




Effect of Release Timing of App Innovations based on Mobile Platform Innovations

Franck Soh ^a and Varun Grover^b

^aInformation Systems and Supply Chain Management Department, Bryan School of Business and Economics, The University of North Carolina at Greensboro, Greensboro, North Carolina, USA; ^bInformation Systems Department, Sam M. Walton College of Business, University of Arkansas, Fayetteville, Arkansas, USA

ABSTRACT

This study focuses on app innovations based on mobile platform innovations (MPIs), examining how app developers can time the app innovations release to best leverage MPIs and increase app financial performance. We suggest that the performance is contingent on the adoption curve of mobile platform generations and the level of backward compatibility of the MPIs. We find support for our hypotheses after analyzing 1,213 MPI-based app innovations on the iOS mobile platform ecosystem. The main theoretical contribution of this study, supported empirically, is to better understand the role of the platform generation adoption curve and MPIs' level of backward compatibility in the assessment of the effect of MPI-based app innovation release timing on complementor's performance. We encourage third-party developers to create MPI-based app innovations more prominently and release them early during the growth stage of the adoption curve while prioritizing MPIs with no backward compatibility.



KEYWORDS

Mobile platforms; innovation; release timing; backward compatibility; adoption curve; app releases; online innovations; platform ecosystems


Introduction

Mobile platforms provide important entrepreneurial opportunities for third-party developers. In 2017, the Android mobile platform passed the milestone of more than 2 billion monthly active Android devices [60] while the iOS mobile platform reached 1.3 billion active devices in 2018 [9]. Together the iOS and Android mobile platforms are active on more than 3 billion devices monthly. This means that any third-party developer either a company or an individual that proposes a service or content through Android or iOS can reach a market of more than 3 billion active devices. Several Google services (e.g., Gmail, Google Play, Google Maps, and YouTube) offered on mobile platforms have reached 1 billion users [49]. While these services can be accessed through various platforms, the primary driver is app users on mobile platforms. For example, 75 percent of the 1 billion Gmail's user base access the service through mobile devices [37].

However, despite having access to a large market through mobile platforms, third-party developers face serious challenges to maintain or increase the performance of their

CONTACT Franck Soh  f_sohnoume@uncg.edu  Information Systems and Supply Chain Management Department, Bryan School of Business and Economics, Office 489, Bryan Building, The University of North Carolina at Greensboro, Greensboro, NC-27407, USA.

Color versions of one or more of the figures in the article can be found online at www.tandfonline.com/xxxx.

 Supplemental data for this article can be accessed on the [publisher's website](#)

© 2020 Taylor & Francis Group, LLC

apps over time. Mobile platform ecosystems are hypercompetitive. In 2017, the number of apps submitted on the iOS platform averaged 9,000 per month [59]. Most top performing apps fail to sustain their performance over time (See [32]). In fact, apps lose 90 percent of their active users 30 days after the installation [8]. To compete in such highly competitive ecosystems, it is important that third-party developers leverage mobile platform innovations (MPIs) such as ARKit, SiriKit, and CoreML¹ in order to create new app features and increase app performance. MPIs represent platform technologies not previously available to developers adding to current technologies mix. MPIs do not include changes in existing platform technologies. Following previous research on how to profit from innovations [64, 63], we argue that in hypercompetitive environments, release timing is critical. Such markets change rapidly and have narrow product-market opportunities. Depending on whether the timing to enter a market is right or wrong, firms might seize or lose these important opportunities [61]. App developers' views about release timing are divergent. For example, HSBC and Discover introduced FaceID-based authentication feature 10 months [48] and 4 months [47], respectively, after the release of iOS FaceID. Thus, there is a practical dilemma about release timing.

While previous research highlights the importance of external developers for platform firms [46], a minimal amount is known about how these developers can leverage mobile platforms to increase their app performance. Against this backdrop, we propose to study the extent to which app developers leverage MPIs to improve app performance in a hypercompetitive environment by answering the following research question: *what is the effect of MPI-based app innovations release timing on app performance?* We define MPI-based app innovations as new app features (e.g., authentication features) that are created based on a given MPI. MPI-based app innovations can be released as new apps or updates of existing apps. To address the research question, we build on literature which suggests that entry timing plays an important role in innovation performance [7]. We argue that release timing is critical for MPI-based app innovations to realize greater app performance. Following the structure conduct performance (SCP) paradigm ([14, 50]), we posit that mobile platform characteristics drive the effectiveness of release timing. We suggest that the effect of release timing is contingent on the adoption curve of mobile platform generations and the level of backward compatibility of MPIs. We posit that the growth stage offers a window of opportunity during which MPI-based app innovations that are released early have an advantage over those that released late. Furthermore, we argue that during the maturity and decline stage, early entrants lose their advantage over late entrants. Finally, we suggest that during the growth stage, the advantage of early entrants over late entrants is reinforced when the MPIs supported has no backward compatibility.

To test our research model, we collected data about MPI-based app innovations that support the following iOS MPIs: FaceID, CoreML, ARKit, Apple Pay, TouchID and "Hey Siri." These MPI-based app innovations were released during the period September 2014 to July 2018. The results of our analyses support our hypotheses and thereby provide guidance on how companies can maximize the impact of their MPI-based app innovations. The rest of the paper is organized into four sections. The next section provides the theoretical background of the study. We present the role of MPIs, release timing, and adoption curve. The third section lays out the research model and the hypotheses

development regarding the impact of MPI-based app innovations release timing on app performance. The fourth section describes the empirical study including method, analysis, and findings. Finally, the fifth section discusses the contributions, implications, and limitations of the study.

Theoretical Background

In a hypercompetitive mobile environment with significant potential entrepreneurial opportunities [51], achieving superior performance in the market is an important concern for third-party developers. There is a large body of literature about app performance in mobile platforms (e.g., [12, 38, 58, 76, 77]). Prior literature emphasizes the effect of mobile platform technologies (e.g., toolkits) on app developer innovation [31, 72]. However, the influence of mobile platform technologies on app performance is poorly acknowledged. The goal of this study is to unravel how third-party developers can leverage MPIs (i.e., new mobile platform technologies) to achieve higher app performance. To do this, in the following sections, we discuss performance, the role of MPIs, release timing strategies, and the adoption curve to lay out the theoretical foundation for the study.

App Performance

The literature has not been consistent with the measure of performance in mobile platform ecosystems. Previous studies use metrics such as the number of downloads [58], app ratings [41], and revenues [43] which provide a limited assessment of app performance. Increasingly, scholars consider app performance to be accurately captured by app ranking in top charts [32, 40]. App ranking is a comprehensive metric that includes several factors such as app rating, user reviews, user retention, and the number of downloads [66]. Apps appear in different top charts based on the category of the app (e.g., finance, and lifestyle), the type of app (i.e., free versus paid versus grossing apps), the type of mobile device (e.g., handheld versus tablet). The rank of the app in the top chart indicates the app performance relative to competitor apps. Apps with the lowest rank are considered leaders in the market in terms of creativity, and innovativeness [5]. As MPI-based app innovations successfully reach a large market size, the app rank in the top chart improves signaling the technological leadership of the app in the market. Online Supplemental Appendix A presents a review of previous studies (including the performance variables) that examine the antecedents of app performance in mobile platform ecosystems. In this study, we argue that the impact of an MPI-based app innovation on app performance represents third-party developers' value appropriation and is tied to the size of the market the app innovation can successfully reach. In the following section, we describe the role of MPIs in facilitating app performance.

Role of Mobile Platform Innovations

Mobile platforms provide the foundation for apps to run. Moreover, mobile platforms provide the functional logic for the mobile device to be operational. Each generation of mobile platform introduces three types of MPIs: (1) apps, (2) core services, and (3) support for new hardware components (i.e., sensors). The first type includes the creation

of new apps or the improvement of existing apps that end-users can interact with and/or third-party developers can support. For example, “Hey Siri” is a type 1 MPI that was introduced in the fifth generation of the iOS platform. The second type includes the creation of new core services or the improvement of existing ones. End-users cannot use these core services unless they are supported by apps. Thus, the second type of MPI is uniquely intended for third-party developers. For example, the eleventh generation of iOS mobile platform introduced augmented reality and machine learning that only third-party developers can use to create augmented reality and machine learning apps for end-users. The third type is related to the addition of new hardware components (e.g., sensors and chips) or the improvement of existing ones. All MPIs that are intended to support or control a sensor (e.g., fingerprint sensor, TrueDepth camera system) or chip (e.g., NFC chip) are considered type 3. For example, the eighth generation of iOS mobile platform introduced touch ID, a fingerprint sensor that can be used for authentication. These three types of MPIs have different levels of backward compatibility. Compared to types 1 and 2 MPIs, type 3 MPIs are less likely to be backward compatible because they are highly dependent on the hardware (i.e., specific features of the physical device). Similarly, type 1 MPIs are more likely to be backward compatible compared to type 2 MPIs.

MPIs play an important role in app innovations. Each generation of the mobile platform (e.g., iOS 11) introduces several innovations (e.g., ARKit, FaceID, and CoreML). Third-party developers have the option to create app innovations based on these MPIs by proposing MPI-based app features. This study focusses on MPI-based app innovations. MPIs facilitate app innovations through toolkits including application programming interfaces (APIs) [10, 15, 75, 19, 20, 11, 70, 71].

Furthermore, MPIs play an important role in influencing the *reprogrammability* of mobile devices. We define the reprogrammability of a mobile device as the extent to which its functional logic can be extended to include additional functions (e.g., audio editing, video recording, and word processing). A mobile device that is highly reprogrammable is characterized by a separation between the functional logic and the device [75]. Therefore, the level of backward compatibility of MPIs influences the reprogrammability of mobile devices. The MPIs that are not backward compatible are the least distant to the device (type 3) while the MPIs that are backward compatible are the most distant to the device (type 1). Hence, more mobile devices are reprogrammable by app innovations supporting MPIs that are backward compatible. The level of backward compatibility of MPIs is significant since it influences the market that can be reached by app innovation.

Role of Release Timing

Release timing is important for the impact of MPI-based app innovations as they build on MPIs. It represents the entry timing of MPI-based app innovations (i.e., app update or new app with an MPI-based app feature) in the ecosystem. Products’ entry timing is an important concept to understand new product development performance. Prior literature extensively discusses the advantages associated with an early and late entry timing in the market. Through mechanisms such as technology leadership, preemption of scare assets, and switching costs, early entrants can outperform late entrants [16]. Early entrants build technology leadership through experience and R&D patenting. Moreover, they preempt late entrants by occupying geographic and distribution channel spaces. Early entrants

penetrate the market on a large scale compared to late entrants that target small-scale market niches. Finally, by increasing switching costs, they deter consumers to adopt late entrants' products. Early entrants face less competition, increasing their presence in the market. Late entrants face more competition and incur the costs of evaluating the competitors' offerings [33]. Since in the early stage there is no dominant brand or design in the market, early entrants face less difficulty to influence customers' attitudes and perceptions and build brand loyalty [28]. Furthermore, since customers are exposed to the early entrants' offering for a longer time period, they develop more familiarity with the early entrants' offerings.

Nevertheless, early entrants face high uncertainty risks because of lack of information [33]. Because of that uncertainty, they might not be able to choose the correct positioning for their offerings. Moreover, they might not be able to undertake the right competitive strategy. On the other hand, late entrants benefit from information and learning opportunities [25]. They can learn from early entrant's incorrect positioning and capture shifts in consumers' preferences to better position their offering in the market. Moreover, they can outperform early entrants by introducing products of high quality, differentiating themselves from early entrants' offerings. Thus, understanding the time to enter a market is difficult since early and late entry timings provide various advantages to compete in the market. Previous research suggests that in such competitive environments there is an optimal entry timing [29] or a window of opportunity [61] during which it is advantageous to enter a market. Firms that enter the market during the window of opportunity are more likely to survive. Therefore, the concept of a window of opportunity is important to understand the effect of MPI-based app innovation release timing on app performance.

Role of the Adoption Curve of Mobile Platforms

The adoption curve is important to understand the impact of MPI-based app innovations. According to the theory of innovation diffusion, the adoption curve follows an S curve [53]. The adoption curve goes through two major stages including the growth stage, and the maturity and decline stage. The growth stage represents the stage during which the number of adopters is rising. During the maturity and decline stage, the number of adopters slowly stops increasing before decreasing. The adoption curve is obtained by cumulating the number of adopters over time (See [Figure 1](#)). The S curve is characterized by five categories of adopters including innovators (2.5 percent of adopters), early adopters (13.5 percent of adopters), early majority (34 percent of adopters), late majority (34 percent of adopters), and laggards (16 percent of adopters) [53]. The ending point of the growth stage and the start point of the maturity and decline stage are determined empirically by observing the adoption curve. The point of the adoption curve where the curve becomes flat indicates the ending of the growth stage and the beginning of the maturity and decline stage. We focus on the adoption curve of each generation of mobile platforms. The performance of apps is heavily influenced by demand heterogeneity across the adoption curve of mobile platforms (see [52]). The two stages of the adoption curve of mobile platforms are distinct in terms of the categories of adopters. While laggards mostly appear during the maturity and decline stage, all the other four categories mostly appear during the growth stage. Compared to the other categories, laggards are more risk-averse and skeptic to innovations [53]. Thus, MPI-based app innovations released during the

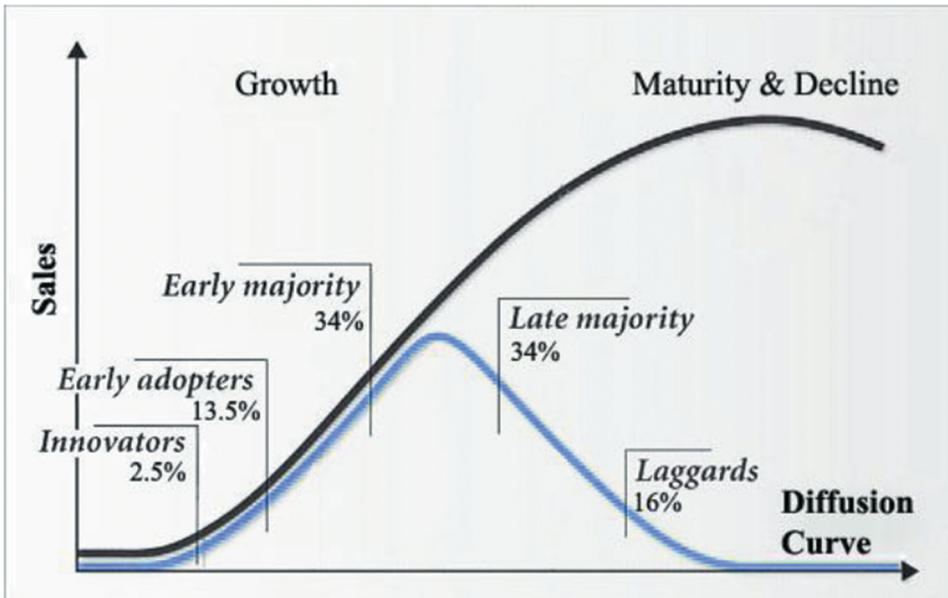


Figure 1. Adoption curve of innovations.⁴

growth stage face a different market of mobile platform users compared to those released during the maturity and decline stage.

Research Model and Hypotheses Development

We propose the following research model (See Figure 2) to understand how MPI-based app innovation release timing can increase performance. We argue that the effect of MPI-based app innovation release timing is contingent on the adoption curve of mobile platform generations. Moreover, we posit that the effect of MPI-based app innovation release timing is influenced by the level of backward compatibility of the MPI. Table 1 presents the construct definitions.

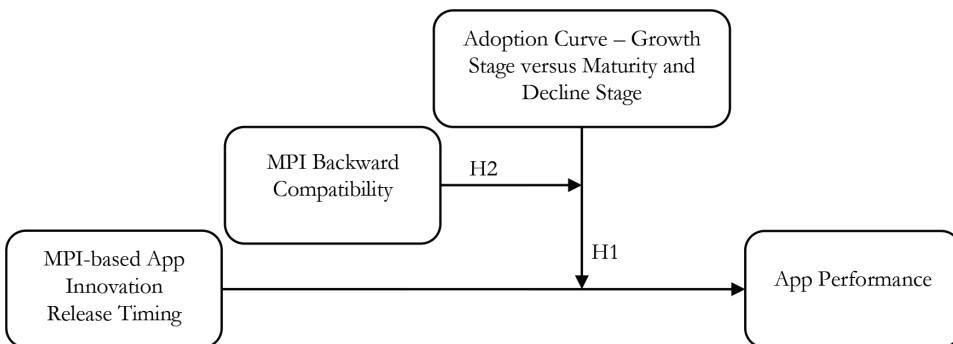


Figure 2. Research model.

Table 1. Construct definitions.

Construct	Definition	References
MPI-based App Innovation Release Timing	Length of time between the release dates of the MPI-based app innovation and the first mover MPI-based app innovation (i.e., first app to support the MPI)	[36]
MPI Backward Compatibility Adoption Curve	Extent to which an MPI is supported by older generations of mobile devices released before the MPI's release date.	[24]
App Performance	Number of adopters of an innovation. App position in top charts	[53] [38]

An MPI-based app innovation release timing represents the length of time between the release dates of the MPI-based app innovation and the first mover MPI-based app innovation (i.e., the first app to support an MPI) (See Figure 3). The effect of release timing is contingent upon the window of opportunity which emerges during the adoption curve. Compared to the maturity and decline stage, the growth stage offers a window of opportunity for MPI-based app innovations. This is the period during which MPI-based app innovations can successfully reach the market. The rate of success is high when MPI-based app innovations target a market that is still unreached since it is more challenging to influence a market that has already been reached by competitor MPI-based app innovations.

Effect of the Adoption Curve

We argue that during the growth stage, early entrants have a greater effect on app performance than late entrants. During the growth stage, the size of the unreached market at the time of entry of an MPI-based app innovation decreases over time. The rate of growth of the market is the highest at the beginning of the adoption curve of mobile platform generations and progressively diminishes over time. For example, 17 percent of the market adopted iOS 11 during the first week after its release (see Online Supplemental Appendix C). This 17 percent represents the unreached part of the market for any MPI-based app innovation that enters the market during that first week. Moreover, the following week (i.e., the second week after the iOS 11's release date), an additional 12 percent of the market adopted iOS 11. Compared to MPI-based app innovations that entered during the first week of iOS 11, those that enter during the second week of iOS 11 target an unreached market of smaller size. Until the end of the growth stage, the size of the unreached market progressively diminishes. Thus, in

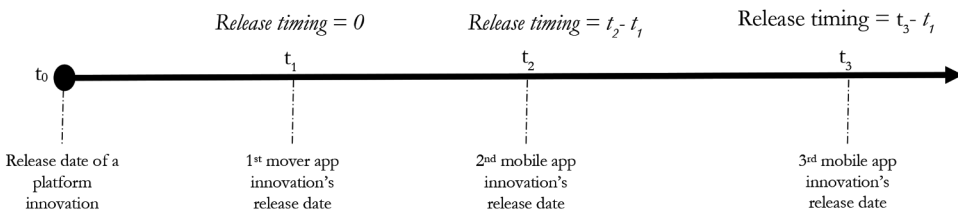


Figure 3. Conceptualization of app innovation entry timing.

terms of the size of the unreached market, early entrants have an advantage over late entrants during the growth stage.

Furthermore, throughout the growth stage, different categories of mobile platform generation's adopters emerge in the following chronological order: innovators, early adopters, early majority, and late majority according to the theory of innovation diffusion [53]. At the beginning of the growth stage, the mobile platform's adopters are classified as innovators while at the end of the growth stage mobile platform generation's adopters are classified as the late majority. The main difference among the categories of mobile platform generation's adopters is the attitude toward innovation. Innovators are more likely to take risks and try new features while the late majority is more skeptical about innovation [53]. The late majority uses innovation after being certain of the quality of the innovation. Thus, it justifies its decision to adopt on the experience of other adopters. The level of skepticism increases while the willingness to take risks decreases throughout the adoption curve [53]. MPI-based app innovations that enter early during the growth stage target an unreached market that is willing to take risks, thus more likely to adopt innovation. However, MPI-based app innovations that enter late during the growth stage target an unreached market that is skeptical about innovation, thus less likely to adopt innovation. Late during the growth stage, new mobile platform generation adopters prefer MPI-based app innovations that have been tried by previous adopters suggesting that they are more favorable for MPI-based app innovations released early than those released late. Therefore, in terms of the attitude of the unreached market toward innovation, early entrants have an advantage over late entrants during the growth stage. Based on these two reasons regarding the size and attitude of the unreached market, we argue that:

Hypothesis 1a (HQ1a): During the growth stage, the earliness of MPI-based app innovations has a positive effect on app performance.

During the maturity and decline stage, there is a minimal amount to no growth in the market. Early and late entrants are no more distinguishable based on the size of the unreached market. During the maturity and decline stage, most of the market has been reached, and the lack of growth in the market causes early entrants to lose their advantage over late entrants in terms of the size of the unreached market. Moreover, during the maturity and decline stage, new mobile platform generation adopters fall into the category of laggards who are highly skeptical about innovation [53]. MPI-based app innovations that enter during the maturity and decline stage do not catch the interest of laggards who are more likely to not adopt, take a long time before making the decision to adopt, or adopt MPI-based app innovations that are already being used by most mobile platform users. Thus, in terms of the attitude of the unreached market toward innovation, early entrants lose also their advantage over late entrants during the maturity and decline stage. Therefore, we argue that:

Hypothesis 1b (HQ1b): During the maturity and decline stage, the earliness of MPI-based app innovations release timing has no influence on app performance.

Effect of Backward Compatibility

The growth stage provides a window of opportunity for MPI-based app innovations. Moreover, the significance of this window of opportunity is influenced by the level of backward compatibility of the MPIs. Backward compatibility represents the extent to which an MPI is supported by older generations of mobile devices released before the MPI's release date (see [24]). For example, ARKit and FaceID are two MPIs introduced in 2017 by Apple. ARKit is backward compatible while FaceID is not. ARKit can be used on the latest generations (by the release date of iOS 11) of mobile devices (e.g., iPhone 8 and iPhone 8 plus) and on some older generations (e.g., iPhone 6s, iPhone 6s Plus, iPhone SE, iPhone 7, iPhone 7 Plus). However, FaceID can only be used on the latest generation (by the release date of iOS 11) of mobile device (i.e., iPhone X).

Release timing during the window of opportunity is critical when MPIs are not compatible with previous generations of mobile device because the size of the market that can be reached is considerably reduced to the latest generations of mobile devices. For example, MPI-based app innovations supporting FaceID can only reach iPhone X users, while MPI-based app innovations supporting ARKit can reach users of iPhone X and older iPhone generations (e.g., iPhone 6s). Considering the limited size of the market that can be reached during the growth stage, late entrants have a serious disadvantage over early entrants because the competition is fierce. MPI-based app innovations that are released later during the growth stage face high competition from incumbent MPI-based app innovations. When the MPIs are compatible with previous generations of mobile platforms, late entrants face less competition from incumbent MPI-based app innovations since the size of the market that is still unreached is large. Late entrants can differentiate their MPI-based app innovations from early entrants' MPI-based app innovations (i.e., high reprogrammability) to target the unreached market.

Moreover, during the growth stage, the MPIs with no backward compatibility offer a niche market for MPI-based app innovations. Users are required to obtain the latest generation of mobile device (e.g., iPhone X) and mobile platform (e.g., iOS 11), thus incurring a high cost. The user's willingness to pay a high cost to have the latest generation of mobile device and mobile platform signals a high user's interest in the MPIs. In contrast, when the MPI is supported by previous generations of mobile device, the users incur a small cost to use MPIs since they do not have to upgrade their mobile device (e.g., iPhone 7) but only the mobile platform (e.g., iOS 11). Therefore, the early entrant advantage holds if the users show an interest in MPIs. The advantage of early entrants over late entrants increases in a market wherein users have a high interest in the MPIs. Thus, it is critical for MPI-based app innovations to enter early during the growth stage when the MPIs are not compatible with older generations of mobile devices. Therefore, we argue that:

Hypothesis 2 (HQ2): During the growth stage, the earliness of MPI-based app innovations has a greater impact on app performance when the MPIs are not backward compatible.

Methods

We tested our hypotheses in the context of the iOS mobile platform within the U.S. market. We choose the iOS mobile platform over the Android mobile platform because the iOS market is more effective at replacing older generations of the iOS mobile platform with newer generations than the Android market. A close examination at the adoption rate of both the latest iOS and Android mobile platforms in 2017 shows that in September 2018, a year after the release of the mobile platforms, iOS 11 was adopted by 85 percent of compatible devices [45] while Oreo 8 was adopted by less than 15 percent of compatible devices [26]. Since this study is about the performance of app innovations enabled by mobile platform innovations, it is important to study a mobile platform whose market is dominantly operating on the latest generation. Furthermore, The iOS mobile platform ecosystem is hypercompetitive. Market data indicate in the 1st quarter of 2018, more than a thousand new iOS apps were released daily [60]. App innovations supporting the latest mobile platform generation are released at a fast pace. Market data reveal that four months after the release of iOS 11, close to 2, 000 third-party apps that support ARKit ([27, 44]) were released in the App store. Such fast paced and highly competitive market creates several challenges to sustain the performance of app innovations, even for a few weeks.

The focal app innovations are those supporting at least one of the following MPIs: FaceID, ARKit, CoreML, “Hey Siri”, ApplePay, and TouchID. FaceID, CoreML, and ARKit were introduced in iOS 11 released in 2017. “Hey Siri” was introduced in iOS 10 released in 2016 while ApplePay and TouchID were introduced in iOS 8 released in 2014. We choose these six MPIs because they have various levels of backward compatibility (see Online Supplemental Appendix B). Moreover, these six MPIs have been identified as key features by Apple to facilitate third-party development and have attracted considerable attention from iOS users. For example, a month after the release of iOS 11, ARKit-only apps cumulated over 3 million downloads.

We use the description and release notes of iOS third-party apps to identify focal MPI-based app innovations. The identification is based on a keyword search mechanism. We use the following keywords taking into account the letter case: “faceid,” “face id,” “siri,” “coreml,” “core ml,” “touch id,” “touchid,” “applepay,” “apple pay,” “arkit,” and “ar kit.” We ensure this keyword-based approach is reliable by reading random release notes and confirming that these release notes describe MPI-based app innovations. We are able to identify MPI-based app innovations since the release of the MPIs until July 2018. Thus, MPI-based app innovations that support CoreML, ARKit, and FaceID are identified from September 2017 to July 2018, those supporting “Hey Siri” are identified from September 2016 to July 2018, and those supporting ApplePay and TouchID are identified from September 2014 to July 2018

Data

The data are collected from two primary sources: appfigures (www.appfigures.com) and App Annie (www.appannie.com) which are two leading companies in app market data and insights. The app market data provided by appfigures and App Annie have been used in previous studies (e.g., [32, 23, 3, 2, 40]). Appfigures and App Annie have been

accumulating data about iOS apps from Apple iTunes since 2009. Several iOS third-party developers and analysts are extensively relying on the appfigures and App Annie database to have a comprehensive view of the iOS apps market. Specifically, we use appfigures to collect data about app performance. Moreover, we use appfigures to search for iOS apps that support at least one of the following MPIs: FaceID, ARKit, CoreML, “Hey Siri”, ApplePay, and TouchID. Finally, we use App Annie to collect iOS apps’ description and release notes. The dataset contains information on iOS MPI-based app innovations that support FaceID, ARKit, CoreML, “Hey Siri”, ApplePay, and TouchID. We observe the market performance by looking at the iOS apps that appear in the following ranking lists: the top 1,000 free apps, the top 1,000 paid apps, and the top 1,000 grossing apps from the date of the MPI release until July 2018. The position of an iOS app in any of these ranking lists is a clear indication of market performance considering the millions of apps that are currently available on the iOS app market. These ranking lists are usually used to understand apps market performance [32]. The iOS apps that are ranked as top apps are updated regularly. Most highly ranked apps are updated more than once per month. Thus, large periods of observation such as a year or month will not be appropriate to measure app performance. Shorter periods of observation are necessary to identify the performance of iOS MPI-based app innovations. We choose to observe the daily performance of iOS app in order to capture the app performance dynamics in the market over a period. In addition to data on iOS apps ranking, we obtained data about each iOS MPI-based app innovations release timing by examining the app release notes. The final dataset comprises 1,214 iOS MPI-based app innovations of which 548 have no backward compatibility. This dataset represents all MPI-based app innovations for which we can have complete data on key variables of interest. Moreover, these MPI-based app innovations support at least one of the six MPIs (i.e., FaceID, ARKit, CoreML, “Hey Siri”, ApplePay, and TouchID). The number of MPI-based app innovations supporting FaceID, ARKit, CoreML, “Hey Siri”, ApplePay, and TouchID is, respectively, 371, 40, 6, 74, 177, and 545.

Measures

Dependent Variable

We examine the performance of iOS app by observing the variation (delta) of the app ranking following a pre-post design. Apps are ranked based on market performance. App ranking in the top free and top paid charts is based mainly on the number of downloads. Regarding the top grossing chart, app ranking is based mainly on the revenues generated. We calculate the average app ranking before and after the release date of the MPI-based app innovation. Our data contain app ranking 4 weeks before and 4 weeks after the entry (release) of the MPI-based app innovation. The difference between the average post-entry app ranking and the average pre-entry app ranking is used to measure app performance. The average post-entry app ranking is calculated during a period of 21 days after the release timing, and the average pre-entry app ranking is calculated during a period of 7 days before the release timing.²

Independent Variables

Entry timing has been extensively used in the strategic management literature to explain firm performance (e.g., [7, 1, 22, 16, 36]). Following previous studies [1], we assess the

release timing of an MPI-based app innovation by measuring the length of time between the release dates of the MPI-based app innovation and the first mover MPI-based app innovation. Specifically, we count the number of days between the release dates of the MPI-based app innovation and the first mover MPI-based app innovation. The first app among all apps to release an MPI-based feature is identified as first-mover. Moreover, we examine the effect of the MPIs by using a dummy variable backward compatibility. The dummy variable takes a value 1 if the MPI is compatible with previous generations of mobile devices, and a value of 0 if the MPI is only compatible with the same-year generations of mobile devices. ARKit, CoreML, “Hey Siri,” and TouchID have backward compatibility while FaceID and ApplePay do not. Finally, we separate the growth stage from the maturity and decline stage of mobile platform generations. Based on previous studies on the diffusion of innovations (e.g., [53]), we consider the maturity and decline stage to start when the level of adoption of a given mobile platform generation reaches 86–88 percent. After that threshold, the adoption curve becomes flat before going down. We use publicly available data on the adoption rate of iOS and iPhone to estimate the end date of the growth stage, and the start date of the maturity and decline stage (see Online Supplemental Appendix C). We conclude that the latest possible end date of the growth stage is 260 days after the release date of a given mobile platform generation. Moreover, the maturity and decline stage starts right after the growth stage, thus 261 days after the release date of a given mobile platform generation.³

Control Variables

We control for variables that may influence the performance of MPI-based app innovations. Some apps appear in multiple categories. The number of categories can affect the discoverability of the app and, thus, can influence the performance of MPI-based app innovations. We control for this effect by measuring the number of categories per app as of July 2018. We use a time-invariant app-level control since we are unable to observe changes in the number of categories per app over time. Even though we focus on the U.S. market, app discoverability can also be influenced by the number of languages supported by the app and the number of countries that can access the app. For example, apps that support English and additional languages can reach more non-English speakers in the U.S. than apps that only support English. Moreover, an app that is available in multiple countries can have higher discoverability in the U.S. market because of social influence. For example, an individual in the United States may be influenced to use an app if this app is recommended by a friend or a close relative. In some cases, that friend or close relative can be someone outside the United States. We control for these effects by measuring the number of languages supported by each app and the number of countries where the app is accessible as of July 2018. We use time-invariant app-level controls since we are unable to observe changes in the number of languages and countries per app over time. We control for the app quality and usability on the app performance by measuring the number of and the average app ratings. Apps with high quality and usability are used by and can attract more mobile platform users. We also control for the seniority of the app by measuring the number of days between the date of the first release of the app and the date of the release of the focal MPI-based app innovation. Senior apps have an advantage over recent apps because mobile platform users are more likely to be familiar with senior apps than recent apps. Moreover, app seniority has been used to take into account brand strength,

marketing capabilities, and app development superiority [32]. We control for the price effect on MPI-based app innovations performance by measuring the price of the app as of July 2018. We also control the complexity of the app using the size of the code. We use time-invariant app-level controls since we are unable to observe changes in the app price and app complexity over time. Furthermore, we consider different types of ranking whether it is handheld versus tablet, or top free versus top paid versus top grossing. Finally, we use dummy variables to control for the year-specific effects and the app category-specific effects. Table 2 presents the measurement for each variable of the study.

Analysis

We used a pretest-posttest design to examine the effect of a given MPI-based app innovation release timing on app performance. We conducted an ordinary least squares (OLS) regression-described by equation 1- with robust standard errors on the sample. We ensure there is no major concern regarding the OLS assumptions while conducting the analyses (see Online Supplemental Appendix E).

Results

The descriptive statistics and the correlation matrix are reported in Table 3. Moreover, we report the results from OLS in Table 4. The models predict app performance. All the standard errors are robust to correct for heteroscedasticity and autocorrelation. App performance is high when the app rank is low. Thus, a negative coefficient indicates an increase in performance. Model 1 tests the effect of MPI-based app innovation release timing. Model 2 tests the effect of the interaction between the MPI-based app innovation release timing and the level of backward compatibility of MPIs. Model 3 tests the effect of MPI-based app innovation release timing during the stage of growth while model 4 examines the effect of MPI-based app innovation release timing during the stage of maturity and decline. Model 5 tests the effect of the interaction between the MPI-based app innovation release timing and the level of backward compatibility of MPIs during the

Table 2. Variable measurement.

Variable Name	Variable Measurement
App Performance	Difference between post-release timing app ranking and pre-release timing app ranking
MPI-based App Innovation Release Timing	Difference between the release date of the MPI-based app innovation and the release date of the pioneer (in number of days)
Backward Compatibility	Equals 1 if the MPI is supported by mobile devices released before the release date of the MPI, and 0 otherwise
Average App Rating	Average app rating up to the release date of the MPI-based app innovation
Number of App Ratings	Number of app ratings up to the release date of the MPI-based app innovation
App Seniority	Number of days since the app has been available to users
App Size	The size of the code file of the app (in bits)
App Price	The price of the app (in dollars)
Number of App Categories	Number of categories where the app is listed
Geographical App Accessibility	Number of countries where the app is available
Linguistic App Accessibility	Number of languages supported by the app
Type of App Ranking	Equals 1, 2, or 3 if the app appears respectively among top free, top paid, or top grossing apps (dummy variables)
Type of Device	Equals 1, or 2 if the app appears respectively among handheld top apps or tablet top apps (dummy variable)

Table 3. Data descriptive and correlations.

Variable	N	Mean	SD	Min	Max	1
1. App Performance	1,213	-.85	103.51	-725.1	532.25	1.00
2. MPI-based App Innovation Release Timing	1,213	436.13	369.94	0	1,272	-0.05
3. Backward Compatibility	1,213	.55	.50	0	1	0.04
4. Average App Rating	1,213	3.73	1.01	0	5	-0.07*
5. Number of App Ratings	1,213	11051.9	69011.52	0	1.2e+06	0.01
6. App Seniority	1,213	1472.98	894.02	2	3556	-0.01
7. App Size	1,213	8.06e+7	1.04e+8	2.14e+6	1.79e+9	0.07*
8. App Price	1,213	1.89	7.44	0	119.99	-2e-04
9. Number of App Categories	1,213	1.87	.36	1	4	0.06*
10. Geographical App Accessibility	1,213	131.83	53.33	1	155	0.04
11. Linguistic App Accessibility	1,213	7.68	8.16	1	41	0.03
12. Type of App Ranking	1,213	1.76	.87	1	3	0.09*
13. Type of Device	1,213	1.41	.49	1	2	0.04

Variable	2	3	4	5	6	7
2. MPI-based App Innovation Release Timing	1.00					
3. Backward Compatibility	0.30*	1.00				
4. Average App Rating	-0.05	0.10*	1.00			
5. Number of App Ratings	0.11*	0.03	0.08*	1.00		
6. App Seniority	-0.08*	-0.24*	0.04	0.07*	1.00	
7. App Size	-0.01	-0.02	-0.02	0.01	0.04	1.00
8. App Price	-0.06*	0.08*	0.09*	-0.03	0.03	-0.04
9. Number of App Categories	1.1e-03	0.04	0.11*	-0.08*	-0.02	0.07*
10. Geographical App Accessibility	-0.06*	0.08*	0.06	-0.05	-0.05	-0.04
11. Linguistic App Accessibility	-0.04	0.02	0.09*	-0.04	0.02	0.26*
12. Type of App Ranking	-0.06*	0.14*	0.06*	-0.03	0.02	-0.07*
13. Type of Device	-7e-04	0.02	0.03	-0.02	0.01	0.05

Variable	8	9	10	11	12
8. App Price	1.00				
9. Number of App Categories	0.08*	1.00			
10. Geographical App Accessibility	0.11*	0.22*	1.00		
11. Linguistic App Accessibility	0.05	0.19*	0.16*	1.00	
12. Type of App Ranking	0.20*	0.20*	0.31*	0.01	1.00
13. Type of Device	0.03	0.05	0.12*	0.06*	0.07*

* p < 0.05.

stage of growth. Thus, models 3 and 4 are used to test the hypotheses H1a and H1b, respectively. Finally, model 5 is used to test the hypothesis H2.

Models 1 and 2 of Table 4 are provided as exploratory analyses examining the effect of release timing and backward compatibility over both the growth, and the maturity and decline stages. The results indicate that the MPI-based app innovation release timing statistically does not predict app performance. However, when we distinguish between the growth stage and the maturity and decline stage, the results show that MPI-based app innovation release timing is an important predictor of app performance depending on the stage of the adoption curve.

In H1a, we posit that the effect of MPI-based app innovation’s release timing on the app performance is positive during the growth stage. This hypothesis is supported as the coefficient of MPI-based app innovation release timing in Model 3 (See Table 4) is positive and statistically significant (.21, p-value < .01). Thus, during the growth stage, early entrants have a greater impact on app performance than late entrants. An increase of one day in the release timing is associated with a loss of .21 position in the ranking. Moreover, Model 4 of Table 4 indicates that there is no effect between the MPI-based app innovation release timing and app performance during the maturity and decline stage as

Table 4. Results.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	3.4e-03 (1.3e-02)	2.2e-02 (1.8e-02)	.21* (9.7e-02)	-1.2e-02 (2.6e-02)	.39** (.14)
Backward Compatibility		36** (12)			45* (20)
MPI-based App Innovation Release Timing * Backward Compatibility		-3.9e-02* (1.9e-02)			-.43** (.15)
Average App Rating	-8** (3)	-8.1** (3)	-2.9 (4.6)	-10* (4.3)	-3.2 (4.5)
Number of App Ratings	3.3e-05 (3.9e-05)	3.5e-05 (3.9e-05)	3.1e-06 (3.1e-05)	6.7e-05 (4.4e-05)	7.5e-07 (2.9e-05)
App Seniority	-1.0e-03 (3.7e-03)	-1.5e-04 (3.7e-03)	-1.4e-03 (5.9e-03)	-1.9e-03 (4.9e-03)	2.6e-04 (5.9e-03)
App Size	6.4e-08* (2.6e-08)	5.7e-08* (2.8e-08)	6.2e-08 (5.2e-08)	4.7e-08 (3.4e-08)	8.2e-08 (5.4e-08)
App Price	-.13 (.37)	-.15 (.36)	7.1e-02 (.27)	-4.7+ (2.8)	.13 (.27)
Number of App Categories	.71 (10)	.22 (10)	1.2 (14)	9.5 (15)	-.21 (13)
Geographical App Accessibility	-1.4e-02 (6.0e-02)	-2.0e-02 (6.0e-02)	-5.9e-02 (9.1e-02)	-1.5e-03 (8.1e-02)	-5.1e-02 (9.2e-02)
Linguistic App Accessibility	-.34 (.4)	-.38 (.39)	.11 (.63)	-.4 (.53)	3.2e-02 (.64)
Type of App Ranking (Top Paid)	17* (8.8)	16+ (8.8)	34** (12)	19 (16)	33** (12)
Type of App Ranking (Top Grossing)	24** (8.8)	23** (8.8)	35** (12)	22 (14)	33** (12)
Type of Device (Tablet)	6.9 (6.5)	7.4 (6.5)	3.2 (8.3)	12 (10)	4.2 (8.2)
Constant	-6.1 (13)	-35* (17)	-25+ (14)	3.6 (23)	-49* (20)
N	1213	1213	600	610	600
R Squared	0.063	0.069	0.099	0.092	0.112

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001
 Year and category dummies included.

posited in H1b. The coefficient of MPI-based app innovation release timing in Model 4 is statistically non-significant (-1.2e-02, p-value > .1). Thus, during the maturity and decline stage, early entrants do not a statistically greater impact on app performance than late entrants.

In H2, we posit that the effect of MPI-based app innovation’s release timing on the app performance is moderated by the level of backward compatibility of the MPI such that during the growth stage, the effect is positive and stronger for MPI-based app innovations supporting MPIs that are not compatible with previous generations of mobile devices. Based on Model 5 of Table 4, we can conclude that H2 is supported as the coefficient for the interaction term between MPI-based app innovation’s release timing and MPI’s backward compatibility is negative and statistically significant (-.43, p-value < .01). Model 5 of Table 4 indicates that the effect of release timing on app performance is positive (.39) when there is no backward compatibility. Thus, early entrants have a greater impact on app performance than late entrants when the MPIs are not supported by previous generations of mobile devices. An increase of one day in the release timing is associated with a loss of .39 position in the ranking. This effect is reduced when the MPI is backward compatible. Interestingly, the effect release timing on app performance becomes negative (-.04) when there is backward compatibility, indicating that late entrants have

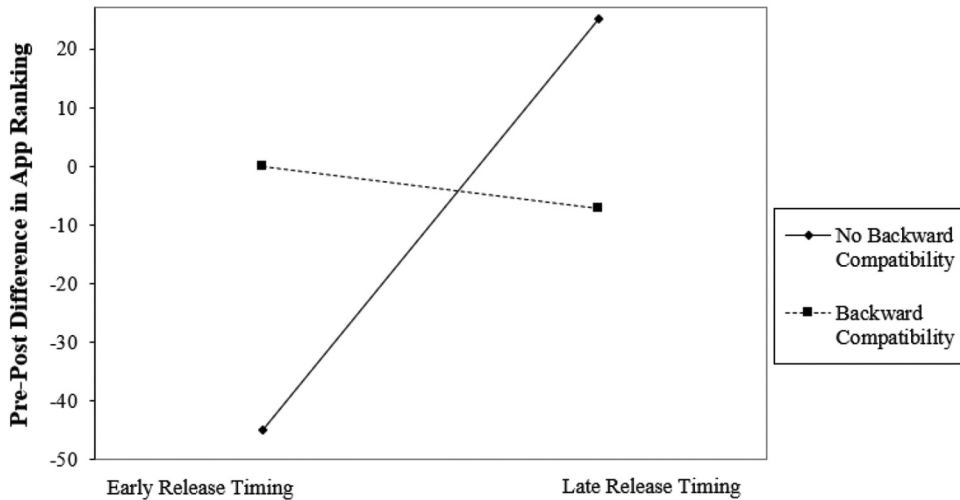


Figure 4. Effect of MPI-based app innovation release timing across different levels of MPI backward compatibility during the growth stage.

a greater impact on app performance than early entrants when the MPI is backward compatible. Figure 4 illustrates the moderation.

Robustness Checks

We conduct several robustness checks to ensure that our main findings are robust to issues such as omitted variables, measurement errors, selection bias, and endogeneity. The variance in app performance can be explained by the novelty of the MPI. The customer interest in a specific MPI might drive its interest in the MPI-based app innovation. For example, if customers are more interested in FaceID compared to “Hey Siri,” we expect MPI-based app innovations that support FaceID to have a greater impact on app performance than MPI-based app innovations supporting “Hey Siri.” Thus, we control for each type of MPI (i.e., ARKit, FaceID, TouchID, “Hey Siri,” ApplePay, and CoreML). The results (See Table 5, for more details see Online Supplemental Appendix E) indicate that during the growth stage, MPI-based app innovations that support ARKit and FaceID have a greater impact on app performance than MPI-based app innovations that support ApplePay. Importantly, the results corroborate the main findings, indicating that our main analyses are robust.

In the next set of analyses, we show that our findings are robust to alternative measures of app performance. Initially, we measure app performance over the period [-7; +21]. Specifically, we adopt a pre-post design wherein we calculate the difference between pre-release timing app ranking and the post-release timing app ranking. We measure the pre-release timing app ranking by averaging the app ranking over 7 days before the release date of the MPI-based app innovation. Moreover, we measure the post-release timing app ranking by averaging the app ranking over 21 days after the release date of the MPI-based app innovation. Using averages allows us to mitigate the volatility of app ranking. We check the robustness of our main findings by using alternative time periods to measure the

Table 5. Results while controlling for the mobile platform innovations.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	2.0e-02 (3.2e-02)	3.8e-02 (3.6e-02)	.22* (.11)	-9.9e-03 (4.7e-02)	.4** (.14)
Backward Compatibility		12 (14)			36 (22)
MPI-based App Innovation Release Timing * Backward Compatibility		-2.1e-02 (2.0e-02)			-.44** (.15)
MPI (ARKit)	59 (41)	57 (42)	-68* (32)	1.3e+02* (53)	-67* (33)
MPI (CoreML)	45 (38)	46 (38)	-1.2 (.27)		-17 (.27)
MPI (FaceID)	16 (38)	23 (38)	-51* (24)	-8.8 (60)	-73** (27)
MPI ("Hey Siri")	48 (32)	48 (32)	-43 (36)	56 (43)	-47 (36)
MPI (TouchID)	-1.7 (8.7)	0 (.)	4.4 (18)	11 (11)	0 (.)
Observations	1213	1213	600	613	600
R Squared	0.071	0.072	0.102	0.108	0.114

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

Table 6. Results using [-3, +7] time period to measure the app performance.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	8.7e-03 (1.6e-02)	3.7e-02+ (2.0e-02)	.35* (.14)	-2.0e-02 (2.9e-02)	.51** (.18)
Backward Compatibility		36* (15)			14 (24)
MPI-based App Innovation Release Timing * Backward Compatibility		-5.1e-02* (2.0e-02)			-.35+ (.18)
Observations	1184	1184	588	596	588
R Squared	0.078	0.082	0.144	0.092	0.151

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

app performance. We use the following time periods: [-3; +7], [-3; +14], [-3; +21], [-3; +28], [-7; +7], [-7; +14], and [-7; +28]. The results (see Table 6, for more details see Online Supplemental Appendix F) support our main findings, showing that our main analyses are robust.

We identify the growth stage, and the maturity and decline stage by using publicly available data about the adoption rate of iOS and iPhones. We conclude that the growth stage starts with the release date of iOS and ends approximately 260 days later. The growth stage is followed by the maturity and decline stage. We conduct several analyses using alternative and conservative measures of the two stages. We consider the growth stage ends earlier using the following time periods: [0, 200], [0, 210], [0, 220], [0, 230], [0, 240], and [0, 250]. Moreover, we consider the maturity and decline stage starts later using the following start date: 270, 280, 290, 300, 310, and 320 days after the release date of iOS. The results (See Table 7, for more details see Online Supplemental Appendix G) are consistent with the main findings indicating that our main analyses are robust.

By deciding the MPIs to consider in this study, our analyses can suffer from selection bias. We selected MPIs that have been identified as major innovations by Apple and

Table 7. Results using alternative time periods for the growth, and maturity and decline stages.

	Model 1: Growth Stage [0, 200]	Model 2: Growth Stage [0, 200]	Model 1: Maturity and Decline Stage [270, -]	Model 2: Maturity and Decline Stage [280, -]
MPI-based App Innovation Release Timing	.37* (.17)	.64* (.31)	-1.8e-03 (3.2e-02)	-3.3e-02 (3.0e-02)
Backward Compatibility		45+ (27)		
MPI-based App Innovation Release Timing * Backward Compatibility		-.48+ (.27)		
Observations	425	425	575	575
R Squared	0.140	0.149	0.085	0.085

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

received a significant consumer’s interest. These MPIs release dates cover the period from 2014 to 2017. To show that our main analyses are robust and do not suffer any selection bias, we reduce the sample size by considering MPIs released in 2017. We limit the analyses to MPI-based app innovations supporting CoreML, ArKit, and FaceID which are major MPIs released in 2017 by Apple. The results (see Table 8, for more details see Online Supplemental Appendix H) corroborate the main findings, indicating that our main analyses are robust and selection bias is not a major concern in our study.

Previous studies tend to consider release timing to be an endogenous variable (e.g., [7]), thus raising the issue of endogeneity in our analyses. Several econometric approaches have been developed to control for endogeneity. Following previous studies [34, 55], we use the Garen two-stage econometric model to take into account any potential endogeneity [18]. The Garen approach extends the Heckman approach to account for selection variables that are continuous [18]. Since release timing is a continuous variable, the Garen approach is suitable to correct for selection bias. Developers may self-select into releasing MPI-based app innovation early or late based on several factors. Some of these factors are observable (e.g., user rating) while others are not (e.g., managerial preferences). In the first stage, we regress the variable release timing on several variables that are likely to influence the timing of MPI-based app innovation. Using the residuals η from the first stage, we calculate the interaction term $\eta \times$ release timing. We include both the residuals and the interaction term in the second stage to correct for endogeneity. The residuals η account for selection bias while the interaction term $\eta \times$ release timing accounts for unobserved heterogeneity over the range of the selection variable. We analyze the first and second

Table 8. Results using a different sample.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	.23+ (.14)	.25+ (.15)	.28+ (.16)	-15 (9.7)	.34* (.17)
Backward Compatibility		33 (38)			56 (46)
MPI-based App Innovation Release Timing * Backward Compatibility		-9.9e-02 (.2)			-.62* (.26)
Observations	417	417	369	48	369
R Squared	0.117	0.118	0.125	0.830	0.140

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

Table 9. Results using Garen approach to control for endogeneity.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release	4.1e-02	8.1e-03	.26*	3.0e-03	.58***
Timing	(4.6e-02)	(4.7e-02)	(.13)	(8.9e-02)	(.16)
η	-8.0e-02	4.6e-03	-5.9e-02	-.22+	-.12
	(6.5e-02)	(6.7e-02)	(8.7e-02)	(.13)	(.1)
MPI-based App Innovation Release	1.6e-05	2.5e-05	3.7e-05	1.6e-04	-7.0e-04+
Timing * η	(4.6e-05)	(4.6e-05)	(3.1e-04)	(9.9e-05)	(3.6e-04)
Backward Compatibility		38**			42+
		(13)			(23)
MPI-based App Innovation Release		-4.2e-02*			-.61***
Timing * Backward Compatibility		(1.9e-02)			(.17)
Observations	1213	1213	600	613	600
R Squared	0.064	0.069	0.099	0.109	0.118

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
 Control variables, year, and category dummies included.

stage models using OLS with robust standard errors to correct for heteroscedasticity. The results (see Table 9, for more details see Online Supplemental Appendix I) show that the correction terms are not statistically significant (p-value above .10). Moreover, the results support our main findings, thus indicating that our main analyses are robust, and endogeneity is not a major concern in our study.

The effect of release timing on app performance can be explained by differences between third-party developers who adopt MPIS and those who do not. In order to alleviate such concern, we conduct a robustness check using a difference-in-difference type approach. We create a control group using propensity score matching. We match the third-party developers adopting MPIS with those who do not adopt based on the app quality (mean app ratings are 3.65 for control group and 3.86 for treatment group at the time of MPI-based app innovation release date) and app ranking (mean app ranks are 312.6 for control group and 312.14 for treatment group at the time of MPI-based app innovation release date). As the third-party developers in the control group do not support MPIS, the value of release timing in the control group corresponds to the release timing of third-party developers in the treatment group. We measure pre-post difference in app rankings and app downloads. The results (see Table 10, for more details see Online Supplemental Appendix J) support our main findings indicating that release timing is critical for app performance especially when MPIS have no backward compatibility. We reach similar conclusion by reducing our sample to free apps (see Table 11, for more details see Online Supplemental Appendix K).

While app ranking provides an accurate representation on app performance, we also test our hypotheses using app downloads as an alternative measure of app performance. We follow previous research [17] to estimate the number of app downloads from the app rank. The results (see Table 12, for more details see Online Supplemental Appendix L) show that MPI-based app innovations released early increase app downloads to a greater extent than those released later. Hence, the results corroborate our main findings.

App price plays an important role in determining app performance. In our main analyses, we ensure that there are no changes of price during our observation period which goes from 7 days before the release of MPI-based app innovation to 28 days after. Moreover, we conduct additional analyses using a reduced sample focusing on apps that

Table 10. Results using matched samples following a difference-in-difference approach,

	Dependent Variable = App Ranking				
	Model 1	Model 2	Model 3	Model 4	Model 5
Release Timing	5.5e-04 (1.3e-02)	-8.5e-02 (.13)	-5.2e-03 (1.7e-02)	-.13 (.14)	-8.8e-02 (.14)
Treatment	6.7 (12)	-24 (21)	-7.4 (17)	-47+ (25)	12 (32)
Release Timing * Treatment	5.3e-03 (1.8e-02)	.3* (.13)	2.0e-03 (2.4e-02)	.41** (.15)	8.2e-02 (.16)
N	2411	1167	1244	847	707
R Squared	0.035	0.054	0.080	0.066	0.092

	Dependent Variable = App Downloads				
	Model 1	Model 2	Model 3	Model 4	Model 5
Release Timing	.54 (.48)	8+ (4.2)	1.1 (.74)	6.6+ (3.9)	12** (4.5)
Treatment	4.5e+02 (4.4e+02)	1.5e+03* (7.6e+02)	9.5e+02 (1.2e+03)	1.8e+03+ (9.2e+02)	3.2e+03+ (1.9e+03)
Release Timing * Treatment	.4 (.99)	-8.8* (4.4)	1.3 (1.7)	-10+ (5.3)	-12 (8)
N	2411	1167	1244	847	707
R Squared	0.058	0.161	0.069	0.214	0.189

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

Table 11. Results using matched samples of free apps following a difference-in-difference approach.

	Dependent Variable = App Ranking				
	Model 1	Model 2	Model 3	Model 4	Model 5
Release Timing	-1.1e-02 (1.6e-02)	-.2 (.15)	-2.4e-02 (2.2e-02)	-.24 (.16)	-.19 (.16)
Treatment	6.8 (14)	-12 (26)	-21 (20)	-34 (28)	26 (49)
Release Timing * Treatment	5.4e-03 (1.9e-02)	.27+ (.15)	9.6e-03 (2.6e-02)	.36* (.18)	.15 (.2)
N	1104	482	622	421	318
R Squared	0.020	0.046	0.048	0.054	0.097

	Dependent Variable = App Downloads				
	Model 1	Model 2	Model 3	Model 4	Model 5
Release Timing	.62 (.62)	9+ (5)	1.5 (.95)	7.4 (4.7)	15** (5.2)
Treatment	4.7e+02 (5.2e+02)	1.7e+03+ (9.2e+02)	1.0e+03 (1.3e+03)	2.1e+03+ (1.1e+03)	3.0e+03 (2.0e+03)
Release Timing * Treatment	.49 (1.2)	-10+ (5.4)	1.3 (1.9)	-13+ (6.7)	-13 (9)
N	1104	482	622	421	318
R Squared	0.068	0.179	0.083	0.232	0.217

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

are free. The results for both app ranks (see Table 13, for more details see Online Supplemental Appendix M) and app downloads (see Table 14, for more details see Online Supplemental Appendix N) indicate that our main findings are robust.

The effect of release timing on app performance might result from the difference of innovativeness of MPI-based app innovation as third-party developers' app innovations are not created equal. Following prior literature [68], we measure the text similarity score

Table 12. Results using app downloads as measure of app performance.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	-1.2 (5.6)	.68 (2.3)	-12* (5.9)	3.7 (9.3)	-13* (5.5)
Backward Compatibility		4.1e+02 (1.5e+03)			-2.5e+03 (2.2e+03)
MPI-based App Innovation Release Timing * Backward Compatibility		3.2 (2.4)			21* (9.8)
N	1060	1060	516	544	516
R Squared	0.071	0.069	0.249	0.121	0.141

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

Table 13. Results using app ranking as measure of app performance with a sample of free apps.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	9.9e-03 (2.8e-02)	3.8e-02 (2.8e-02)	.29+ (.16)	1.0e-02 (7.6e-02)	.34+ (.2)
Backward Compatibility		22 (18)			50 (32)
MPI-based App Innovation Release Timing * Backward Compatibility		-8.3e-02** (3.0e-02)			-.58* (.29)
N	601	601	282	319	282
R Squared	0.187	0.200	0.222	0.332	0.250

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

Table 14. Results using app downloads as measure of app performance with a sample of free apps.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release Timing	-1.5 (8.1)	-1.8 (2.5)	-18* (9.2)	3 (12)	-17* (8.2)
Backward Compatibility		1.9e+03 (3.1e+03)			-3.3e+03 (4.5e+03)
MPI-based App Innovation Release Timing * Backward Compatibility		7.3 (4.9)			47+ (25)
N	601	601	282	319	282
R Squared	0.109	0.170	0.277	0.169	0.283

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

between an MPI-based app innovation and all prior MPI-based app innovations using the app release notes. We incorporate an additional control variable in our model to capture the effect of app innovation differentiation. Since app innovation differentiation is assessed using the text similarity score, high value indicates low app innovation differentiation. The results obtained for both app ranks (see Table 15, for more details see Online Supplemental Appendix O) and app downloads (See Table 16, for more details see Online Supplemental Appendix P) show that our main findings are robust.

The lagged effect of user base measured as the number of app reviews might create endogeneity [39]. In Online Supplemental Appendix Q, we explain why our model is robust to this endogeneity issue. Moreover, we use an instrumental variable estimator following Anderson and Hsiao [4] suggestion and find that the results (see Table 17) support our main findings.

Table 15. Results using app ranking as measure of app performance and controlling for app innovation Differentiation.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release	1.7e-02	5.2e-02*	.23*	-6.7e-03	.34+
Timing	(1.7e-02)	(2.1e-02)	(.11)	(4.1e-02)	(.17)
Backward Compatibility		34*			41+
		(14)			(25)
MPI-based App Innovation Release		-5.7e-02**			-.4*
Timing * Backward Compatibility		(2.2e-02)			(.18)
App Innovation Differentiation	10*	11*	13*	5.5	17*
	(4.3)	(4.3)	(6.7)	(6)	(7)
N	1213	1213	600	613	600
R Squared	0.074	0.076	0.106	0.101	0.110

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.

Control variables, year, and category dummies included.

Table 16. Results using app downloads as measure of app performance and controlling for app innovation differentiation.

	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release	-1.5	1.4	-10*	3.8	-13*
Timing	(5.6)	(2.7)	(4.4)	(9.3)	(5.8)
Backward Compatibility		7.1e+02			-2.6e+03
		(1.6e+03)			(2.3e+03)
MPI-based App Innovation Release		2.7			22*
Timing * Backward Compatibility		(2.6)			(10)
App Innovation Differentiation	3.9e+02	3.3e+02	56	6.5e+02	-1.9e+02
	(3.8e+02)	(4.1e+02)	(1.8e+02)	(5.9e+02)	(2.2e+02)
N	1060	1060	516	544	516
R Squared	0.075	0.072	0.265	0.126	0.143

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.

Control variables, year, and category dummies included.

Table 17. Results with instrumental variable estimator.

	DV = App Ranking				
	Model 1	Model 2	Model 3	Model 4	Model 5
MPI-based App Innovation Release	2.4e-02	3.7e-02	.24*	-4.4e-02	.43**
Timing	(2.1e-02)	(2.3e-02)	(.12)	(3.9e-02)	(.17)
Backward Compatibility		28*			56*
		(14)			(25)
MPI-based App Innovation Release		-3.5e-02			-.51**
Timing * Backward Compatibility		(2.5e-02)			(.18)
Average App Rating	-1.7	-1.4	-30*	14	-29*
	(9.9)	(9.8)	(14)	(14)	(14)
Number of App Ratings	3.9e-04	3.7e-04	1.0e-03	9.5e-04	1.1e-03
	(7.0e-04)	(7.6e-04)	(9.4e-04)	(1.7e-03)	(8.3e-04)
N	1213	1213	600	610	600
R Squared	0.077	0.081	0.116	0.123	0.131

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.

Control variables, year, and category dummies included.

Finally, we conduct time-series analyses over an observation period of 7 days before the MPI-based app innovation release date and 28 after. We limit our sample to free apps. We find that release timing has a curvilinear effect on app performance supporting our hypotheses that the effect of release timing depends on the adoption curve of each

Table 18. Times-series results with a sample of free apps.

	DV = App Ranking			DV = App Downloads		
	Model 1	Model 2	Model 3	Model 1	Model 2	Model 3
MPI-based App Innovation Release Timing	.33*** (8.7e-02)	.28** (9.2e-02)	.3* (.14)	-32** (11)	-23* (11)	-36* (18)
Release Timing * Release Timing	-2.2e-04** (6.9e-05)	-2.3e-04*** (6.9e-05)	-2.6e-04* (1.2e-04)	2.4e-02* (9.7e-03)	2.6e-02** (1.0e-02)	3.8e-02* (1.7e-02)
Backward Compatibility		60* (29)	-54 (36)		-6.0e+03 (4.3e+03)	3.2e+03 (5.0e+03)
Release Timing * Backward Compatibility		-9.7e-02* (4.0e-02)	5.6e-02 (.16)		15* (7.4)	3.1 (20)
Squared Release Timing * Backward Compatibility			3.7e-05 (1.4e-04)			-1.6e-02 (1.7e-02)
Observations	21177	21177	21177	21177	21177	21177

+p < 0.10. *p < 0.05. **p < 0.01. ***p < 0.001.
Control variables, year, and category dummies included.

platform generation. The results are presented in Table 18, and for more details see Online Supplemental Appendix R.

Discussion

Research Contributions

Our research provides several research contributions. First, our study contributes to the body of literature on the performance of third-party apps developed for mobile platform ecosystems such as iOS or Android. The determinants of third-party apps performance have been studied extensively in the literature (e.g., [13, 21, 30, 35, 42, 54, 56-58, 62, 69, 73, 74, 76, 77]). For example, the prior literature suggests the importance of factors such as app price, app category, and app rating. Despite some notable findings in the literature, there is a minimal amount of evidence about the role of MPIs on app performance. Considering that it is challenging to survive in hypercompetitive environments, our research provides a significant contribution to the literature on the performance of third-party apps. We distinguish between MPI-based app innovations that are enabled by MPIs (e.g., ARKit, and CoreML) and those that rely entirely on the developers’ internal resources. By focusing on apps innovations enabled by the MPIs, we unveil the role of the platform provider on third-party apps performance through the release of MPIs. Theoretically, our research highlights the influence of MPIs on the relationship between MPI-based app innovation – app performance. This study is particularly important as platform providers release MPIs on a yearly basis and increasingly third-party developers are supporting these MPIs.

Second, our study contributes to the discussion about the relationship between entry timing and product performance. There is a rich body of literature that examines the link between entry timing and product performance [36]. Prior research about the timing to enter a market provides conflicting empirical findings [61, 25, 7, 29]. For example, some studies found that in the PC industry late entrants tend to have a greater market position and higher survival rates than early entrants (e.g., [65]). While other studies argue that early movers target greater, uncertain revenues opportunities while late movers target

lower, certain revenues opportunities [36]. The latest developments on the relationship between entry timing and product performance posit the firms that enter during the window of time between the emergence of a dominant category and the emergence of a dominant design perform better than firms that enter during any other phase [61]. Our study extends this body of literature by showing theoretically and empirically that the relationship between release timing and product performance for MPI-based app innovations is contingent on the stage of adoption curve. We demonstrate that compared to the maturity and decline stage, the growth stage provides a window of opportunity for MPI-based app innovations. During the growth stage of the adoption curve, early entrants have an advantage over late entrants, leading to greater app performance. However, during the maturity and decline stage of the adoption curve, early entrants lose their advantage over late entrants. These findings imply that future studies should consider the stages of the adoption curve when examining the release timing for service and product innovations.

Finally, we indicate that MPIs play an important role in understanding the release timing for new app services. As third-party developers are building upon mobile platform resources to develop MPI-based app innovations, our research provides evidence that the level of backward compatibility is critical to determine the release timing. We extend previous research on backward compatibility that examines the influence of backward compatibility on the adoption of multigenerational platforms [24, 67]. Backward compatibility affects both the platform and service providers. The size of the market for MPI-based app innovations that supports MPIs with no backward compatibility is smaller, thus encouraging third-party developers to build upon MPIs that are backward compatible. The literature suggests that service providers are better off when the platform provider supports backward compatibility [24]. Our findings extend the prior literature by demonstrating that MPI-based app innovations that support MPIs that are not backward compatible have a positive effect on app performance when they enter the market early during the growth stage. Hence, the release timing when MPIs have no backward compatibility must be early during the growth stage. This study implies that future research should consider backward compatibility when examining the impact of release timing.

Practical Implications

The mobile apps market is hypercompetitive and requires third-party developers to be very active through digital innovations. Our research provides several practical implications about how MPIs are leveraged to achieve greater app performance. First, our study informs third-party developers about the optimal time to enter the market. Our findings suggest that developers should release their MPI-based app innovations early during the growth stage to increase their app performance. The risks of preemption are mitigated during the growth stage compared to the maturity and decline stage. Thus, third-party developers should not wait until the market reaches the maturity and decline stage. We recommend third-party developers to study MPIs ahead of their release date (i.e., beta versions) to understand how they could be used to create new app services. In doing so, third-party developers will be able to enter the market early. When third-party developers are limited in their ability to support multiple MPIs, they should prioritize the MPI that will take less time to support.

Second, our study implies that third-party developers should pay attention to the level of backward compatibility of MPIs when timing their MPI-based app innovations. Whether the MPIs are backward compatible or not has a significant impact on the size of the market MPI-based app innovations can reach. MPIs with backward compatibility are more attractive for third-party developers since their MPI-based app innovations can reach a large market. Our findings imply that release timing is critical when MPIs are not backward compatible. Third-party developers that enter the market late during the growth stage have a disadvantage compared to early entrants. However, this disadvantage is mitigated when the MPI supported is backward compatible. Third-party developers who lack the capability (e.g., experience) to be among the first movers should focus on creating app services enabled by MPIs that are backward compatible. The risk of preemption is less severe in that case. Moreover, third-party developers who can be pioneers should prioritize MPIs that are not backward compatible. By doing so, they differentiate themselves from followers in terms of app performance distinctly.

Finally, our study provides some implications for platform providers who rely on MPIs to support third-party development. Platform providers such as Apple release on a yearly basis new platform services to facilitate the development of third-party apps on their latest mobile devices and mobile platforms. Because these MPIs have different levels of backward compatibility, our study implies that platform providers should be aware of the limitations of certain MPIs (i.e., those with no backward compatibility) to enhance the app performance of third-party developers who cannot be among the first movers. As mobile device prices increase, platform users are more likely to use the same mobile device for a longer period. Thus, there is a growing gap between the levels of adoption of new generations of the mobile device (e.g., iPhone X) and the mobile platform (e.g., iOS 11). This trend should encourage platform providers to develop MPIs that are backward compatible, loosely coupled and less dependent on the hardware. Since the competition on mobile apps market is growing, third-party developers are looking for ways to reach larger market size. Thus, the priority of the platform providers should be to create MPIs that unlock large markets allowing third-party developers to reach more platform users. For example, MPIs that allow third-party apps to be cross-platform and operate on multiple platforms (e.g., iMessage and iOS, Apple Watch, and iPhone) should be the focus of platform providers. As the number of devices (e.g., iPad, iPhone, and iWatch) and mobile platforms (e.g., iOS, WatchOS, and iMessage) continue to increase, platform providers should orient their digital strategies in the creation of MPIs that are independent of the type and generation of mobile device and mobile platform.

Limitations and Future Research

The statistical inferences of this study are subject to a couple of limitations that offer directions for future research. First, the findings are limited to a sample of six MPIs. We observe major MPIs because it is easier to identify MPI-based app innovations that support them. We provide some robustness checks to control for selection bias in the choice of MPIs. Future research can extend our analyses to a large group of MPIs. This will require some innovative techniques to identify MPI-based app innovations that are enabled by these MPIs.

Second, the measure of app performance is limited to 4 weeks before and after the entry of the MPI-based app innovation. Previous research suggests using an aggregate measure of app ranking such as app ranking per month instead of the daily ranking since it is very volatile [32]. Our data have 4 weeks of app ranking allowing us to aggregate the daily app ranking up to one month. Hence, we can observe the short-term impact of MPI-based app innovations release timing. Future research could include the long-term impact of MPI-based app innovations release timing by predicting app performance after two months or more.

Third, our measure of app performance relies solely on Apple top free, paid, and grossing app rankings. Although this measure has been used on the literature (e.g., [38, 32]) and is consistent with our hypotheses, it might not capture appropriately the app performance. The measure does not include the costs related to the innovation. Finally, our study is limited to the iOS platform. We are not sure our results hold in different contexts. Future research can investigate other platform-based ecosystems (e.g., MacOS) to check the generalizability of our findings.

Conclusion

In this study, we investigate how MPIs are leveraged at the app level. Service co-creation is an important aspect of platform-based ecosystems and occurs when third-party developers build upon MPIs to create new app services. In platform-based ecosystems, the impact of MPI-based app innovations depends on their ability to successfully reach the market. We examine the role of MPI-based app innovation release timing on app performance. We test our hypotheses on MPI-based app innovations that support iOS innovations (i.e., FaceID, ARKit, CoreML, Apple Pay, TouchID, and “Hey Siri”) released during the period 2014-2018.

Our findings confirm that the effect of MPI-based app innovation release timing is contingent on the stage of the adoption curve and the level of backward compatibility of the MPI supported. We find that during the growth stage, MPI-based app innovations that enter the market early have a greater effect on app performance than those that enter late. However, during the maturity and decline stage, MPI-based app innovation release timing does not influence app performance. We also found that during the growth stage, the effect of MPI-based app innovation release timing on app performance is reinforced when the MPI supported has no backward compatibility.

The key contribution of this study is to highlight the role of MPI-based app innovation release timing in mobile platform ecosystems that are characterized by arm’s-length relationship between complementors and platform owners. Our study sheds light on the adoption curve of a platform generation as a significant yet under-explored factor to explain complementor performance. Considering (a) that a platform generation adoption curve highlights demand heterogeneity in platform markets wherein early platform adopters are different from late platform adopters, and (b) the platform technologies introduced in new platform generations have a different level of backward compatibility, we demonstrate that the effect of MPI-based app innovation release timing on complementor’s performance is contingent on both platform demand heterogeneity and the platform technology’s level of backward

compatibility. Hence, third-party developers should define their release timing strategy based on the platform generation adoption curve and the level of backward compatibility.

Notes

1. ARKit, SiriKit, and CoreML are technologies developed by Apple. ARKit and CoreML were introduced in 2017 with iOS 11, while SiriKit became part of iOS 6 in 2012. ARKit is a technology that enables augmented reality experiences for app users. SiriKit facilitates the creation of voice commands that can work with the Apple voice assistant Siri. Finally, CoreML allows the creation of machine learning models for apps.
2. We randomly selected some third-party apps from the sample to ensure no other MPI-based app innovations are released during the time period [-7; +21]. Our observations indicate that mainly minor improvements (e.g., bug fixes, performance improvements, or stability improvements) are released. Moreover, we conducted several robustness checks using different time periods. The results are consistent across these time periods (See Robustness Checks Section)
3. No other MPI is released during the growth stage.
4. Adapted from [53]

ORCID

Franck Soh  <http://orcid.org/0000-0002-6131-5861>

References

1. Adner, R.; and Kapoor, R. Value creation in innovation ecosystems: How the structure of technological interdependence affects firm performance in new technology generations. *Strategic Management Journal*, 31, 3 (2010), 306–333.
2. Agarwal, S. *Three essays on interfirm interdependence and firm performance* (Doctoral dissertation, University of Pennsylvania) (2017).
3. Anam, A.I.; and Yeasin, M. Accessibility in smartphone applications: What do we learn from reviews? In: *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, (2013), 35–36.
4. Anderson, T.W.; and Hsiao, C. Estimation of dynamic models with error components. *Journal of the American statistical Association*, 76, 375 (1981), 598–606.
5. Apple. Best of 2017, 2018. Apple. <https://developer.apple.com/app-store/best-of-2017/apps-of-the-year/>(accessed on November 19, 2018).
6. Arora, S.; Ter Hofstede, F.; and Mahajan, V. The implications of offering free versions for the performance of paid mobile apps. *Journal of Marketing*, 81, 6 (2017), 62–78.
7. Bayus, B.L.; and Agarwal, R. The role of pre-entry experience, entry timing, and product technology strategies in explaining firm survival. *Management Science*, 53, 12 (2007), 1887–1902.
8. Bonnie, E. The mobile marketer’s guide to mastering user retention, 2017. *Clevertap*. <https://clevertap.com/blog/guide-to-user-retention/>(accessed on October 3, 2018).
9. Clover, J. Apple now has 1.3 billion active devices worldwide, 2018. *MacRumors*. <https://www.macrumors.com/2018/02/01/apple-now-has-1-3-billion-active-devices-worldwide/>(accessed on October 3, 2018).
10. Constantinides, P.; Henfridsson, O.; and Parker, G.G. Introduction—platforms and infrastructures in the digital age. *Information Systems Research*, 29, 2 (2018), 381–400.

11. Dal Bianco, V.; Myllarniemi, V.; Komssi, M.; and Raatikainen, M. The role of platform boundary resources in software ecosystems: A case study In *Software Architecture (WICSA)*, (2014), 11–20.
12. de Reuver, M.; Sørensen, C.; and Basole, R.C. The digital platform: a research agenda. *Journal of Information Technology*, 33, 2 (2018), 124–135.
13. Dixon, J.; Barkhordari, R.; and Sivanesan, S. Star apps: App originality and interdependence as predictors of app success. In: *23rd American Conference Information Systems (AMCIS)*, (2017), 1–10.
14. Domowitz, I.; Hubbard, R.G.; and Petersen, B.C. Business cycles and the relationship between concentration and price-cost margins. *The Rand Journal of Economics*, 17 (1986), 1–17.
15. Eaton, B.; Elaluf-Calderwood, S.; Sorensen, C.; and Yoo, Y. Distributed tuning of boundary resources: The case of apple's iOS service system. *MIS Quarterly*, 39, 1 (2015), 217–243.
16. Garcia-Sanchez, J.; Mesquita, L.F.; and Vassolo, R.S. What doesn't kill you makes you stronger: The evolution of competition and entry-order advantages in economically turbulent contexts. *Strategic Management Journal*, 35, 13 (2014), 1972–1992.
17. Garg, R.; and Telang, R. Inferring app demand from publicly available data. *MIS Quarterly*, 37, 4 (2013), 1253–1264.
18. Garen, J. The returns to schooling: A selectivity bias approach with a continuous choice variable. *Econometrica*, 52, 5 (1984), 1199–1218.
19. Ghazawneh, A.; and Henfridsson, O. Governing third-party development through platform boundary resources. In *The International Conference of Information Systems*, (2010), 1–18.
20. Ghazawneh, A.; and Henfridsson, O. Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, 23, 2 (2013), 173–192.
21. Goldbach, T.; and Benlian, A. How social capital facilitates clan control on software platforms to enhance app-developers' performance and success. In *36th International Conference of Information Systems (ICIS)*, (2015), 1–18.
22. Gómez, J.; and Maicas, J.P. Do switching costs mediate the relationship between entry timing and performance? *Strategic Management Journal*, 32, 12 (2011), 1251–1269.
23. Goul, M.; Marjanovic, O.; Baxley, S.; and Vizecky, K. Managing the enterprise business intelligence app store: sentiment analysis supported requirements engineering. In *45th Hawaii International Conference of Information Systems (HICSS)*, (2012), 4168–4177.
24. Hann, I.H.; Koh, B.; and Niculescu, M.F. The double-edged sword of backward compatibility: The adoption of multigenerational platforms in the presence of intergenerational services. *Information Systems Research*, 27, 1 (2016), 112–130.
25. Hawk, A.; Pacheco-De-Almeida, G.; and Yeung, B. Fast-mover advantages: Speed capabilities and entry into the emerging submarket of atlantic basin LNG. *Strategic Management Journal*, 34, 13 (2013), 1531–1550.
26. Horwitz, J. iOS 11 hits 85% adoption ahead of iOS 12; android oreo at 14.6%, 2018. *VenutreBeat*. <https://venturebeat.com/2018/09/04/ios-11-hits-85-install-rate-ahead-of-ios-12-android-oreo-at-14-6/>(accessed on October 3, 2018).
27. Horwitz, J. Apptopia: ARKit use by iOS app developers is modest and slowing, 2018. *VenutreBeat*. <https://venturebeat.com/2018/01/03/apptopia-arkit-use-by-ios-app-developers-is-modest-and-slowng/>(accessed on October 3, 2018).
28. Isobe, T.; Makino, S.; and Montgomery, D.B. Resource commitment, entry timing, and market performance of foreign direct investments in emerging economies: The case of Japanese international joint ventures in China. *Academy Management Journal*, 43, 3 (2000), 468–484.
29. Jiang, Z.; Qu, X.S.; and Jain, D.C. Optimal market entry timing for successive generations of technological innovations. *MIS Quarterly*, 43, 3 (2019), 787–806.
30. Kajanan, S.; Ramasubbu, N.; Pervin, N.; Dutta, K.; and Datta, A. Takeoff and sustained success of apps in hypercompetitive mobile platform ecosystems: An empirical analysis. In: *33rd International Conference of Information Systems (ICIS)*, (2012), 1–18.

31. Kankanhalli, A.; Ye, H.; and Teo, H. Comparing potential and actual innovators: An empirical study of mobile data services innovation. *MIS Quarterly*, 39, 3 (2015), 667.
32. Kapoor, R.; and Agarwal, S. Sustaining superior performance in business ecosystems: evidence from application software developers in the iOS and android smartphone ecosystems. *Organization Science*, 28, 3 (2017), 531–551.
33. Kerin, R. A.; Varadarajan, P. R.; and Peterson, R. A. First-mover advantage: A synthesis, conceptual framework, and research propositions. *Journal of Marketing*, 56, 4 (1992), 33–52.
34. Khuntia, J.; Kathuria, A.; Saldanha, T.J.; and Konsynski, B.R. Benefits of IT-enabled flexibilities for foreign versus local firms in emerging economies. *Journal of Management Information Systems*, 36, 3 (2019), 855–892.
35. Kim, K.; and Viswanathan, S. The experts in the crowd: The role of experienced investors in a crowdfunding market. *MIS Quarterly*, 43, 2 (2019), 347–372.
36. Klingebiel, R.; and Joseph, J. Entry timing and innovation strategy in feature phones. *Strategic Management Journal*, 37, 6 (2016), 1002–1020.
37. Lardinois, F. Gmail now has more than 1b monthly active users, 2017. *TechCrunch*. <https://techcrunch.com/2016/02/01/gmail-now-has-more-than-1b-monthly-active-users/> (accessed on November 19, 2018).
38. Lee, G.; and Raghu, T.S. Determinants of mobile apps' success: evidence from the app store market. *Journal of Management Information Systems*, 31, 2 (2014), 133–170.
39. Li, Z.; and Agarwal, A. Platform integration and demand spillovers in complementary markets: Evidence from Facebook's integration of Instagram. *Management Science*, 63, 10 (2017), 3438–3458.
40. Liang, C.; Shi, Z.; and Raghu, T.S. The spillover of spotlight: Platform recommendation in the mobile app market. *Information Systems Research*, 30, 4 (2019), 1296–1318.
41. Linares-Vásquez, M.; Bavota, G.; Bernal-Cárdenas, C.; Di Penta, M.; Oliveto, R.; and Poshyvaryk, D. API change and fault proneness: A threat to the success of android apps. In *Proceedings 2013 9th Joint Meeting on Foundations of Software Engineering*, (2013), 477–487.
42. Liu, C.Z.; Au, Y.A.; and Choi, H.S. Effects of freemium strategy in the mobile app market: An empirical study of google play. *Journal of Management Information Systems*, 31, 3 (2014), 326–354.
43. Liu, C.; Jozani, M.; and Choo, R. Canalization or increased diffusion? An empirical analysis on the impact of the recommendation system in the mobile app market. In *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS)*, (2018), 1432–144.
44. Miller, C. ARKit app downloads said to hit 13m, dominated by games, (2018). *9To5Mac*. <https://9to5mac.com/2018/03/28/arkit-apps-download-stats/> (accessed on October 3, 2018).
45. Owen, M. Adoption of iOS 11 reaches 85 percent ahead of release of iOS 12, (2018). *AppleInsider*. <https://appleinsider.com/articles/18/09/04/adoption-of-ios-11-reaches-85-percent-ahead-of-release-of-ios-12> (accessed on October 3, 2018).
46. Parker, G.; Van Alstyne, M.; and Jiang, X. Platform ecosystems: How developers invert the firm. *MIS Quarterly*, 41, 1 (2017), 255–255.
47. Perala, A. Discover enables face id login for mobile app, 2017. *Findbiometrics*. <https://mobileidworld.com/discover-enables-face-id-login-for-mobile-app/> (accessed on May 20, 2019).
48. Perala, A. HSBC becomes latest bank to embrace face id login, 2018. *Findbiometrics*. <https://findbiometrics.com/hsbc-face-id-login-505101/> (accessed on May 20, 2019).
49. Popper, B. Google announces over 2 billion monthly active devices on Android, 2017. *THE VERGE*. <https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users> (accessed on October 3, 2018).
50. Porter, M.E. *Competitive Strategy*, New York: Free Press, 1985.
51. Qiu, Y.; Gopal, A.; and Hann, I.H. Logic pluralism in mobile platform ecosystems: A study of indie app developers on the iOS app store. *Information Systems Research*, 28, 2 (2017), 225–249.
52. Rietveld, J.; and Eggers, J.P. Demand heterogeneity in platform markets: Implications for complementors. *Organization Science*, 29, 2 (2018), 304–322.

53. Rogers, M.E. *Diffusion of Innovation* (5th ed.), New York: Free Press, 2003.
54. Rollin, R.; Steinmann, S.; Schramm-Klein, H.; Neus, F.; and Nimmermann, F. Drivers of market success for mobile gaming apps-results of a choice-based conjoint experiment. In *38th International Conference of Information Systems (ICIS)*, (2017), 1–20.
55. Saldanha, T.; Mithas, S.; and Krishnan, M. Leveraging customer involvement for fueling innovation: The role of relational and analytical information processing capabilities. *MIS Quarterly*, 41, 1 (2017), 267.
56. Shulman, J.D.; and Geng, X. Does it pay to shroud in-app purchase prices? *Information Systems Research*, 30, 3 (2019), 856–871.
57. Siegfried, N.; Koch, O.; and Benlian, A. Drivers of app installation likelihood—a conjoint analysis of quality signals in mobile ecosystems. In *36th International Conference of Information Systems (ICIS)*, (2015), 1–18.
58. Song, C.; Park, K.; and Kim, B.C. Impact of online reviews on mobile app sales: Open versus closed platform comparison. In: *Pacific Asia Conference of Information Systems (PACIS)*, (2013), 1–11.
59. Statista, Number of newly developed applications/games submitted for release to the iTunes app store from 2012 to 2018, 2018. *Statista*. <https://www.statista.com/statistics/258160/number-of-new-apps-submitted-to-the-itunes-store-per-month/>(accessed on October 3, 2018).
60. Statista, Average number of new iOS app releases per day from 3rd quarter 2016 to 1st quarter 2018, 2018. *Statista*. <https://www.statista.com/statistics/276705/ios-app-releases-worldwide/>(accessed on October 3, 2018).
61. Suarez, F.F.; Grodal, S.; and Gotsopoulos, A. Perfect timing? Dominant category, dominant design, and the window of opportunity for firm entry. *Strategic Management Journal*, 36, 3 (2015), 437–448.
62. Sun, T.; Shi, L.; Viswanathan, S.; and Zheleva, E. Motivating effective mobile app adoptions: Evidence from a large-scale randomized field experiment. *Information Systems Research*, 30, 2 (2019), 523–539.
63. Teece, D.J. Profiting from technological innovation: Implications for integration, collaboration, licensing and public policy. *Research Policy*, 15, 6 (1986), 285–305.
64. Teece, D.J. Profiting from Innovation in the digital economy: Enabling technologies, standards, and licensing models in the wireless world. *Research Policy*, 47, 8 (2018), 1367–1387.
65. Tegarden, L.F.; Hatfield, D.E.; and Echols, A.E. Doomed from the start: What is the value of selecting a future dominant design? *Strategic Management Journal*, 20, 6 (1999), 495–518.
66. Walz, A. Deconstructing the app store rankings formula with a little mad science, 2015. *MOZ*. <https://moz.com/blog/app-store-rankings-formula-deconstructed-in-5-mad-science-experiments/>(accessed on November 19, 2018).
67. Wang, Q.; Chen, Y.; and Xie, J. Survival in markets with network effects: Product compatibility and order-of-entry effects. *Journal of Marketing*, 74, 4 (2010), 1–14.
68. Wang, Q.; Li, B.; and Singh, P.V. Copycats vs. original mobile apps: A machine learning copycat-detection method and empirical analysis. *Information Systems Research*, 29, 2 (2018), 273–291.
69. Wang, Y.; Song, J.; and Aguirre-Urreta, M. An empirical investigation of factors impacting application downloads in mobile app stores. *SIGHCI Proceedings*, (2015), 1–5.
70. Wulf, J.; and Blohm, I. Fostering value creation with digital platforms: A unified theory of the application programming interface design. *Journal of Management Information Systems*, 37, 1 (2020), 251–281.
71. Xue, L.; Song, P.; Rai, A.; Zhang, C.; and Zhao, X. Implications of application programming interfaces for third-party new app development and copycatting. *Production and Operations Management*, 28, 8 (2019), 1887–1902.
72. Ye, H.; and Kankanhalli, A. User service innovation on mobile phone platforms: Investigating impacts of lead userhood, toolkit support, and design autonomy. *MIS Quarterly*, 42, 1 (2018), 165.
73. Ye, H.; and Kankanhalli, A. Value cocreation for service innovation: Examining the relationships between service innovativeness, customer participation, and mobile app performance. *Journal of the Association for Information Systems*, 21, 2 (2020), 294–312.

74. Yin, P.L.; Davis, J.P.; and Muzyrya, Y. Entrepreneurial innovation: Killer apps in the iPhone ecosystem. *American Economic Review*, 104, 5 (2014), 255–259.
75. Yoo, Y.; Henfridsson, O.; and Lyytinen, K. Research Commentary—The new organizing logic of digital innovation: an agenda for information systems research. *Information Systems Research*, 21, 4 (2010), 724–735.
76. Zhao, X.; Tian, J.; and Xue, L. Herding and software adoption: A re-examination based on post-adoption software discontinuance. *Journal of Management Information Systems*, 37, 2 (2020), 484–509.
77. Zheng, J.; Qi, Z.; Dou, Y.; and Tan, Y. How mega is the mega? Exploring the spillover effects of wechat using graphical model. *Information Systems Research*, 30, 4 (2019), 1343–1362.

About the Authors

Franck Soh (f_sohnoume@uncg.edu; corresponding author) is an Assistant Professor of Information Systems at Bryan School of Business and Economics, The University of North Carolina at Greensboro. He received his Ph.D. degree in Information Systems from University of Arkansas. His research interests include business value of IT, competition in mobile platform ecosystems, healthcare IT, customer service performance, open source, and social media. Dr. Soh has published in *Journal of Computer Information Systems* and has presented his research in several conferences and symposiums, including International Conference on Information Systems.

Varun Grover (vgrover@uark.edu) is the David D. Glass Endowed Chair and Distinguished Professor of Information Systems at the Sam M. Walton College of Business, University of Arkansas. His work focuses on the impacts of digitalization on individuals and organizations. He has published extensively in the information systems field, with over 400 publications, 250 of which are in major refereed journals. He has served as senior editor of several major journals, is a recipient of numerous awards and a Fellow of the Association for Information Systems. He has been invited to present numerous keynote addresses at various forums around the world.