particularly pronounced for digital platforms who must credibly commit to the new generation while simultaneously addressing the canonical chicken-or-egg problem for each generation (Caillaud and Jullien 2003). Accordingly, platform owners require a deliberate planned obsolescence strategy in order to successfully execute generational evolution.

Recent research has examined strategies whereby a platform owner can promote generational evolution. Examining the video games market, these studies have recommended reducing the cost of entry for existing developers (Kretschmer and Claussen 2016) and developing 1st-party complements (Cennamo 2018) to kick-start the new platform generation's network effects. While this work demonstrates the inherent trade-offs faced by a platform owner attempting to launch a new generation, they fall short of evaluating platform generation's complete life-cycle by omitting consideration of the abandonment of the prior generation. This

aspect is critical for generational platform evolution because a platform derives its value from its network. Therefore, an old generation can only be made obsolete by completely transitioning consumers and complementors away from it.

In this research, we expand on prior research which examines platform evolution by considering the processes through which a new platform generation emerges and an old generation is made obsolete. In doing so, we identify strategies adopted by platform owners to promote the generational evolution of their platform and examine the efficacy of such strategies. For this, we evaluate each of the key decisions which jointly determine the success of a platform's generational evolution through a series of utility-based empirical models. Our modeling provides a generalizable framework for studying such dynamics across various types of platforms.

Our study is motivated by the generational evolution of Microsoft's Windows operating system (OS) between the years of 2009 to 2017. During this time period, Microsoft released or ended support for 5 generations of the Windows OS. We leverage weekly market share data and a complete panel of the population of software products which ever supported any Windows generation. Combining this with Microsoft's release and end-of-life schedule for each Windows generation, we are able to determine the drivers of success for the generational evolution of Microsoft's Windows OS which is captured through four distinct outcomes - user adoption of the new generation[1], developer entry in support of the new generation, incumbent developer support of the new generation, and developer abandonment of the old generation.

Our research makes several contributions to platforms strategy research. First, we present an empirical examination of the *entire* life-cycle of a platform generation, including adoption *and* abandonment. In doing so, we are able to develop a more thorough understanding of the dynamics of generational platform evolution compared to previous empirical research studying the same phenomenon (e.g., Kretschmer and Claussen 2016; Cennamo 2018). Second, our analysis of the abandonment process provides insights into some of the drawbacks of developing a strong

[1]Following prior research (e.g., Caillaud and Jullien 2003; Armstrong 2006), we model users as single-homing. This implies that users simultaneously adopt a new generation and abandon a prior generation - making the adoption and abandonment decisions one and the same.

platform with substantial network effects. In particular, a new generation will have to compete against a dominant prior generation in order for the platform to successfully evolve. We contribute to a growing stream of research which cautions against the "winner-take-all" paradigm (Eisenmann et al. 2006) by demonstrating instances in which too much success can lead to sub-optimal outcomes for the platform (e.g., Cennamo 2018). Third, our research also elaborates theory regarding the impact of 1st-party complements by demonstrating an additional motivation for developing such complements, as a signaling mechanism to promote generational evolution. We advance economics research on planned obsolescence by demonstrating a novel commitment mechanism that can be used to signal commitment to a new generation *and* demonstrate waning commitment to an old generation - thus providing a commitment mechanism with multiple strategic levers. Below, we review previous research on planned obsolescence in digital platforms emphasizing the influence of network effects and first-party complements. We then develop a model of platform competition across generations and test our predictions using data on five generations of the Windows desktop OS platform, available between 2009 and 2017.

PLANNED OBSOLESCENCE IN DIGITAL PLATFORMS

Platform Evolution

Our research builds on earlier work that has attempted to establish a research direction for the study of platform evolution (Tiwana et al. 2010) and proposed a unifying framework for the same (Gawer 2014). Extant research suggests that platforms may evolve in two ways: (i) a platform's capabilities may be updated over time by complementary development (e.g., Tiwana 2015) or (ii) a platform may evolve through generational changes (e.g., Kretschmer and Claussen 2016). The former stream of research informs us that platform dynamics may change over the lifespan of a platform generation (Cennamo 2018) and provide a generalizable way to capture such changes (Chu and Manchanda 2016). However, our research is more closely related to the latter stream of research.

With the exception of the pioneering work of Clements and Ohashi (2005) on the video

games market, generational platform evolution has only recently been directly examined. Research on the video games market has found that a platform owner can overcome the classic chicken-or-egg problem (Caillaud and Jullien 2003) in generational platform evolution by making it easier for developers to port their existing complements to the new generation (Kretschmer and Claussen 2016) or by developing their own first-party complements (Cennamo 2018). However, previous research has not addressed the underlying issue that generational platform evolution often results in both the new and old platform generations competing against one another. Accordingly, a platform owner requires a deliberate obsolescence strategy to make prior generations - which already have an installed base - obsolete.

Obsolescence of Digital Goods

Competition between sequential releases has been studied in economics as *obsolescence* (Bulow 1986). Originally founded on the study of durable goods, early research examined the economic viability (Swan 1970) and welfare impact (Fishman et al. 1993) of reducing the longevity of durable goods. Later research expanded on the strategies available for producers to render old generations economically, if not physically, obsolete by making old generations "less useful" compared to the latest generation (Lee and Lee 1998). Economic obsolescence is a necessity for the sequential release of information goods (i.e., textbooks, software, platforms) which do not naturally decay (Iizuka 2007). Critical to the examination of obsolescence is consumers' forward looking behavior, whereby consumers may preempt a shortened usable life for a new generation by reducing their optimal investment levels, and therefore necessitating a credible commitment from the producer (Waldman 1993). While obsolescence research does not directly evaluate digital platforms, these issues are particularly relevant for generational platform evolution because (i) a platform owner can only make old generations obsolete by eliminating their existing user base and (ii) the platform owner must convince *both* consumers and developers to adopt the latest generation in lieu of the prior generation. To understand how a platform owner overcomes such challenges, we follow the aforementioned research on obsolescence in conceptualizing

generational platform evolution in terms of competition between platform generations.

Platform Competition

Digital platforms derive most of their value from network effects, i.e., connecting consumers with developers (Shapiro and Varian 1998; Parker and Van Alstyne 2005). As such, platform competition requires successfully convincing consumers to adopt, and developers to support, a focal firm's platform rather than one offered by its competitors (Eisenmann et al. 2006). This often results in a market "tipping" towards a single platform which then dominates the market (Katz and Shapiro 1994). The prevalence of network effects makes it difficult for new platforms to compete against existing platforms which command an installed base advantage (Zhu and Iansiti 2012). This means that for each subsequent platform generation, a platform owner is forced to resolve the chicken-or-egg problem in the presence of their own prior dominant platform generation. To overcome this challenge, prior research suggests developing in-house complements to kick-start the adoption process (Cennamo 2018). However, the use of first-party products risks destabilizing the delicate balance between platform owners and third-party complementors (Zhu 2019).

Seeding First-Party Development

Substantial research has examined the competitive dynamics that arise from a platform owner introducing its own first-party complements (for a comprehensive review, see Zhu 2019). While researchers appear convinced that such developments benefit consumers, as they can gain more variety, they have found mixed results when evaluating the impact on 3rd-party complementors. Some scholars find that 1st-party development can increase awareness for a product category and therefore increase demand for the category, thus increasing complementors' development incentives (e.g., Cennamo et al. 2018; Foerderer et al. 2018). Others, meanwhile, find that 1st-party products intensify competition and therefore reduce development incentives (e.g., Wen and Zhu 2019; Li and Agarwal 2017). A key determinant of whether first-party development helps or harms developers appears to be the platform owner's motivation in

developing its own complements (Gawer and Cusumano 2002). For example, third-party complementors appear to benefit from a platform owner introducing its own complements to develop an underutilized market segment (Gawer and Henderson 2007). Likewise, a platform owner undergoing generational platform evolution may find it beneficial to seed first-party products in order to increase network effects (Cennamo 2018) and signal commitment to the new platform generation (Waldman 1993).

In this study we build on research in platform evolution to understand the process whereby a platform transitions from one generation to the next. We do this by conceptualizing generational platform evolution as the platform equivalent of an obsolescence strategy, where the new generation must compete against the old generation. Accordingly, we draw upon prior theories of platform competition to examine how a platform owner can accelerate the transition between generations of its platform. We develop a holistic model whereby consumers elect which platform generation to occupy and developers elect which generation(s) to support. We extend previous research by also considering the process whereby consumers and developers abandon old releases. By considering the full life-cycle of a platform generation - from release to abandonment - we are able to observe the implications of a platform owner's first-party product strategy throughout the platform life-cycle. Additionally, by incorporating elements of both generational evolution and within-generation evolution, our research presents the most complete picture of platform evolution to date.

MODEL OF GENERATIONAL PLATFORM EVOLUTION

We begin our analysis of generational platform evolution by constructing a general model of platform competition across generations. Our model is motivated by desktop operating systems (OSs), the context of our study, as this setting captures a wide variety of nuanced decision-making processes. Specifically, this context allows us to model single-homing users and multihoming developers[2] (Armstrong 2006), new developers, that are often the focus of research on platform

[2]Note that in our context, multihoming refers to temporal multihoming - where a developer can use multiple sequential generations of the same platform. This is in contrast to classical competition research in which multihoming refers to simultaneously using multiple competing platforms.

competition (e.g., Gandal et al. 2000; Clements and Ohashi 2005; Zhu and Iansiti 2012), and incumbent developers that have already invested in a prior generation (Kretschmer and Claussen 2016). As mentioned earlier, in addition to support of the new generation by entrant and incumbent developers, we also account for user adoption of the new generation and developer abandonment of the old generation. We do so by developing models of each of the four decisions below.

User Adoption of a New Platform Generation

Recall that users are assumed to single-home, and therefore the decision to adopt a new generation and abandon an old generation occurs simultaneously. Formally, users are assumed to adopt a new platform generation in order to benefit from cross-side and same-side network effects (Chu and Manchanda 2016). In the desktop OS market, these result from a preference for compatibility (same-side network effects) and for greater software variety (cross-side network effects). Formally, a representative user i's benefit from adopting a new generation r at time t is dependent on (i) the number of other users who are already using the generation N_{rt}^{u} (Chu and Manchanda 2016) as well as (ii) the number of applications available from third-parties N_{rt}^{3rd} and (iii) the number of first-party applications from the platform owner available for the new generation $MSFT_{rt}$ - which jointly determine the breadth of software availability (Boudreau 2011). In addition to network effects, a user's decision to adopt the latest generation will also be determined by (iv) user idiosyncratic needs and preferences, i.e., group purchases, ξ_i (Berry et al. 1995), (v) the cost of adoption, i.e., licensing costs, P_r, and (vi) the standalone benefit from the new generation, i.e., visual or performance improvements, α_r (Zhu and Iansiti 2012). Accordingly, the net indirect utility of a representative user adopting a generation r at time t is given by:

$$V_{irt} = f(N_{rt-1}^{u}, MSFT_{rt}, N_{rt}^{3rd}, \alpha_r, P_r, \xi_i) \quad (1)$$

In our context, it is assumed that users already have a previous generation of the platform $j \neq r$ which they can continue to use at no additional cost and obtain the benefit V_{ijt} where

$P_j = 0$. Note that a user's decision to adopt the latest generation is a function of the attributes of the new generation r as well as those of the user's current generation j. Accordingly, a user's decision to adopt the latest platform generation can be modeled as a discrete choice where a user adopts the latest generation if and only if:

$$V_{irt} - V_{ijt} = f(N^u_{rt-1}, N^u_{jt-1}, MSFT_{rt-1}, MSFT_{jt-1}, N^{3rd}_{rt-1}, N^{3rd}_{jt-1}, \alpha_r, \alpha_j, P_r, \xi_i) \geq 0 \qquad (2)$$

Equation 2 demonstrates that a user's adoption decision is a function of SSNEs, CNEs, and platform strategies of both the new and old generations. Provided that users are known to adopt a platform to benefit from SSNEs and CNEs (Chu and Manchanda 2016), then we should expect the following hypotheses to hold:

Hypothesis 1a. Users will be more likely to adopt a new generation if a greater number of users currently use the new generation

Hypothesis 1b. Users will be less likely to abandon an old generation if a greater number of users currently use the old generation

Hypothesis 2a. Users will be more likely to adopt a new generation if a greater number of developers currently support the new generation

Hypothesis 2b. Users will be less likely to abandon an old generation if a greater number of developers currently support the old generation

In addition to the standalone SSNEs and CNEs, users may also benefit from the functionality provided by first-party applications (Zhu 2019).

Hypothesis 3a. Users will be more likely to adopt a new generation if a greater number of first-party products are available for the new generation

Hypothesis 3b. Users will be less likely to abandon an old generation if a greater number of first-party products are available for the old generation

However, platforms research has repeatedly found that the value of a platform ecosystem is primarily derived from the variety of applications provided by diverse contributors (Boudreau 2011). This would suggest that both first-party products and the intrinsic characteristics of the platform generation will be less influential in a user's adoption/abandonment decision than

classical SSNEs and CNEs.

> *Hypothesis 4. First-party products will be less influential in a user's adoption decision when compared to third-party products*

Developer Support of a New Platform Generation

Modeling generational platform evolution is more complex on the developer side because there are multiple types of developers and they may multihome across platform generations. Consider that platform complements can be produced by incumbent developers, that already have an application supporting a prior generation, or entrant developers, that are entering the platform for the first time with a new application. Additionally, for incumbent developers, the decision to support a new generation is distinct from the decision to stop supporting a prior generation. Further, when a developer decides to stop supporting a prior generation, they must also decide how to go about it - abandoning the old generation in favor of a new generation (i.e., switch) or exit the market entirely. Provided that entrants' and incumbents' support decisions are materially distinct while they share similar considerations during abandonment, we separately model each of these decisions.

Formally, we assume that all developers (entrants and incumbents) are profit-maximizing and the market has free entry and exit, subject to a fixed cost of entry (Gandal et al. 2000). Developers are also assumed to develop expectations about future adoption and support of the new and old generations from the platform owner's actions, user adoption, and other developers' support decisions. Expectations are assumed to be correct on average. We further expand on each model below.

Developer Support

We consider the profit-maximizing decision of a developer i to support a generation r with an application in category c. At every time t a developer supporting a generation r gains revenues ρ_{irct} and incurs a maintenance cost m_{irct}. Accordingly, the per-period profits to developer i of

supporting the platform generation r at time t are:

$$\pi_{irct} = \rho_{irct} - m_{irct} \tag{3}$$

When a developer decides to support the new generation, the developer will incur a fixed support cost C_i and will then be able to profit from future sales of its software application. Accordingly, the δ discounted future profit stream of a developer which begins supporting a release r at time t is given by:

$$-C_i + \sum_{j=0}^{\infty} \delta^j \pi_{irct+j} \tag{4}$$

The infinite time horizon model in Equation 4 tells us that by adding support for the latest platform generation in the current time period t, the developer will earn a profit stream of $\pi_{irct}, \pi_{irct+1}, ...$ for each of the following time periods at some discounted rate $\delta \in (0, 1)$. However, if the developer decides to support in the following time period $t + 1$, then their expected profit will be given by Equation 5.

$$-\delta C_i + \sum_{j=1}^{\infty} \delta^j \pi_{irct+j} \tag{5}$$

A developer will find it profitable to support the generation at time t if profits from supporting are at least as much as from waiting another time-period. Following Gandal et al. (2000) we assume free-entry and exit subject to a fixed cost of entry. Accordingly, a developer will support the latest generation at the first time t in which Equation 6 holds.

$$\pi_{irct} \geq \frac{C_i(1 - \delta)}{\delta} \tag{6}$$

Equation 6 parsimoniously defines a developer's choice to support a new generation as one of reaching a specified threshold of profitability. However, the decision to support a new generation by developing a new software application or by adding support to an exisiting product are qualitatively different choices. For this reason, we separately model each support decision

below.

Developer Support - Entrants

Entrant developers, those without a current software application, must simultaneously decide when to support a new generation as well as which type of application to develop. Formally, an entrant developer i will support a generation r with an application in category c at time t if Equation 6 holds. For a specific entrant, the decision depends on their expected profits π_{irct} which are a function of (i) the number of users of the platform generation N^u_{rt}, (ii) the number of other 3rd-parties supporting generation r with a software application in the target category N^{3rd}_{rct}, (iii) the presence of first-party software applications in the target category $MSFT_{rct}$, and (iv) a developer idiosyncratic error term which captures a developer i's differential ability to produce a software application of category c for a platform generation r ϵ_{irc}. Additionally, an entrant developer's support decision will depend on the capabilities offered by the new generation α_r and the cost of developing a new software application, which we assume to be constant for all entrants within the category same C_c. Accordingly, an entrant developer i will support a generation r with a new software application in category c at the first time t when:

$$\pi_{irct}(N^u_{rt-1}, N^{3rd}_{rct}, MSFT_{rct}, \alpha_r, \epsilon_{irc}) \geq \frac{C_c(1-\delta)}{\delta} \tag{7}$$

The expected number of new entrants μ_{rct} is then given by the expected number of potential entrants that exceed the threshold as parameterized by common characteristics $N^u_{rt-1}, N^{3rd}_{rct}, MSFT_{rct}, \alpha_r$ and developer heterogeneity ϵ_{irc}.

$$\begin{aligned}\mu_t &= E\left[N^{entrants}_{rct} \mid N^u_{rt-1}, N^{3rd}_{rct}, MSFT_{rct}, \alpha_r\right] \\ &= E\left[\sum_{i \in I} \mathbf{1}\left(\pi_{irct}(N^u_{rt-1}, N^{3rd}_{rct}, MSFT_{rct}, \alpha_r, \epsilon_{irc}) \geq \frac{C_c(1-\delta)}{\delta}\right) \mid N^u_{rt-1}, N^{3rd}_{rct}, MSFT_{rct}, \alpha_r\right]\end{aligned} \tag{8}$$

Because developers rely on consumers to purchase and use their software applications (Gandal et al. 2000), entry is likely to be accelerated by greater user adoption:

Hypothesis 5. Developer entry will be accelerated by greater user adoption.

Additionally, entrants likely bring new functionality which is not provided by existing offerings and are therefor unlikely to be deterred by competition. Rather, entrants will likely benefit from additional developers and developer support provided by other third-party developers (SSNEs) (Singh et al. 2011).

Hypothesis 6. Developer entry will be accelerated by greater third-party support

In addition to gaining support from other third-party developers, entrants will likely also benefit from first-party development within a category as the platform owner is able to credibly signal commitment to the new generation (Waldman 1993) while showcasing the capabilities of the new generation (Gawer and Cusumano 2002).

Hypothesis 7. Developer entry will be accelerated by greater first-party support

Developer Support: Incumbents

Incumbent developers have an existing application supporting a prior generation j which makes them heterogeneous in their ability to accrue profits. As with the entrant model, we assume the fixed cost of adding support to be equal across developers within a category c. Therefore, we can write the profit of an individual developer π_{irct} as a deviation from the mean profit of all developers in their category $\overline{\pi}_{rct}$. Define such a deviation as $\nu_{irct} = \overline{\pi}_{rct} - \pi_{irct}$. Following Saloner and Shepard (1995), we can rewrite Equation 6 as follows:

$$\nu_{irct} \leq \overline{\pi}_{rct} - \frac{C(1-\delta)}{\delta} \tag{9}$$

Equation 9 shows that developers with high profitability (low ν_{irct}) support early while others wait. Note that ν_{irct} is defined over π_{irct} and, as in the entrant model, is therefore also a function with the parameters of $\pi_{irct} = f(N^u_{rt}, N^{3rd}_{rct}, MSFT_{rct}, \alpha_r)$. However, incumbents may multihome

by maintaining support of multiple generations. To accomodate this, let R_{it} denote the set of all generations supported by developer i at time t. The developer's profit function then becomes $\pi_{irct} = f(N^u_{R_{it}t}, N^{3rd}_{R_{it}ct-1}, MSFT_{R_{it}ct}, \alpha_r)$, henceforth suppressed for notational simplicity.

Average profitability will vary over time as users adopt and other developers support the new generation. To see this, use 9 to define ν^*_{rt} as the ν_{irct} of the developer that is just indifferent between supporting and not supporting the generation at time t.

$$\nu^*_{rt} \equiv \overline{\pi}_{rt} - \frac{C(1-\delta)}{\delta} \tag{10}$$

Accordingly, developers with $\nu_{irct} < \nu^*_{rt}$ will strictly prefer to support at time t. Let $H(\cdot)$ be the cumulative distribution function of all developer deviations ν_{irct}. We can then write the probability that a developer will support r in period t, if they do not yet support r, in the form of the following hazard rate function:

$$r(t) = \frac{H\left(\nu^*_{rt+1}\right) - H\left(\nu^*_{rt}\right)}{1 - H\left(\nu^*_{rt}\right)} \tag{11}$$

Clearly $r(t)$ is dependent on the distribution of profit deviations $H\left(\nu^*_{rt}\right)$, which is in turn dependent on developers' profit function π_{irct}. Accounting for these dependencies, Equation (11) can be conditioned on the parameters of π_{irct}, yielding

$$r(t; N^u_{R_{it}t}, N^{3rd}_{R_{it}ct-1}, MSFT_{R_{it}ct}, \alpha_r) = \\ \frac{H\left(\nu^*_{rt+1} \mid N^u_{R_{it}t}, N^{3rd}_{R_{it}ct-1}, MSFT_{R_{it}ct}, \alpha_r\right) - H\left(\nu^*_{rt} \mid N^u_{R_{it}t}, N^{3rd}_{R_{it}ct-1}, MSFT_{R_{it}ct}, \alpha_r\right)}{1 - H\left(\nu^*_{rt} \mid N^u_{R_{it}t}, N^{3rd}_{R_{it}ct-1}, MSFT_{R_{it}ct}, \alpha_r\right)} \tag{12}$$

Equation 12 tells us that incumbent developers' likelihood of supporting a new generation at any time period is directly a function of network effects and first-party support. As with entrants, we would expect that greater user adoption of the new generation will result in incumbent developers being more likely to support the new generation (Gandal et al. 2000).

Hypothesis 8. Incumbent developers will be more likely to support a new generation if more users adopt it

In contrast to entrant developers, however, incumbent developers already have experience with developing applications for the platform and the competitive effect of other developers is likely to outweigh the benefits of shared resources (Kretschmer and Claussen 2016).

Hypothesis 9. Incumbent developers will be less likely to support a new generation if more third-party developers support it

This, however, may not be the case with first-party products which may be used by the platform owner to signal commitment to the new generation (Waldman 1993) or to demonstrate new capabilities (Gawer and Cusumano 2002).

Hypothesis 10. Incumbent developers will be more likely to support a new generation if more first-party products support it

Developer Abandonment

Assuming free exit, a developer will choose to stop supporting (abandon hereafter) an old generation if the profits from abandoning are greater than from maintaining support. When a developer supports a single generation, the abandonment decision follows trivially from Equation 3 and a developer abandons when the per-period revenues are lesser than the per-period cost of maintaining support.

$$\pi_{irct} \leq 0 \iff \rho_{irct} \leq m_{irct} \tag{13}$$

In the case when a developer is multihoming across multiple generations, the decision becomes one of marginal profitability. To see this, let R_{it} denote the set of all generations supported by developer i at time t and $R_{it}^{\sim r}$ be the same set excluding r. Likewise, define $\pi_{ijct}^{\sim r}$, $\rho_{ijct}^{\sim r}$, $m_{ijct}^{\sim r}$ as the profits, revenues, and maintenance cost of supporting generation j if r is no longer supported. Note that $\pi_{ict}^{\sim r}$ may be less than, equal to, or greater than π_{ijct} depending on the trade-offs faced by the focal developer. For example, maintaining support for an old generation r may restrict the features that a developer can introduce in future updates, making $\rho_{ijct}^{\sim r} \geq \rho_{ijct}$. Alternatively, the developer is likely to maintain a shared code base resulting in a lower marginal

cost of maintenance when supporting more generations $m^{\sim r}_{ijct} \geq m_{ijct}$. When evaluating these trade-offs, a developer will choose to abandon the old generation and only support newer generations, henceforth *switch*, if switching will be more profitable than multihoming and doing so will net non-negative profits:

$$\sum_{j \in R^{\sim r}_{it}} \pi_{ijct} \geq \sum_{j \in R_{it}} \pi_{ijct} \text{ and } \sum_{j \in R^{\sim r}_{it}} \pi_{ijct} > 0 \tag{14}$$

If profits are negative, regardless of the developer's support decision, the developer will elect to exit the market entirely.

$$0 \geq max \left\{ \sum_{j \in R^{\sim r}_{it}} \pi_{ijct} \, , \sum_{j \in R_{it}} \pi_{ijct} \right\} \tag{15}$$

The three conditions set forth in Equations 13, 14, and 15 jointly determine how and when a developer will abandon an old generation. Further, Equation 13 is a special case of Equation 15 where $\sum_{j \in R^{\sim r}_{it}} \pi_{ijct} = 0$, confirming that abandonment for a single-homing developer is equivalent to exiting the market entirely. Equation 16 combines the potential abandonment regimes.

$$\begin{aligned} &\text{Let } S_{1irct} = \sum_{j \in R_{it}} \pi_{ijct} \text{ and } S_{2irct} = \sum_{j \in R^{\sim r} it} \pi^{\sim r}_{ijct} \\ &\begin{cases} \text{Switch} & \text{if } S_{2irct} \geq 0 \text{ and } S_{1irct} \leq S_{2irct} \\ \text{Exit} & \text{if } S_{2irct} \leq 0 \text{ and } S_{1irct} \leq 0 \end{cases} \end{aligned} \tag{16}$$

S_{1irct} denotes the profits from supporting all currently supported generations and S_{2irct} denotes the profits from no longer supporting generation r. Equation 16 formalizes the idea that a developer will switch if it is more profitable to support all generations except r and will abandon if any regime will be unprofitable. Figure 1 provides a graphical representation of Equation 16. Note that single-homing developers are not shown in Figure 1 as they occupy the infinitesimally small area where $S_{2irct} = 0$ and decision to abandon an old release is equivalent to exiting the

market. From Figure 1 and Equation 16, it is clear that the decision to abandon r is determined by the profitability of continuing to support r (S_{1irct}) whereas the decision to switch or to exit is fully determined by the developer's profitability in the absence of r (S_{2irct}) which in turn are determined by network effects and generation specific capabilities: $\pi_{irct} = f(N^u_{R_{it}t}, N^{3rd}_{R_{it}ct-1}, MSFT_{R_{it}ct}, \alpha_r)$.

Insert Figure 1 about here

As with the earlier developer support models, we would expect that greater user adoption of r would result in greater profitability and therefore lower rates of abandonment for r (Gandal and Halaburda 2016).

Hypothesis 11. The greater the number of users of the old generation, the less likely that developers will abandon it

Additionally, the cost of maintaining support of an old generation will likely be a function of the platform owner's and other developers' commitment to the old generation (Waldman 1993). This would imply that greater product support, both first-party and third-party, would increase the profitability of supporting an old generation and therefore decrease the likelihood that developers will abandon it.

Hypothesis 12. The greater the number of third-party developers supporting the old generation, the less likely that developers will abandon it

Hypothesis 13. The greater the number of first-party products supporting the old generation, the less likely that developers will abandon it

DATA AND METHODS

Our analysis is based on weekly market share data for five generations of the Windows platform, henceforth 'releases', collected from Statcounter between 2009 and 2017. For three such releases, we are able to observe market share data from the date of release as well as the end-of-life data for two of them, as shown in Figure 2. In addition to the releases observed in

Figure 2, we are also able to observe the abandonment of Windows XP over the same time-period. We combine these market share data with a complete panel of all software products supporting any version of Windows between 2009 and 2017 from a large software aggregator that collected software information directly from users' devices. This granted us with access to 85,790 software products which collectively contained 303,045 versions. Provided that planned obsolescence is a relative measure with respect to the release date and the date of subsequent releases, all timelines below are presented as weeks from the original release for adoption/support and weeks since next release for abandonment.

Insert Figure 2 about here

Figure 3 shows the rate of user adoption for each of the 3 Windows releases mentioned above. Notice that user adoption of Windows releases is a multi-year process whereby only 2 out of the 3 releases ever achieved 50% market share and required approximately 2 years to do so. This timeline is rather long considering that the time between releases in only approximately 3 years, implying that a Windows release requires two-thirds of its life-cycle to successfully capture half of the market - if it reaches that milestone at all.

Insert Figure 3 about here

A similar trend emerges when considering the time for incumbent developers to support the new release. Figure 4 shows that half of incumbent developers do not yet support the new release two years after the new release is made available. This would seem to indicate reluctant support for Microsoft's planned obsolescence strategy as developers are not quickly supporting the new release and users are similarly slow to adopt it.

Insert Figure 4 about here

Further, Figure 5 shows that developers are also slow to abandon old releases. Specifically, few developers choose to switch from one release to the newer release after the newer one is made available. Figure 5a shows that after 6 years (two Windows release cycles), fewer than 10% of developers had switched to the newer release. In contrast, developers are much more likely to exit the market entirely as shown in Figure 5b. Despite a higher likelihood of exit, developers are still slow to do so, requiring close to two years after a newer release for developers to exit the market.

Insert Figure 5 about here

Below, we further develop the utility models outlined above to construct estimation equations which will allow us to determine the drivers of succes for Microsoft's planned obsolescence strategy. In doing so, we will also allow the platform dynamics to evolve within each release as well as between releases by allowing the relevant coefficients to be time varying (Chu and Manchanda 2016). Note, that in contrast to prior research, we allow *all* network effects to be time-varying by estimating them over six semi-annual time-periods covering the three year release cycle for adoption and support. Due to the rarity of abandonment events, we estimate a single parameter for each network effect in our abandonment models.

User Model Estimation

Assuming that users' utility function (Equation 1) takes the Cobb-Douglas form (e.g., Berry et al. 1995; Rysman 2004) and that users single-home (Chu and Manchanda 2016), Equation 2 can be aggregated and estimated using OLS. Formally, we estimate the following regression model (Zhu and Iansiti 2012):

$$\ln\left(\frac{s_{rt}}{s_{jt}}\right) = \sum_{p=1}^{6}\beta_{1p}\ln(N^{u}_{rt-1}) + \sum_{p=1}^{6}\beta_{2p}\ln(N^{u}_{jt-1}) + \sum_{p=1}^{6}\beta_{3p}\ln(MSFTrt) + \sum_{p=1}^{6}\beta_{4p}\ln(MSFTjt) + \sum_{p=1}^{6}\beta_{5p}\ln(N^{3rd}rt) + \sum_{p=1}^{6}\beta_{6p}\ln(MSFTjt) + \sum_{p=1}^{6}\beta_{p} + \gamma_r + \gamma_j + \epsilon_{rjt} \tag{17}$$

Where $\gamma_r = \ln(\alpha_r - P_r)$ and $\gamma_j = \ln(\alpha_j)$ while β_p captures unobserved time-heterogeneity.

Equation 17 provides an operationalization of users' aggregate adoption decisions. Specifically, when the value of Equation 17 is greater than zero, the average user will prefer to adopt the latest platform generation and abandon the old generation in the process. Accordingly, positive coefficients indicate increased adoption likelihood due to the observed factor and negative coefficients indicate that users prefer to stay with their old generation as a result of the estimated component.

Developer Model Estimation

Developer Entry

Under mild assumptions on ν_{irc}, namely that it follows a distribution which will generate overdispersion (e.g., gamma distributed), Equation 8 can be estimated using a negative binomial model conditioned on $N^{u}_{rt-1}, N^{3rd}_{rct}, MSFT_{rct}$, and α_r. Formally, we estimate the number of new entrants in category c supporting a new release r at time t using the following negative binomial for each semi-annual period p:

$$N^{entrants}_{rct} \sim \text{Negative Binomial}(\mu_{rct}, \theta)$$
$$\log(\mu_{rct}) = \sum_{p=1}^{6}\beta_{0p} + \sum_{p=1}^{6}\beta_{1p}N^{u}_{rt-1} + \sum_{p=1}^{6}\beta_{2p}N^{3rd}_{rct} + \sum_{p=1}^{6}\beta_{3p}MSFT_{rct} + \alpha_r + \zeta_c + \epsilon_{rct} \tag{18}$$

Incumbent Developers

Assuming that changes in profitability due to network effects are proportional across releases of each OS, we allow that these changes vary over a release's life-cycle (Chu and Manchanda 2016). Specifically, we use the same time periods as in the other estimation models. Additionally, we further decompose developer heterogeneity to the random component ϵ_{irct} and controls for the developer's age_{it}, number of prior updates to their software application $updates_{it}$, and number of generations supported $generations_{it}$. Together, these additional controls capture a developer's observable ability to innovate and maintain their existing software application. Altogether, this yields the following piecewise exponential model to estimate the hazard rate shown in Equation (12):

$$r(t) = \sum_{p=1}^{6} \beta_{0p} + \sum_{p=1}^{6} \beta_{1p} N^{u}_{rt-1} + \sum_{p=1}^{6} \beta_{2p} N^{3rd}_{rct} + \sum_{p=1}^{6} \beta_{3p} MSFT_{rct} + \beta_4 age_{it} + \beta_5 updates_{it} + \beta_6 generations_{it} + \alpha_r + \zeta_c + \epsilon_{irct} \tag{19}$$

where β_{0p} is the constant hazard rate for each period $p \in [1, 6]$.

In Equation (19), we separately measure the network effects for each of the 6 time periods denoted by the subscript p. Accordingly, β_{1j} - β_{3j} capture the time varying effects of SSNEs (first-party and third-party) and CNEs. β_4 to β_6 estimate the importance of prior development experience. Meanwhile, release α_r and category ζ_c fixed effects control for time-invariant heterogeneity.

Developer Abandonment

As detailed in Equation 16, developers have two methods for abandoning an old release: (i) switch to only supporting newer releases or (ii) exit the market and stop supporting *all* releases. Equation 16 and Figure 1 also show that S_{2irct} fully determines whether a developer will switch or exit whereas S_{1irct} determines when a developer will abandon by either method. In other words,

we would be able to separately model developers at risk of switching or exit if we could perfectly observe a developer's contemporaneous counterfactual profitability, S_{2irct}. However, the fundamental problem of causal inference implies that we cannot observe both S_{1irct} and S_{2irct} and must therefore model switching and exiting as competing potential outcomes.

Provided that a developer's decision of when to abandon is determined by a threshold on S_{1irct}, derivations synonymous with those of incumbent support give us the following hazard rate functions for abandonment:

$$H^{\text{Abandon}}(\cdot) = \begin{cases} H^{\text{Switch}}(\cdot) & \text{if } S_{2irct} \geq 0 \\ H^{\text{Exit}}(\cdot) & \text{if } S_{2irct} \leq 0 \end{cases} \tag{20}$$

We must estimate H^{Abandon} because S_{2irct} is time-variant and, therefore, the hazard function, $H^{\text{Switch}}(\cdot)$ or $H^{\text{Exit}}(\cdot)$, which is applicable to a developer is itself a function of covariates. Accordingly, we jointly estimate developers' likelihood of abandoning an old release, H^{Abandon}, using Fine and Gray's (1999) method. This method removes biases introduced by separately estimating $H^{\text{Switch}}(\cdot)$ and $H^{\text{Exit}}(\cdot)$ while still allowing us to recover subdistribution-specific parameters. For consistency, we paramaterize both subdistribution hazards as in Equation 21 which incorporates controls for developer age_{it} and $updates_{it}$. Additionally, we include a control, $stayer_{it}$, for whether the developer only supports the old release and not a newer release at t as well as an interaction with Microsoft presence since single-homing developers may be differentially affected by the platform owner's first-party product strategy. Finally, we also include an indicator, EOL_{rt}, for whether the focal release has been declared end-of-life.

$$\begin{gathered} H^{\text{Abandon}}(\mathbf{X}\beta) = H^{\text{Switch}}(\mathbf{X}\beta^{\text{Switch}}) + H^{\text{Exit}}(\mathbf{X}\beta^{\text{Exit}}) \\ \mathbf{X} = \left\{ \begin{array}{c} N^{u}_{rt-1}, N^{u}_{jt-1}, N^{3rd}_{rt}, MSFT_{rt}, MSFT_{jt}, age_{it}, updates_{it}, \\ stayer_{it}, stayer_{it} \times MSFT_{rt}, EOL_{rt}, \alpha_r, \zeta_c \end{array} \right\} \end{gathered} \tag{21}$$

Identification

Our parameters of interest, which indicate the impact of the user and third-party developer base of each OS release, are identified using weekly variation in user adoption and third-party developer support and abandonment. We also rely on variance over time and across categories to estimate parameters related to developer support and abandonment as well as that of first-party products and release-specific fixed effects. Below we address issues commonly raised in simultaneous equation models and semi-structural estimation.

Market Size

In contrast to most prior structural models (i.e., Chu and Manchanda 2016), the market size is either clearly defined for several of our models and can be estimated under mild assumptions for the others. Our incumbent support and abandonment models have a clearly pre-defined "market size" as all of the incumbent third-party developers which have not yet supported the new release or abandoned the old release. Our third-party entrant model, meanwhile, side-steps the issue by considering the rate at which new third-party software applications are developed rather than the proportion of all potential new applications which support the new release. This allows us to refrain from making heroic assumptions about the potential developer pool as well as the scope and scale of developers. Likewise, our user adoption model also side-steps the issue as the dependent variable is a ratio of market shares - structurally removing the potential user market size from the equation. However, we include the number of users of the new and old release as measures of user-side network effects. For this, we make mild assumptions by considering that the size of the installed base is simply the release's market share of all desktop computers scaled by the total number of internet active individuals in the United States (Chu and Manchanda 2016). In effect, we assume that every internet connected individual has one computer for which they are making separate adoption decisions. Note that unlike other structural models for which the market size affects the dependent variable, any measurement error in our estimate of the market size would simply result in a change of magnitude for the affected

parameters and reduced efficiency. In other words, this relatively mild assumption on the size of the user base is inconsequential at best and makes our results more conservative at worst.

Simultaneity

In a simultaneous system, where actions of one agent affect those of another, there is a potential for a simultaneity confound. In our context, we expect that users adopt an OS release because developers support it, and developers support the release because users have adopted it. Typical solutions to this type of confound are the use of instrumental variables and two-stage least squares (2SLS) regression. We adopt excluded variables and two-stage least squares equations using distinct functional forms to limit the potential for such confounding. Specifically, we use standard two-stage least squares in the user adoption model. In this model, user adoption is instrumented using the Consumer Present Situation Index as well as Reddit mentions and interest for the new releases (The Conference Board 2023). We instrument for developer support using the number of venture capital deals (Pitchbook 2023). We further add efficiency to our first-stage equations by also using the total number of reported vulnerabilities (National Institute of Standards and Technology 2023), the consumer price index for computer peripherals (U.S. Bureau of Labor Statistics 2023) for the new release as well as old and new release fixed effects. This 2SLS regression is the foundation for our other models as the estimated market share is used as an independent variable in the third-party developer support and abandonment models. In effect, this means that we are using two-stage least squares for each of our estimation models, where the first-stage equation follows a functional form determined by the structural equations defined previously. This methodology has been shown to be consistent and does not hinder our ability to determine whether the parameters are statistically different from zero (Hansen 2021).

In order to assess the validity of the instruments used, we used the statistics provided by the Stata procedure xtivreg2 (Schaffer 2020). Specifically, relevance is assessed using the Cragg-Donald and Kleibergen-Paap Wald F statistics while evidence of exogeneity is presented in the Hansen J statistic. Because we are estimating a spline model, all instruments were checked using the linear form first. Using the linear form, we find that the instruments are relevant

(Kleibergen-Paap rk LM statistic = 49.860; p-value < 0.0000) and that the system of equations does not exhibit weak identification (Cragg-Donald Wald F statistic = 14.640; Kleibergen-Paap rk Wald F statistic = 8.235). We find weak evidence of over-identification - potentially violating exogeneity (Hansen J statistic = 5.832; p = .0541). However, the problematic instrument is the spline itself (Hansen J statistic without spline = 4.477; p = 0.1066). The test statistics become more complicated when estimating the spline model. The reason for this is that each spline variable enters the equation six times, ones for each segment, and for each endogenous regressor, $\frac{5}{6}$ instruments are zero by construction. This leads to poor results for the relevance (Kleibergen-Paap rk LM statistic = 11.532; p-value < 0.0000) and weak identification tests (Cragg-Donald Wald F statistic = 0.337; Kleibergen-Paap rk Wald F statistic = 0.316). However, we can again confirm the exogeneity of the instruments (Hansen J statistic = 16.185; p = .1829). Accordingly, we can conclude that the instruments are relevant and exogenous but may be sensitive to the parametric form of the test.

RESULTS

We present the results of our empirical analyses below. As mentioned above, generational platform evolution requires success across several key decisions from both users and developers, each of which is separately modeled and the results for which are presented below. Specifically, Table 1 identifies the drivers of user adoption (Hypotheses 1 through 4). Tables 2 and 3, meanwhile, demonstrate the drivers of developer support for a new release (Hypotheses 5 through 10). Table 4 concludes by identifying the drivers of two distinct types of developer abandonment (Hypotheses 11 through 13). All results are discussed in turn below.

Users

Table 1 presents 2-stage least squares estimation results for Equation 17. The results indicate that SSNEs from other users' adoption of the new release increase a user's likelihood of also adopting the new release, providing support for hypothesis H1a. Likewise, CNEs from developers supporting the new release also appear to increase a user's likelihood of adopting the

new release, supporting hypothesis H2a. These results are consistent with extant theory in platform adoption (i.e., Zhu and Iansiti 2012). However, our research also quantifies the extent to which the prior release's ecosystem hinders user adoption of the new release. Our results clearly show that both SSNEs and CNEs from the old release reduce the likelihood that a user will adopt the new release as stated in hypotheses H1b and H2b respectively. This indicates that the success of a new release may be hindered by the success of prior releases. Notably, point estimates indicate that the network effects of the new release are more important than those of the old release. We find that greater user adoption and developer support of the old release compared to the newer one - especially initially - retard the adoption of a new release. This would indicate that a platform requires a deliberate strategy to successfully evolve across generations. However, it appears that a platform's own actions are insufficient to overcome the prior releases' installed base advantage as indicated by insignificant release fixed effects and an insignificant effect of a platform owner's own products. These results do not support hypotheses H3a and H3b but provide strong evidence in support of hypothesis H4. Together, these results suggest that the success of a new release depends on the successful migration of the ecosystem developed for prior releases towards the latest release and that a platform owner's actions alone are unlikely to lead to a successful platform planned obsolescence strategy. Instead, successful evolution is likely to depend on a platform's success with the developer side of the market.

Insert Table 1 about here

Developers

As discussed previously, we model each of the developer decisions using a set of 3 estimation equations. For each estimation, we use generated regressors derived from the user adoption model. In effect, this translates to estimating the developer models using 2-Stage Least Squares with a utility-driven first-stage equation for endogenous user adoption. The estimates

from each of the estimation models are presented in their respective sections below.

Developer Entry

Table 2 presents the estimation results for Equation 18. From the results, it is clear that all network effects, CNEs and SSNEs, increase developer entry for a new release.

We find clear evidence in support of hypothesis H5, third-party developers are more likely to support a new release if users have adopted the release (CNEs). Additionally, our results indicate that user adoption is less important shortly after release, indicating that new developers have likely formed expectations of future user adoption and entrant developers are less likely to support in later periods if their expectations are not fulfilled (Chellappa and Mukherjee 2021).

In addition to the CNEs discussed above, new developers are more likely to support a new release if other developers are active in their intended category, in support of hypothesis H6. This positive SSNE indicates that third-party entrant developers value the shared employee base and amassed knowledge more than they are wary of competition with other developers (Kretschmer and Claussen 2016). This result is consistent with expectations that new developers will require greater resources than seasoned developers. The presence of first-party products also increases entrants' rate of support for the new release.

Entrant developers appear to be more likely to develop a new product for a new release if Microsoft supports the new release with a product in the intended category (hypothesis H7). This would suggest that entrant developers observe Microsoft's product strategy to determine whether the platform owner is committed to the new release and whether the target category is worthy of investment. This is furthered by the fact that Microsoft's presence is more important shortly after release than it is in later periods. Point estimates indicate that Microsoft's presence within a category has the same developer draw as increasing the third-party developer base by 1%. This may be because entrant developers look towards Microsoft in forming their expectations about the future success, and therefore suitability, of the new release.

Together, our developer entry model demonstrates that new developers benefit greatly from network effects of the new release and that the benefits derived vary by network effect type.

Specifically, our results imply entrant developers form expectations about the long-term success of a new release by looking to Microsoft's product strategy. Entrants then rely on these expectations when making development decisions shortly after release and are less likely to continue development if the expectations do not materialize. CNEs and 1st-party product support indicate that entrant developers are influenced by the potential market size and the platform owner's commitment to the new release. SSNEs also indicate that prior support from other 3rd-party developers reduces uncertainty about the market for products serving the new release and facilitates new product development.

Insert Table 2 about here

Incumbent Developers

Table 3 presents the parameter estimates for Equation 19which captures an incumbent developer's support decision.

Our results indicate that incumbent developers are more likely to support the new release shortly after its introduction if more users adopt the new release. However, their adoption is not accelerated by increased user adoption in later periods, contrary to hypothesis H8 which predicts a positive CNE. This result would suggest that early user adoption is an important signal for the success of a new release, but becomes less important as the release matures. This is likely to be the case for two reasons: (i) incumbent developers do not see a substantive drawback to multihoming across releases, and do so quickly. This would be the case if the cost of multihoming is negligible and the benefits from doing so are non-negative. (ii) developers that are unable to multihome across releases are likely to exit the market altogether. In this regime, incumbents are less likely to support the new release as more users adopt because the incumbents that are able to profit from multihoming have already done so shortly after release, indicating the importance of expectations.

Likewise, hypothesis H9 proposed that incumbent developers, like entrants, would benefit

from positive SSNEs. However, our results show that incumbent developers neither benefit from more shared resources nor are dissuaded by potential competition. Again, this would support that incumbent developers have a negligible cost to multihoming and are acting on their expectations of the new release.

Hypothesis H10 predicts that a platform owner may be able to use its own product strategy to influence expectations and increase developer support. We find support for this hypothesis during the first half of the release's life-cycle, indicating that first-party products are an important signal of the platform owner's commitment to the new release. Likewise, continued first-party support of old releases makes developers less likely to support the new release in the first half of the release's life-cycle. This is likely because incumbent developers interpret Microsoft's continued support of the old release to indicate continued commitment for the release (Waldman 1993).

Our estimation results confirm that incumbent developers make nuanced support decisions when presented with a new platform release. We find that their primary concern in making their support decision is the platform owner's first-party product strategy which signals the platform owner's commitment to the new and old releases.

Insert Table 3 about here

Developer Abandonment

As described above, developers may abandon an old release by switching to a newer release or by exiting the market altogether. We jointly estimate the two decisions and present parameter estimates for the determinants of these decisions in Table 4.

Consistent with hypothesis H11, we find that user adoption of a new release, and concurrent abandonment of prior releases, are important determinants in developer abandonment of an old release. Specifically, we observe that developers are less likely to abandon an old release

if users have not abandoned it. Likewise, developers are more likely to abandon an old release if users have already done so. Additionally, we note that switching developers are more sensitive to user adoption decisions than exiting developers. This result is consistent with our theoretical model (see Equation 16 and Figure 1) which shows that switching developers consider the marginal profitability of distinct support regimes whereas exiting developers simply continue to support the old release until it is no longer profitable to do so.

We find that third-party support of the old release differentially affects switchers and exiting developers, providing nuance to hypothesis H12. Specifically, we find that switching developers are less likely to switch if other developers continue to support the old release. This finding is consistent with the incumbent support model which suggests that the marginal cost of multihoming is sufficiently small that developers often choose to do so. Additionally, this result suggests that the marginal cost of multihoming is likely to be comparable across developers within a category. Exiting developers, meanwhile, are more likely to exit as a result of continued third-party support of the old release. In other words, exiting third-party developers are likely to be squeezed out of the market by more profitable third-party developers on the old release whereas profitable developers are able to multihome.

Contrary to the CNEs and SSNEs outlined above, our results indicate that first-party product support does not affect switchers' decision to abandon but it does affect a developer's exit likelihood. Specifically, we find evidence in support of hypothesis H13, that first-party product support of an old or newer release reduces the likelihood that a developer exits the market. This would suggest that the platform owner's presence in the third-party developer's product category brings sufficient attention to the category so as to increase a developers' profitability. However, this is not the case for single-homing developers – they are more likely to exit the market if Microsoft continues to support the old release. Such developers may be less profitable overall and, therefore, more likely to be squeezed out by Microsoft's own products.

Together our results indicate that exiting developers are sensitive to any changes which might reduce their profitability. This is consistent with our theoretical model (see Equation 16 and

Figure 1) which shows that exiting developers are those with few profitable outcomes . As a result, they are highly susceptible to third-party competition as well as to users' adoption decisions. Similarly, exiting developers are sensitive to decisions of the platform owner regarding the old release, whether that be product support or end-of-life.

The results for switching developers, meanwhile, appear to coincide with our incumbent support results. Specifically, we find that developers have a low cost to multihoming and are not dissuaded by competition with other developers. We also find that Microsoft's signaling ability is greatly reduced in the context of developer switching - neither Microsoft's product strategy nor its end-of-life strategy significantly affect a developer's switching decision. Rather, our results suggest that a successful user strategy may be the most effective tool to encourage incumbent developers to switch from an old release to a newer release.

Insert Table 4 about here

CONCLUSION

Taken together, our research provides a comprehensive understanding of platform evolution within and across generations. We also provide a generalizable framework, based in first-principles utility models, which can be applied to a wide variety of platform markets. Our modeling provides novel insights about generational platform evolution by modeling both the adoption and abandonment process of a new platform generation. We also propose a new strategic lever which is available to platform owners for signaling commitment to new and old generations through 1st-party complement development.

From a managerial standpoint, our research also cautions against (i) allowing a platform generation to become too successful and (ii) relying on classical product-based strategies (e.g., end-of-life schedules) to make prior generations obsolete. Instead, our results demonstrate that playing into the networked nature of the platform can provide several benefits from platform

owners. Namely, 1st-party product development can (i) signal commitment to the new generation given the high fixed cost of new product development, (ii) signal an end of commitment for prior generations, and (iii) promote network effects by seeding the market with new complements. Accordingly, our research suggests that Microsoft's strategies to reduce the value of prior generations by charging enterprises for continued support[3] and increase the value of new generations by offering free upgrades[4] are likely to be less effective than coordinating their first-party product strategy with their generational platform evolution strategy.

[3] https://www.techradar.com/computing/windows/microsoft-charging-for-windows-10-updates-is-a-necessar
[4] https://www.microsoft.com/en-us/windows/get-windows-11

REFERENCES

Armstrong, M. 2006. Competition in two-sided markets. ***The RAND Journal of Economics***, 37(3): 668–691. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1756-2171.2006.tb00037.x.

Baldwin, C. Y. and Clark, K. B. 2006. The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? ***Management Science***, 52(7): 1116–1127.

Berry, S., Levinsohn, J., and Pakes, A. 1995. Automobile Prices in Market Equilibrium. ***Econometrica***, 63(4): 841–890. Publisher: [Wiley, Econometric Society].

Boudreau, K. 2010. Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. ***Management Science***, 56(10): 1849–1872.

Boudreau, K. J. 2011. Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation. ***Organization Science***, 23(5): 1409–1427. Publisher: INFORMS.

Bulow, J. 1986. An Economic Theory of Planned Obsolescence. ***The Quarterly Journal of Economics***, 101(4): 729–750. Publisher: Oxford University Press.

Caillaud, B. and Jullien, B. 2003. Chicken & egg: Competition among intermediation service providers. ***The Rand Journal of Economics***, 34(2): 309. Num Pages: 20 Place: Santa Monica, United States Publisher: Rand Corporation.

Cennamo, C. 2018. Building the Value of Next-Generation Platforms: The Paradox of Diminishing Returns. ***Journal of Management***, 44(8): 3038–3069. Publisher: SAGE Publications Inc.

Cennamo, C., Ozalp, H., and Kretschmer, T. 2018. Platform Architecture and Quality Trade-offs of Multihoming Complements. ***Information Systems Research***, 29(2): 461–478. Publisher: INFORMS.

Chellappa, R. K. and Mukherjee, R. 2021. Platform Preannouncement Strategies: The Strategic Role of Information in Two-Sided Markets Competition. ***Management Science***, 67(3): 1527–1545. Publisher: INFORMS.

Chu, J. and Manchanda, P. 2016. Quantifying Cross and Direct Network Effects in Online Consumer-to-Consumer Platforms. ***Marketing Science***, 35(6): 870–893.

Clements, M. T. and Ohashi, H. 2005. INDIRECT NETWORK EFFECTS AND THE PRODUCT CYCLE: VIDEO GAMES IN THE U.S., 1994-2002*. ***Journal of Industrial Economics***, 53(4): 515–542.

Eisenmann, T., Parker, G., and Alstyne, M. W. V. 2006. Strategies for Two-Sided Markets. ***Harvard Business Review***, 84(10): 92–101. Publisher: Harvard Business School Publication Corp.

Fine, J. P. and Gray, R. J. 1999. A proportional hazards model for the subdistribution of a competing risk. ***Journal of the American Statistical Association***, 94(446): 496–509. Num Pages: 14 Place: Alexandria, United Kingdom Publisher: Taylor & Francis Ltd.

Fishman, A., Gandal, N., and Shy, O. 1993. Planned Obsolescence as an Engine of Technological Progress. ***The Journal of Industrial Economics***, 41(4): 361.

Foerderer, J., Kude, T., Mithas, S., and Heinzl, A. 2018. Does Platform Owner's Entry Crowd Out Innovation? Evidence from Google Photos. ***Information Systems Research***, 29(2): 444–460. Publisher: INFORMS.

Gandal, N. and Halaburda, H. 2016. Can We Predict the Winner in a Market with Network Effects? Competition in Cryptocurrency Market. ***Games***, 7(3): 16.

Gandal, N., Kende, M., and Rob, R. 2000. The dynamics of technological adoption in hardware/software systems: The case of compact disc players. ***The Rand Journal of Economics; Santa Monica***, 31(1): 43–61. Num Pages: 19 Place: Santa Monica, United States, Santa Monica Publisher: Rand Corporation.

Gawer, A. 2014. Bridging differing perspectives on technological platforms: Toward an integrative framework. ***Research Policy***, 43(7): 1239–1249.

Gawer, A. and Cusumano, M. A. 2002. ***Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation***. Boston, Mass: Harvard Business Review Press, 1st edition edition.

Gawer, A. and Henderson, R. 2007. Platform Owner Entry and Innovation in Complementary Markets: Evidence from Intel. ***Journal of Economics & Management Strategy***, 16(1): 1–34. Publisher: Wiley Blackwell.

Hansen, B. 2021. ***Econometrics***. Princeton University Press.

Iizuka, T. 2007. An Empirical Analysis of Planned Obsolescence. ***Journal of Economics & Management Strategy***, 16(1): 191–226. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1530-9134.2007.00137.x.

Katz, M. L. and Shapiro, C. 1994. Systems Competition and Network Effects. ***The Journal of Economic Perspectives (1986-1998); Nashville***, 8(2): 93.

Kretschmer, T. and Claussen, J. 2016. Generational Transitions in Platform Markets—The Role of Backward Compatibility. ***Strategy Science***, 1(2): 90–104. Publisher: INFORMS.

Lee, I. H. and Lee, J. 1998. A Theory of Economic Obsolescence. ***The Journal of Industrial Economics***, 46(3): 383–401. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-6451.00077.

Li, Z. and Agarwal, A. 2017. Platform Integration and Demand Spillovers in Complementary Markets: Evidence from Facebook's Integration of Instagram. ***Management Science***, 63(10): 3438–3458. Publisher: INFORMS.

National Institute of Standards and Technology 2023. NVD - Vulnerabilities.

Parker, G. and Van Alstyne, M. 2005. Two-Sided Network Effects: A Theory of Information Product Design. ***Management Science***, 51(10): 1494–1504.

Pitchbook 2023. PE, VC and M&A Deals Data | PitchBook.

Rysman, M. 2004. Competition between Networks: A Study of the Market for Yellow Pages. ***The Review of Economic Studies***, 71(2): 483–512. Publisher: [Oxford University Press, Review of Economic Studies, Ltd.].

Saloner, G. and Shepard, A. 1995. Adoption of Technologies with Network Effects: An Empirical Examination of the Adoption of Automated Teller Machines. ***The RAND Journal of Economics***, 26(3): 479–501. Publisher: [RAND Corporation, Wiley].

Schaffer, M. E. 2020. XTIVREG2: Stata module to perform extended IV/2SLS, GMM and AC/HAC, LIML and k-class regression for panel data models. ***Statistical Software Components***. Publisher: Boston College Department of Economics.

Shapiro, C. and Varian, H. R. 1998. ***Information Rules: A Strategic Guide to the Network Economy***. Boston, Mass: Harvard Business Review Press.

Singh, P. V., Tan, Y., and Mookerjee, V. 2011. Network Effects: The Influence of Structural Capital on Open Source Project Success. ***MIS Quarterly***, 35(4): 813–829. Publisher: Management Information Systems Research Center, University of Minnesota.

Swan, P. L. 1970. Durability of Consumption Goods. ***The American Economic Review***, 60(5): 884–894. Publisher: American Economic Association.

The Conference Board 2023. US Consumer Confidence.

Tiwana, A. 2015. Evolutionary Competition in Platform Ecosystems. ***Information Systems Research***, 26(2): 266–281. Publisher: INFORMS.

Tiwana, A., Konsynski, B., and Bush, A. A. 2010. **Research Commentary** —Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. ***Information Systems Research***, 21(4): 675–687.

U.S. Bureau of Labor Statistics 2023. CPI Home : U.S. Bureau of Labor Statistics.

Waldman, M. 1993. A New Perspective on Planned Obsolescence. ***The Quarterly Journal of Economics***, 108(1): 273–283.

Waldman, M. 1996. Planned Obsolescence and the R&D Decision. ***The RAND Journal of Economics***, 27(3): 583.

Wen, W. and Zhu, F. 2019. Threat of platform-owner entry and complementor responses: Evidence from the mobile app market. ***Strategic Management Journal***, 40(9): 1336–1367. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smj.3031.

Zhu, F. 2019. Friends or foes? Examining platform owners' entry into complementors' spaces. ***Journal of Economics & Management Strategy***, 28(1): 23–28. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jems.12303.

Zhu, F. and Iansiti, M. 2012. Entry into platform-based markets. ***Strategic Management Journal***, 33(1): 88–106. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smj.941.

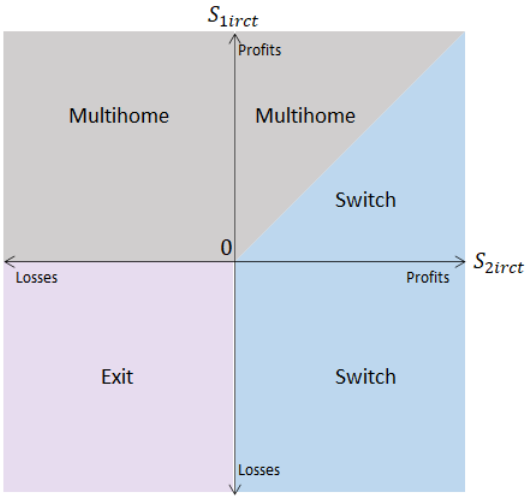


FIGURE 1
Abandonment Regimes

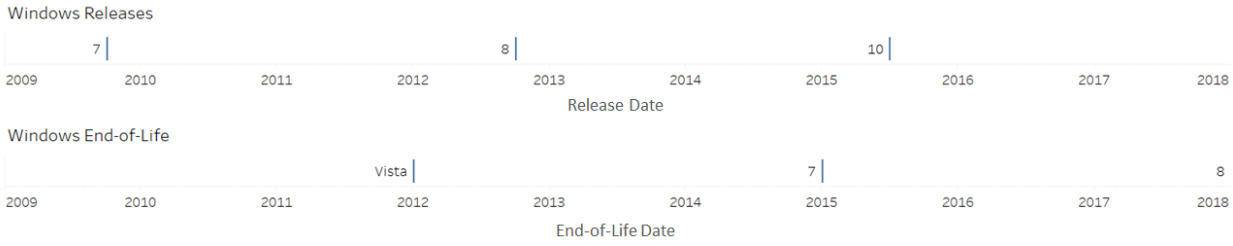


FIGURE 2
Timeline of Windows Releases

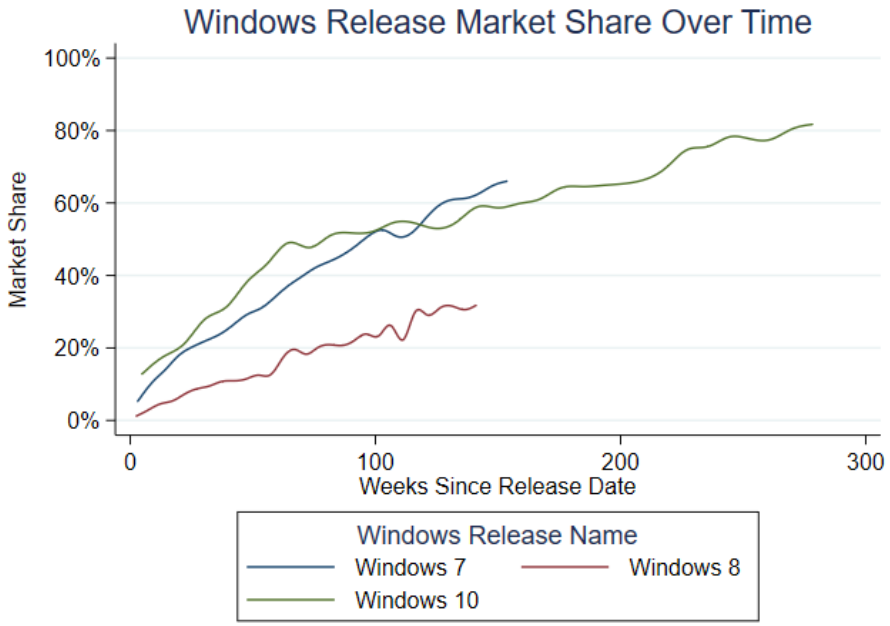


FIGURE 3
Windows Releases' Market Share Over Time

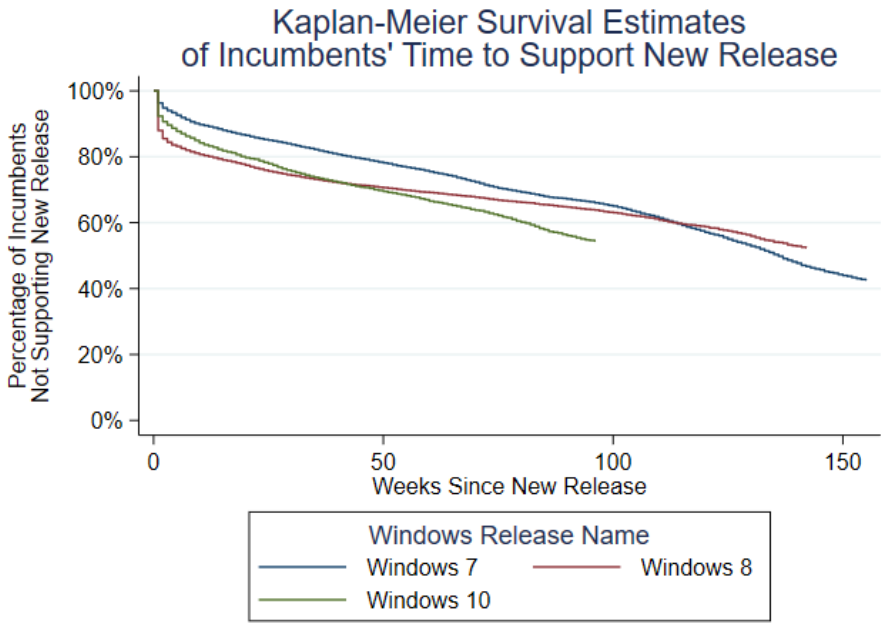
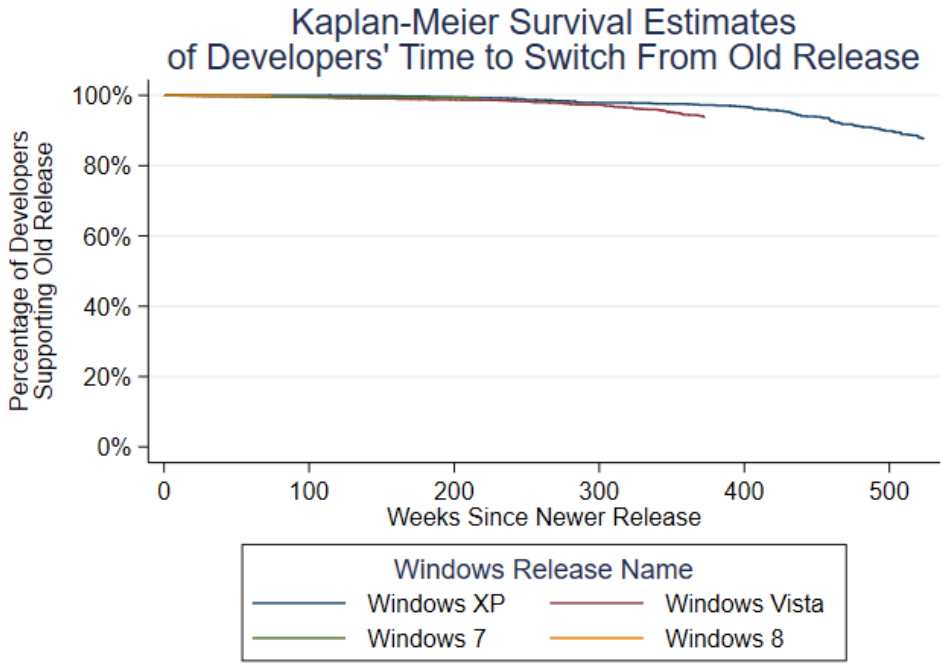
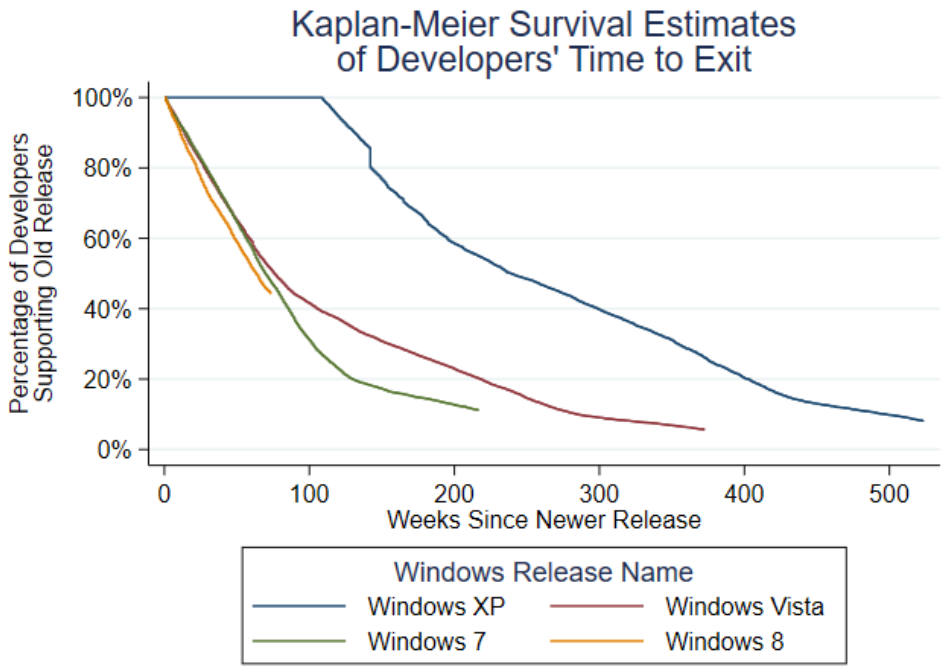


FIGURE 4
Kaplan-Meier Plot of Incumbent Developer Support for New Windows Releases



(a) Developer Switching from Old Release to Newer Release



(b) Developer Exiting Market Entirely

FIGURE 5
Kaplan-Meier Plot of Developer Abandonment After a Newer Windows Releases

TABLE 1
2-Stage Least Squares Estimates of User Adoption

	Relative User Preference
L.Number of Users (Log; New)	1.336***
	(0.172)
L.Number of Users (Log; Old)	-1.112***
	(0.0380)
L.Number of Microsoft Apps (Log; New)	-0.0311
	(0.0427)
L.Number of Microsoft Apps (Log; Old)	0.0561
	(0.0363)
L.Number of 3rd-Party Apps (Log; New)	0.308***
	(0.0616)
L.Number 3rd-Party Apps (Log; Old)	-0.196*
	(0.0937)
Relative Weeks Since Release (Log)	-0.417**
	(0.135)
Constant	-3.393
	(2.720)
New Release Fixed Effects	Yes
Old Release Fixed Effects	Yes
Observations	1118
Choice Groups	9
R2 Within	0.975
R2 Between	1.000
R2 Overall	0.990
Chi2	1774.6

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

TABLE 2
Piecewise Negative Binomial Model of Developer Entry

	Entrants
<26 Weeks	0.347
	(0.607)
26-52 Weeks	-3.800***
	(1.084)
52-78 Weeks	-8.849***
	(1.148)
78-104 Weeks	-11.41***
	(1.445)
104-130 Weeks	-10.24***
	(1.874)
>130 Weeks	-17.39***
	(2.169)
Number of Users (Log)	-0.0591
<26 Weeks	(0.0390)
Number of Users (Log, t-1)	0.132*
26-52 Weeks	(0.0671)
Number of Users (Log, t-1)	0.401***
52-78 Week	(0.0708)
Number of Users (Log, t-1)	0.580***
78-104 Weeks	(0.0909)
Number of Users (Log, t-1)	0.497***
104-130 Weeks	(0.118)
Number of Users (Log, t-1)	0.885***
>130 Weeks	(0.132)
Microsoft Presence	0.170**
<26 Weeks	(0.0567)
Microsoft Presence	0.0398
26-52 Weeks	(0.0653)
Microsoft Presence	0.260***
52-78 Weeks	(0.0723)
Microsoft Presence	0.251**
78-104 Weeks	(0.0792)
Microsoft Presence	0.135
104-130 Weeks	(0.0938)
Microsoft Presence	0.101
>130 Weeks	(0.0898)
Category Support (Log; 3rd)	0.165**
<26 Weeks	(0.0558)
Category Support (Log; 3rd)	0.227***
26-52 Weeks	(0.0576)
Category Support (Log; 3rd)	0.213***
52-78 Weeks	(0.0563)
Category Support (Log; 3rd)	0.0902
78-104 Weeks	(0.0556)
Category Support (Log; 3rd)	0.124*
104-130 Weeks	(0.0596)
Category Support (Log; 3rd)	0.147*
>130 Weeks	(0.0621)
log(r)	0.197
	(0.178)
log(s)	0.902***
	(0.203)
Category Fixed Effects	Yes
Release Fixed Effects	Yes
Observations	7372

Standard errors in parentheses
* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

TABLE 3
Piecewise Exponential Model Estimates of Incumbent Support

	Incumbent Support
Baseline Hazard Rate <26 Weeks	-52.95*** (14.61)
Baseline Hazard Rate 26-52 Weeks	-48.78*** (14.82)
Baseline Hazard Rate 52-78 Weeks	-45.35** (15.10)
Baseline Hazard Rate 78-104 Weeks	-37.49* (15.28)
Baseline Hazard Rate >130 Weeks	35.02 (19.52)
L.Number of Users (Log; New) <26 Weeks	0.230*** (0.0457)
L.Number of Users (Log; New) 26-52 Weeks	0.440*** (0.103)
L.Number of Users (Log; New) 52-78 Weeks	0.328* (0.129)
L.Number of Users (Log; New) 78-104 Weeks	0.0771 (0.170)
L.Number of Users (Log; New) 104-130 Weeks	-0.898 (0.499)
L.Number of Users (Log; New) >130 Weeks	-2.071*** (0.499)
L.Number of Users (Log; Old) <26 Weeks	1.010*** (0.239)
L.Number of Users (Log; Old) 26-52 Weeks	0.597* (0.237)
L.Number of Users (Log; Old) 52-78 Weeks	0.621* (0.250)
L.Number of Users (Log; Old) 78-104 Weeks	0.409 (0.228)
L.Number of Users (Log; Old) 104-130 Weeks	-0.848** (0.315)
L.Number of Users (Log; Old) >130 Weeks	-1.616*** (0.295)
Log Age	0.00941 (0.0246)
Log Updates	0.339*** (0.0116)
Number of OS Releases Supported	4.630*** (0.0184)
Category Fixed Effects	Yes
Release Fixed Effects	Yes
Observations	2269621

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

	Incumbent Support
Microsoft Presence (New) <26 Weeks	0.174*** (0.0438)
Microsoft Presence (New) 26-52 Weeks	0.249*** (0.0656)
Microsoft Presence (New) 52-78 Weeks	0.250** (0.0881)
Microsoft Presence (New) 78-104 Weeks	-0.353** (0.112)
Microsoft Presence (New) 104-130 Weeks	-0.438** (0.153)
Microsoft Presence (New) >130 Weeks	-0.120 (0.183)
Microsoft Presence (Old) <26 Weeks	-0.0146 (0.0591)
Microsoft Presence (Old) 26-52 Weeks	-0.141* (0.0698)
Microsoft Presence (Old) 52-78 Weeks	-0.0541 (0.0953)
Microsoft Presence (Old) 78-104 Weeks	0.000862 (0.121)
Microsoft Presence (Old) 104-130 Weeks	0.324* (0.149)
Microsoft Presence (Old) >130 Weeks	0.714*** (0.186)
3rd-Party Category Support (Log) <26 Weeks	-0.00667 (0.0511)
3rd-Party Category Support (Log) 26-52 Weeks	0.0710 (0.0574)
3rd-Party Category Support (Log) 52-78 Weeks	-0.204** (0.0636)
3rd-Party Category Support (Log) 78-104 Weeks	-0.147 (0.0758)
3rd-Party Category Support (Log) 104-130 Weeks	0.155* (0.0743)
3rd-Party Category Support (Log) >130 Weeks	-0.0798 (0.0837)

TABLE 4
Competing Risk Models of Developer Abandonment

	Abandon	Exit
L.Number of Users (Log; Old)	-2.697***	-0.317***
	(0.255)	(0.0260)
L.Number of Users (Log; Newer)	0.116*	0.0338***
	(0.0467)	(0.00281)
Microsoft Presence (Old)	-0.186	-0.0703***
	(0.180)	(0.0206)
Microsoft Presence (Newer)	0.208	-0.113***
	(0.231)	(0.0156)
3rd-Party Products (log)	-1.583***	0.0516**
	(0.236)	(0.0195)
OS Release EOL	45.11	0.770***
	(.)	(0.0462)
Log Age	0.345	0.0633***
	(0.193)	(0.0133)
Log Updates	0.0918	-0.441***
	(0.0594)	(0.00677)
Stayer		0.420***
		(0.0204)
Stayer × Microsoft Presence (Old)		0.0688**
		(0.0220)
Category Fixed Effects	Yes	Yes
Release Fixed Effects	Yes	Yes
Observations	3195733	5863682

Standard errors in parentheses

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$