Spring 2019

The Effects of Product Feature Complexity, Market Activity, and Update Scheduling on Mobile App Life Cycles

Moonwon Chung

The Effects of Product Feature Complexity, Market Activity, and Update Scheduling on Mobile App Life Cycles

by

Moonwon Chung

Bachelor of Arts
Yonsei University 2010

Master of Science
Seoul National University 2013

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy in

Business Administration

Darla Moore School of Business

University of South Carolina

2019

Accepted by:

Luv Sharma, Major Professor

Manoj K. Malhotra, Major Professor

Yan Dong, Committee Member

Joel Wooten, Committee Member

Cheryl L. Addy, Vice Provost and Dean of the Graduate School

Dedication

I dedicate this dissertation to my family and friends. I would like to thank my beloved wife Jae for always being there for me with undying support all the way. Thank you for the birthday gift you gave me, the Nintendo Switch, without which my disseration would have been completed a year earlier. Thanks to my parents, for being my foundation and all that I am today. Thanks to my father, Chinwha Chung who was the first person in my life to show me how research papers are written during my second year summer in elementary school. Thank you for the constructive feedback and harsh criticism. Thanks to my mother, Insook Lee who constantly challenged me to bring out the best of me by questioning my intellect during my Ph.D. program. Both of you showed me by example, the struggles and what it takes to become an academic. Thanks to my father, mother and sister-in-law, Intak Chang, Eunkyung Shim, and Michelle Chang who had faith in my potential and provided encouragement. Words cannot describe how much I love you all.

I thank Soohoon Park, Taehoon Park, and Sanghoon Cho who joined me in this academic journey at the Darla Moore school of business and provided me with the occasional consultation and alcohol. And finally, thanks to all my peer doctoral students who are almost there, still somewhere, and just beginning. I have enjoyed our wonderful journey together and really learned a lot from you guys.

Acknowledgments

Most of all, my sincerest gratitude goes to Dr. Manoj K. Malhotra, my academic advisor and personal mentor who saw the potential in me. You have become the academic foundation of who I am today. I have seen what it takes to become such a caring advisor through you. And without your support, this research may not have seen the light. I would like to thank Dr. Yan Dong who taught me the ways to conduct rigorous, quality research. I would like to thank Dr. Luv Sharma who showed tremendous leadership and care through my job market and dissertation process. Thank you also for the financial advice. I would like to thank Dr. Joel Wooten who provided valuable feedback for my research and shared precious time for my dissertation as a committee member. Thanks to Dr. Sunny Park who became the parent figure for all the Korean students at the school and provided shelter and food for all of us. Thanks to Drs. Chen Zhou and Sriram Venkataraman who collaborated with me on the mobile money project. I have learned so much from our work together. Thanks to Julia, Cindye, and Valerie for the administrative support, occasional food, and encouragements. I would also like to thank the faculty at the department of Management Science for all their wonderful guidance and work, which continuously inspired me to reach higher. I would like to thank the great staff, Paul, Ben, Nathan, and Bob at the research computing center. Without the HPC center this work would not have been possible. Finally, thanks to the many others not mentioned but shared my joy and grief and helped me put this dissertation together.

Abstract

Rapid advancements in telecommunication devices and the emergence of the mobile app ecosystem have immensely impacted our lives. Innovative apps have helped improve market efficiency in agriculture, contributed to environmental sustainability through peer-to-peer sharing services, and stimulated financial inclusion in developing economies. However, mobile app developers have to deal with challenges that can hinder the app to reach its full potential. In order to achieve commercial success in the hyper-competitive business landscape where freemium business models are dominating, developers need deep understanding on how non-price operational levers such as product design, delivery, and continued service lead to user adoption.

From the two essays that comprise this dissertation, the first essay aims to explain user downloads of free mobile apps during the introduction stage in the lifecycle based on app feature designs and launch timings. The second essay estimates the effect of app enhancement updates on app downloads and explores contextual factors such as update regularity, lifecycle stage, and market activity levels that may further influence the effectiveness of the enhancements. Research questions proposed in the essays are answered by statistical analysis of heteroskedasticity-based instrumental variables regression and difference-in-differences analysis on free iOS mobile game app data acquired from app market Application Programming Interface (API) that contains daily performance observations over a 3.5-year time horizon. Data extraction and sample construction relied on naive Bayes tf-idf document classification algorithms and Bass diffusion model predictions which are performed via multi-thread processing on a high-performance cluster computing (HPC) server.

Our findings suggest that mobile app developers focus on building rich features rather than diversifying features to scale the user base during the introduction stage quickly. Results also show significant interactions between features and market activity whereby market activity-based launch timing strategies are more beneficial for simpler apps. Moreover, our analysis reveals that while an enhancement update is beneficial for an app's performance, its effect can be further reinforced depending on the regularity of update schedules, lifecycle stage of the update, and market activity levels of the time of update in the decline stage. We also find that these effects can significantly increase the lifespan of the app. Future research can investigate whether the findings are generalizable to other app categories and software service contexts.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction and Motivation for the Study

Over the past ten years, we have witnessed a remarkable growth in mobile applications and enjoyed the convenience and economic benefits that these applications have brought to us. These mobile applications have become increasingly important in our lives and businesses by often disrupting existing business models and fostering new business practices. For example, sharing economy apps such as Airbnb and Uber have shown that mobile apps can disrupt well-established industries. Similarly, Apps developed by mobile money service providers such as Safaricom have contributed to the financial inclusion of the under-served population in developing economies, and agriculture apps have helped to resolve the information asymmetry in the crop spot price market. However, most of all, the expansion of the app market and successful transition towards the mobile app ecosystem has been mostly driven by mobile games. Mobile games comprise over 85% of the revenue generated in the mobile app store (Perez, 2013). The most advanced technologies in terms of visuals and features are first experimented with in the gaming category before being disseminated to other app categories, making the gaming category the technology driver in the app market as well. Highly successful mobile games such as Clash of Clans and Angry Birds have changed mobile entertainment for the millennial generation.

Although mobile apps have transformed consumer lives and business practices, developers constantly struggle with challenges that are unique to this market, its

development process, and user characteristics. In order to continue innovations in this fast-paced growing environment, we need a better understanding of how the three facets of market, developer, and users interact and shape the success of mobile applications. Although there has been research that aims to uncover the success factors of mobile apps (Liu, Au, and Choi, 2014; Lee and Raghu, 2014; R. Garg and Telang, 2013; Ghose and Han, 2014), a comprehensive study addressing the unique structure and management strategies over the entire life cycle of mobile apps is critically needed.

The market environment of mobile apps is hyper-competitive with low entry barriers and willingness to pay. Hypercompetition is characterized by intense and rapid competitive moves, in which players constantly search for competitive advantage and erode their competitors' strengths (D'aveni, 2010). It is often cited that the software industry is the most hypercompetitive environment because innovations (S. L. Brown and Eisenhardt, 1997), technological change (Schmalensee, 2000), and turbulence in profitability (Baldwin and Clark, 2000) constantly change. However, the mobile app market within the software industry takes this even further. Average app releases per day can be as high as 4,032 (See Figure 1.1). In addition, because users are hesitant in paying for mobile apps, more and more developers are adopting the freemium business model. Freemium business model refers to a business model that offers the app download for free and extracts revenue from in-app purchases or in-app advertisements. These type of revenue streams accrue slowly over time compared to upfront pricing, and scales proportionally to the size of the user base. Because mobile apps are experience goods which require usage experience for an accurate evaluation of the quality, the freemium business model effectively reduces the burden of trying out the product. Over time, the mobile app revenue generated by freemium apps has grown from 77 % in early 2013 (i.e., $20 billion worldwide) to over 95% by 2018 (i.e., $83.6 billion worldwide) (Taube, 2013). While prior research in the mobile app domain

considers the pricing as an important aspect of understanding the market structure, the prevalence of freemium apps diminishes the role of pricing. As such, there is a need for more research that looks into non-price-based competition factors.

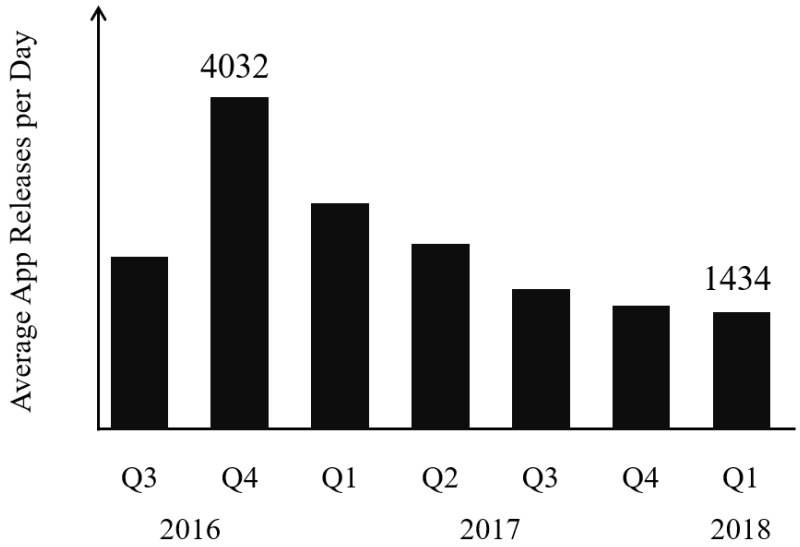
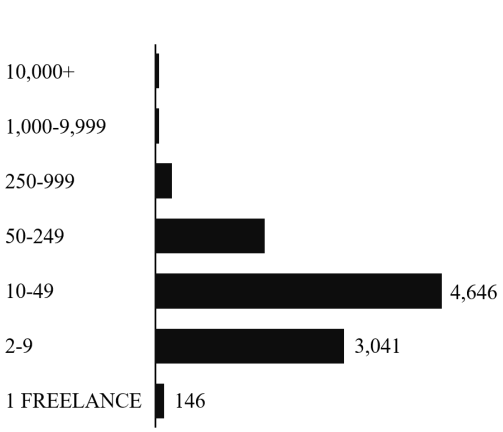


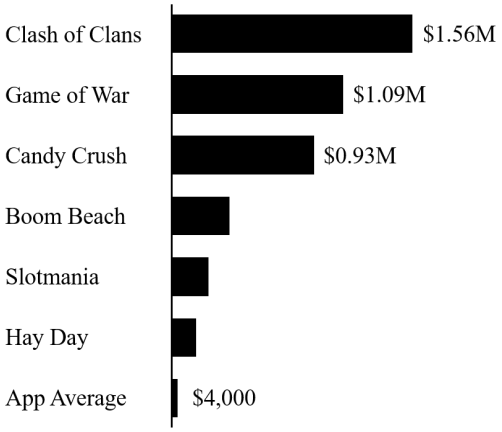
Figure 1.1: iOS App Store: App Releases per Day (Source: Localytics)

The developers of mobile apps are mostly small firms with limited resources (Panko, 2018). As shown in Figure 1.2(a), over 50% of the firms employ less than 50 developers, and a significant number of firms employ less than ten employees (Clutch, 2018). Although there are a large number of app developers, only a small number of successful firms are capable of generating large revenue streams. For example, very successful apps such as Clash of Clans generate over $1.5 million per day, whereas an average app is only able to generate $4,000 per day (See in Figure 1.2(b)). This shows the winner-takes-all nature of the mobile app marketplace competition. Mobile app developers need to shorten time-to-market, while also minimizing development cost to stay relevant in this competition.

At the same time users in the app market are becoming increasingly demanding, with apps showing high churn rates. According to a survey by a mobile app market analytics firm AppDynamics, over 56% of the responders believe that their expec-



(a) Developer Size (Source: Clutch.io)



(b) Daily Revenue of Top Ranked Apps (Source: Statista)

Figure 1.2: Mobile App Developer Profiles

tations of app performance are increasing over time, but do not want to deal with overcomplicated features in the app (Brauer, 2014). As a result, mobile app developers are having difficulties in keeping these increasingly demanding users engaged in the app. The retention rate of mobile apps falls below 50% just 3-4 months after the launch. Mobile games lose 50% of their user base just within two months since launch (See Figure 1.3) (Gordon, 2018).

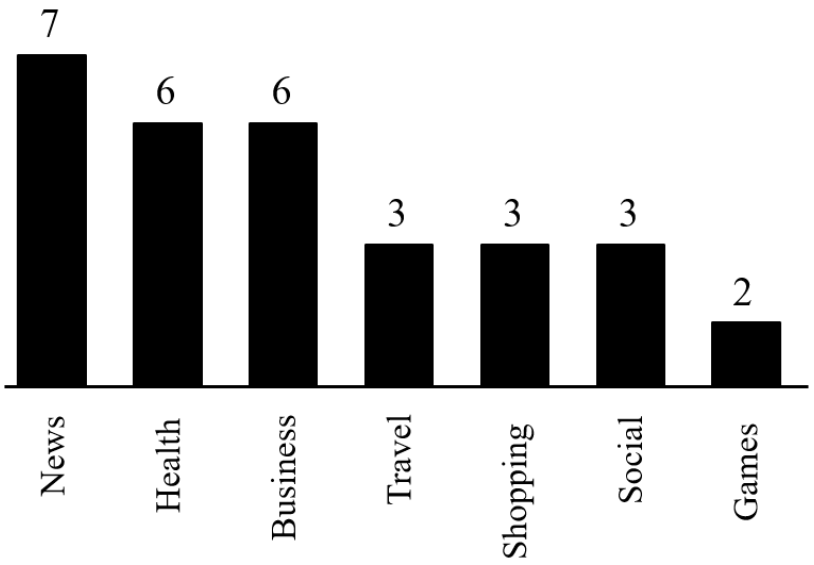


Figure 1.3: Mobile App Median Half Life in Months (Source: Flurry Analytics)

Taking all these factors into account, the resulting app life cycle shows a unique condensed pattern. Mobile app users accumulate instantaneously in the early stage of the life cycle, followed by a constant decay in adoption. Compared to apps that are short-lived, apps that have a longer life cycle show a striking difference in the early stage user base size (See Figure 1.4). The early stage user base size is important for app developers as it facilitates network effects that further enhance the quality of the app through user generated content and its subsequent revenue generation potential from in-app purchases and advertisements.

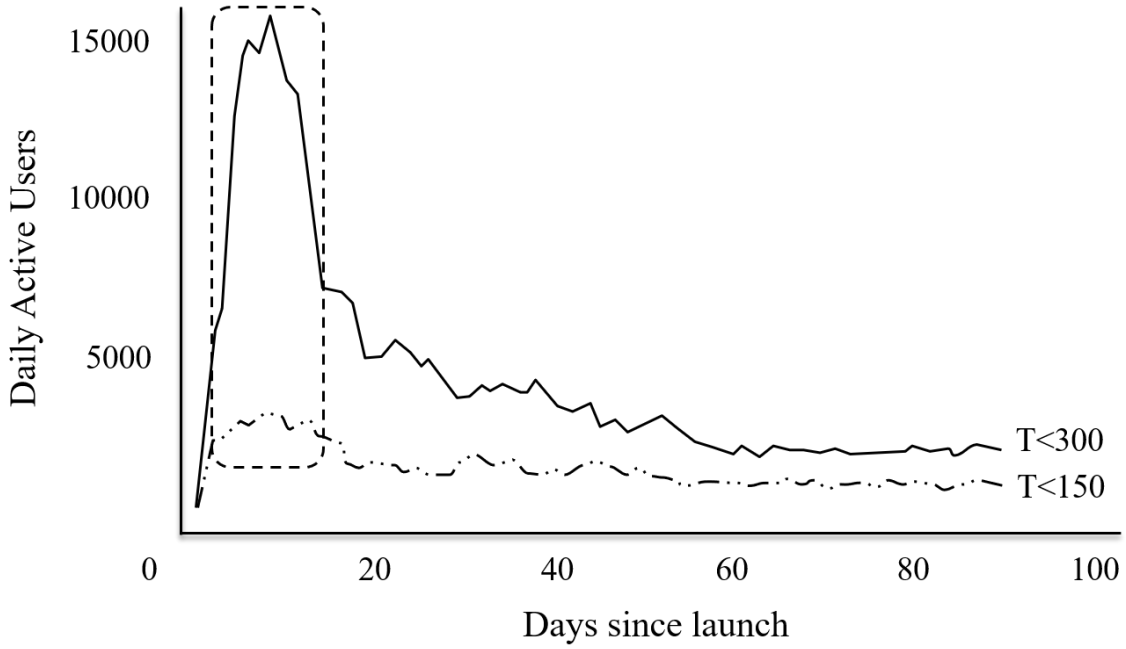


Figure 1.4: Average Daily Active Users Since Launch

To cope with the unique challenges that developers face in the mobile app market, the use of freemium business models is natural. In order to reduce risk, developers need to also bring down the cost and time-to-market of their development projects âĂŞ this is generally done through agile development (See Figure 1.5). One key component of agile development is the use of code generators and software packages, also known as software development kits (SDKs). SDKs are mostly open source developed modularized code libraries that conveniently add certain features to the mobile app

under development. Additionally, developers conduct various forms of beta testing and pre-market sensing activities before the app launch. In the final operations phase, the developers make the final decision on when to deploy the app in the market. Even after the app is launched, there is a significant amount of content that is held back into post-launch updates. These updates not only include responsive bug fixes and patches, but also new content and events held in the app to further stimulate the download and engagement of the users over the app's lifetime. The introduction of new content and addressing issues within the app through bug fixes and patches are all referred to as software maintenance practices. Although app developers are not introducing a new product into the market, substantial content updates for a preexisting mobile app involve significant investments and careful planning as such.

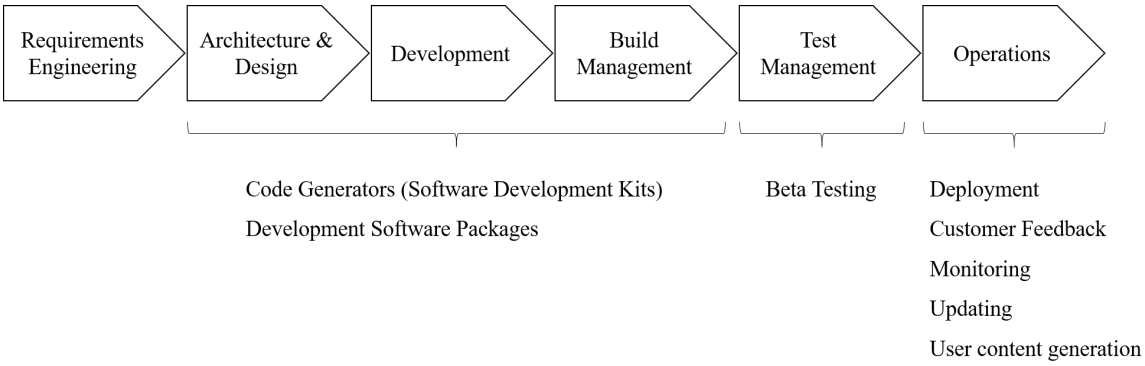


Figure 1.5: Software Development Lifecycle (adapted from (Tuzin et al., 2019))

Although mobile applications have received much research attention recently, no research study has taken a look at the big picture containing the interactions between the market, user, and developer strategies (See Figure 1.6). Also along the continuum of the software development life cycle, there are missing links in our understanding on how to effectively design the app features and maximize app market potential through a carefully planned entry strategy. While prior research in the context of mobile apps has investigated the implications of portfolio management (Lee and Raghu, 2014), demand prediction (R. Garg and Telang, 2013; Ghose and Han, 2014), and customer

feedback (Liu, Au, and Choi, 2014), little is known about the impact of design and development, deployment timings, and app maintenance strategies. To this end, we present the following questions to form the basis for the research contained in this dissertation.

1. What are the impacts of feature design on the early stage user base expansion of a mobile app?
2. How does the launch timing of the app affect the customers' perceptions toward app feature design?
3. Why are some enhancement updates more effective than others?
4. How should mobile app developers schedule their enhancement updates?

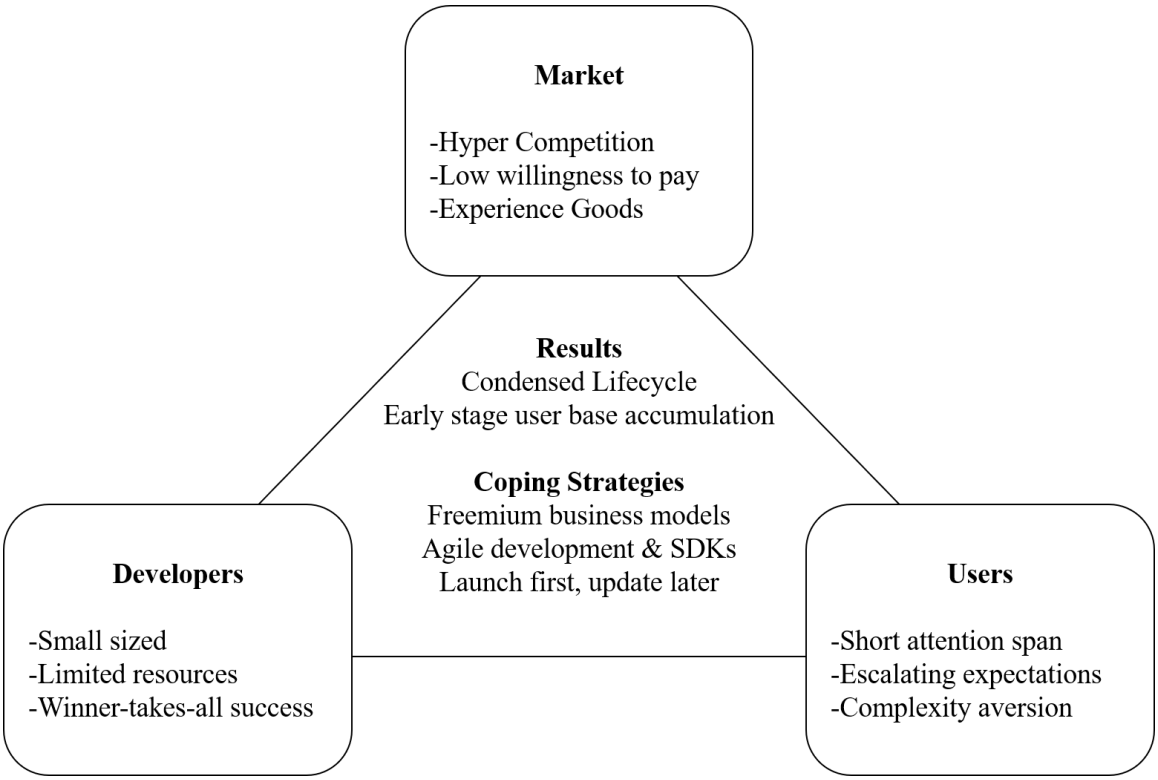


Figure 1.6: Conceptual Framework

1.2 Research Objectives & Model

The objectives of this dissertation are three folds. First, by investigating the relationship between app feature richness and diversity decisions, and the early stage user base size, we can understand how the complexity of the app feature design affects the user's experience and perception of the app's quality. Second, by examining the impact of the market activity level at the time of app launch, we can learn about how a market activity-based product launch timing strategy can further enhance the performance of the application. Moreover, the interactions between the feature complexity and market activity levels can show how decisions about each dimension influences one another. Third, by estimating the app's performance improvement through each content update under various circumstances, we can understand how app characteristics and market characteristics alter the effectiveness of those updates. Findings may provide crucial input to developers by aiding the formulation of software maintenance strategies.

To answer the research questions and achieve the objectives put forth in this dissertation, a novel dataset was extracted from a proprietary applications programming interface (API) server that contains detailed information on app characteristics, developer characteristics, and market performance metrics. A combination of econometrics, machine learning, and big data analytics was deployed to analyze the data. To enhance the causal inference of the models, endogeneity concerns were accordingly addressed, and as shown in the Appendices, various checks were developed to demonstrate the robustness of the findings.

1.3 Dissertation Outline

This dissertation is arranged as follows. Chapter 2 presents the empirical investigation regarding the implications of key operations decisions on product design and market

entry timing on the early stage user base size of free mobile applications. Chapter 3 studies the subsequent stage in the app's life cycle and the implications of software maintenance practices, which are also known as updates. Both chapters 2 and 3 contain a survey of the relevant literature, and discussions about the data, empirical model, and the main findings. Finally, Chapter 4 states the main conclusion and suggestions for future research based on the investigations made in this dissertation.

Chapter 2

Product Design and Launch Strategies for Free Mobile Applications

Abstract

Nowadays, mobile apps are converging towards the freemium business model which shows extremely short lifecycles with almost instantaneous demand saturation. Because most of the freemium apps rely on user content generation and network effects to enhance quality, we focus on the early stage user base size at the demand saturation point as a key indicator of app success. Instead of the four-stage product lifecycle management and pricing strategies which are becoming irrelevant, we focus on pre-launch decisions such as product feature choice and market launch timing and estimate their impact on the early stage user base size. The dataset is extracted from a proprietary application programming interface (API) which contains daily app performance and file structure information of 1,782 free iOS gaming apps in the U.S. over 3.5 years. We propose a framework of categorizing product feature choice as a decision of the number of features (feature richness) and the different types of features (feature diversity) in the app, and a market launch timing strategy based on market seasonal downloading activity at the time of launch. Findings show significant positive effects of feature richness and adverse effects of feature diversity on launch success. This feature choice also interacts with market activity which indicates significant preference shifts between seasonal and non-seasonal users. Post-hoc analyses reveal monetization-quality trade-off and developer learning effects in feature choices.

Our study provides both academic as well as managerial insights on how product architectures impact the effectiveness of market entry timing strategies.

Keywords: Mobile apps, software development kits, seasonality, organizational learning

2.1 Introduction

Increased penetration of smartphones, combined with higher mobile bandwidth and cellular microprocessor capabilities, is impacting our lives as mobile apps conveniently organize our daily activities (e.g., fitness, travel, shopping, financial management) and entertainment (e.g., gaming, social media, multimedia). As a result, consumers are spending more time on their mobile devices. Comparing the daily time spent amongst US millennials from 2012 to 2017, statistics show an increase from 107 minutes to 223 minutes (Statista, 2018b). Relatedly, mobile app markets are exponentially growing, with iOS apps alone having accumulated over 170 billion downloads and consumer-spending of $130 billion between July 2010 and December 2017 (Cheney, 2018).

The two commonly used business models in the mobile app sector are the paid and freemium models. In the paid model, users pay an upfront price before downloading the mobile app. On the other hand, in a freemium model, the mobile app download is free, and the user pays for optional value-added services and features. The freemium business model helps user-base expansion by reducing user-entry cost. The most notable example of a successful freemium product is Angry Birds 2, which accumulated a large user base in a very short period (Grundberg, 2012). Over time the preferred business model has gravitated towards freemium apps, which currently account for over 90% of the revenue generated from apps listed on the iOS and Android platforms (Perez, 2013). The success of freemium apps relies on user generation of content to increase engagement and improve user experience, both of which in turn can influence in-app advertisements and merchandise sales, the two primary sources of revenue

for freemium apps. Reliance on user generation of content requires freemium apps to quickly achieve a critical mass of users. The free nature of the app, along with the importance of user network effects in the success of freemium apps, makes this app segment hypercompetitive with condensed app lifecycles and the winner-takes-all nature of the competition.

As a result of these trends, we see a product lifecycle for the freemium mobile apps which differs significantly from traditional product life cycles (Downes and Nunes, 2014). This segment has a compressed product lifecycle with exponential growth in the initial stages, followed by gradual decay. In most apps, this exponential growth and peak number of users is achieved within a few days to weeks of the app launch. Achieving exponential growth introduces unique operational challenges for mobile app developers, most of whom have limited resources (Panko, 2018). Due to the importance of mobile app launch and initial user growth in determining success, we focus on operational and market-level factors that influence the initial success of freemium gaming apps. Ramachandran and V. Krishnan, 2008 identify product design, launch timing, and pricing as critical factors in determining the commercial success of technology products in an industrial setting.

Prior work on mobile apps has not looked at the challenges associated with product development and factors contributing to launch success. Specifically, in the mobile app context, previous academic research has uncovered the linkage between user ratings (Liu, Au, and Choi, 2014), portfolio strategies (Lee and Raghu, 2014), and pricing (R. Garg and Telang, 2013; Ghose and Han, 2014) on app performance. Much of this research focuses on price-based competition models which fall short in explaining the heterogeneous in-app user base of free apps, its unique lifecycle characteristics, and factors that contribute to their initial success. Since pricing is not a consideration for freemium apps, as per Ramachandran and V. Krishnan, 2008, we focus this paper on studying the impact of feature design and launch timing only

on app launch success.

Nowadays, developers are increasingly relying on modularized software development kits (SDKs) to add features to their apps. SDKs are open-source developed and represent modularized, pre-tested, reusable codes that enable firms to add specific app features with low cost (Atreyi, Ye, and Teo, 2015; Dalmasso et al., 2013). This provides a huge benefit to developers because the majority of them are small-sized businesses with limited resources (Panko, 2018). However, industry reports are recently expressing concerns that there might be too many SDKs installed in a single app, which in turn may undermine the app's performance (Shoavi, 2017). Even though few guidelines are available, addressing the management of feature complexity with increased use of SDKs has become critically important. To do so, we first estimate the impact of the richness and diversity of feature complexity on the app's performance by using the apps' SDK composition information that is identified from reverse-engineered file structures. Here, *feature richness* is defined as the number of SDKs installed (depth or intensity of available features) in the app, whereas *feature diversity* refers to the number of distinct features (breadth or heterogeneity) available in the app.

Identifying the optimal product launch time is known to be an essential operational decision, especially for new products with relatively short life cycles (August, Dao, and Shin, 2015; Calantone et al., 2010). Industry reports suggest that significant seasonal boosts in market demand depend on the month or weekday the app was launched (Datta and Kajanan, 2013). Apps that carefully plan their launch timings can benefit from lower user-acquisition costs and reach peak downloads more efficiently. Therefore, we suggest market activity as another critical factor to consider when determining product launch times and exploit advancements in app market intelligence that provides information on daily sales of all products in the market. In our study, we define market activity as the seasonality component of the subcategory

market demand trends at the time of app launch (Hodrick and Prescott, 1997).

Though this paper, we aim to answer the following research questions: *"what are the individual and joint impacts of feature diversity and richness on the early-stage-user-base expansion of a mobile app?"* and *"how do seasonal and non-seasonal customers perceive app feature diversity and richness?"* To answer these research questions, we formulate an instrumental variables regression model and estimate the effect of feature complexity (richness and diversity), launch timing, and their interaction effects on the app's performance. The initial success of mobile apps is gauged by using the magnitude of the initial peak in number of daily active users. This measure counts the number of users that accessed the app at least once on a given day, making it a good indicator of the user base size. The empirical dataset is extracted from a proprietary application programming interface (API) server which contains daily panel observations of 1,782 free mobile gaming iOS apps in the U.S. over a period of 3.5 years. In this paper, we focus in particular on mobile gaming apps because they provide us with a great setting for our study. Research characterizes the mobile games as possessing the highest degree of competition intensity, thus showing the shortest average lifespan (Gordon, 2018) among all app categories. Consequently, developers in this category continuously experiment with novel development processes and adopt cutting-edge technologies to expand the user base and prolong app lifespan. This is evident in the average app development cost (Dogtiev, 2018) and the average number of SDKs embedded in gaming apps - the highest across all app categories. Second, gaming apps are the revenue-driving category for the entire market, and there is a high degree of heterogeneity in performance because of the winner-takes-all nature of this industry. The mobile gaming revenue, which consists 31% of the users' mobile device spending (Newzoo, 2016) , is anticipated to grow from $36.5 billion in 2016 to $74.5 by 2020 (Statista, 2018a). A small number of well-established apps are taking the lion's share of the generated revenue. For example, a popular app like Clash of

Clans makes over $2.4 million per day while an average app earns $4,000 (Strauss, 2013).

Our results show that a unit increase in feature richness leads to a 6.8% increase in daily active users at the introduction stage, while a unit increase in feature diversity leads to a 23.1% decrease. We also find significant interactions between feature richness and diversity, whereby achieving high feature richness and diversity simultaneously can mitigate the negative effect of feature diversity. We call this the enhancing effect of feature richness. Furthermore, we observe significant interactions between feature diversity and market activity, which suggests that apps with less feature diversity benefit more from market activity-based launch decisions. In order to identify feature diversity mitigation strategies, we conduct post-hoc analyses to better understand the drivers behind the negative impact of feature diversity. Results of the post-hoc analyses reveal that monetization related features drive the negative impact of feature diversity, while publisher experience attenuates feature diversity's negative impact on launch success. This result confirms a trade-off relationship between user experience quality and developer monetization features. Findings call for a careful balance between these two opposing forces. Finally, our result suggests that adding more feature categories should be a strategy that should be pursued only by more experienced publishers.

The remainder of this paper is organized as follows. We first survey the relevant literature and develop our hypotheses in Section 2.2. We describe the data and variables in Section 3 and formulate the econometric model in Section 4. Analysis results are reported in Section 5. Finally, we discuss the theoretical and managerial implications of our findings followed by limitations of this study in Section 6.

2.2 Literature Review

In this section, we review several closely related papers on product design and launch timing strategies that are relevant to the context of freemium mobile apps. We conducted an extensive review of papers in operations management, marketing, and information systems to identify gaps in the literature.

2.2.1 Mobile app design and feature complexity

Mobile app developers face challenges that are different from other product design and development settings. Most of these app developers are small start-up companies with less than 50 employees (Clutch, 2018). Due to these constraints, app development budgets are tight (generally below $250,000) with development time less than three months to ensure quick time-to-market. In this short time, developers are pressed to develop apps compatible across multiple platforms in order to maximize the app's market reach and add advanced features which may require interactions with other apps or sensory modules (e.g., accelerometer, GPS, microphone, cameras) installed in the device. Given these constraints, app developers rely on two strategies to optimize their effort and return on investment: 1) they use software development kits (SDKs) - programming packages or collections of software code libraries that add various features to apps, to quickly add desired features (refer to Table 2.1 for additional details on SDKs) and 2) they generally launch an incomplete product in the market and source content and response from users. Further features are developed and added based on the initial market response for the app.

Given the criticality of user-generated content on the app's commercial success, developers face an important decision on the feature set to include in the launched app that will be most effective in increasing the user base. In the extant literature, features are used synonymously with complexity and refer to the number of components and degree of technical novelty required for a product system (Wang and Tunzelmann,

Table 2.1: List of App Feature Categories and Example SDKs

Features	**Description**	**Example SDK**
Analytics	SDKs that measure and report standard in-app metrics such as pageviews, bounce rate, churn rate, UX data.	Firebase, Flurry, Mixpanel, Localytics, Amplitude, Appsee
Crash Reporting	SDKs that provide comprehensive crash reports, real-time processing, and alerts, smart charts on crash trends and insights	Crashlytics, Firebase Crash Reporting, HockeyApp, Bugly
Development Tools	SDKs that provide the operating system platform for mobile app development	iOS SDK, Android SDK, Unreal SDK, Unity
Multiplayer Platform	SDKs that provide multiplayer support and progress tracking	Photon, GameCenter, Mobage
Social Sharing	SDKs that provide SNS connection for sharing content	Facebook, Twitter, APShareKit, Instagram
Messaging	SDKs that provide user chatting features in the app	Hyphenate, JivoChat, 360Dialog
Testing/Beta management	SDKs that allows internal beta tests or through a selective set of users and generate metrics and reports	Testfairy, Lookback, Testflight, Leantesting,
Attribution	SDKs that assesses user events such as downloads/installs/in-app purchases and relate to acquisition channels	Appsflyer, Kochava, Adjust, Tune, Branch
Datahubs	SDKs that deal with database connections and provide convenient app data management interfaces	Charito, Fivetran, Segment, mparticle
Engagement	Marketing Automation & Push Notification	OneSignal, Branch, Urban Airship, Localytics, Braze
App store intelligence	SDKs that provide information about how the app is performing in the app store as well and mapping the app store, business, or marketing ecosystem in the market	App Annie, Sensor Tower, Apptweak,
Monetization/ Advertising	SDKs that provide additional means of monetization through in-app advertisements, marketplace deals, and cross-promotion	Google Mobile Ads, Facebook, Chartboost, AppLovin, MoPub
Acquisition/ Re-Targeting	SDKs that provide means for user re-engagement such as referral and loyalty programs	Applift, Appvirality
Payment	SDKs that provide apps with an easy payment solution to help apps process payments	OpenIAB, Card.io, Skubit, PayPal, AndroidPay
Location	SDKs that allow apps to track users' location and use geolocation as a tool for mobile marketing, push notifications, monetization, and provide relevant information to users	Factual, Geomoby, Radar, Skyhook, Reveal mobile

Source: Appsee, 2018; SafeDK, 2018

2000). It is known that complexity consists of multiple dimensions such as depth and breadth of the product and service features (Benedettini and Neely, 2012; Jacobs and Swink, 2011; Meyer and Curley, 1991). This classification of complexity based on depth and breadth of features fits well for entertainment products as well.

In the context of mobile apps, we define the depth and breadth dimensions of product complexity as feature richness and feature diversity. Feature richness is measured by the total number of SDKs of an app, whereas feature diversity is measured by the number of distinct feature categories in the app. This is a similar adaptation of the NK-type product complexity measurement (Vickery et al., 2016), where N represents the number of components, and K represents the coupling or dependence between the components (Kauffman and Weinberger, 1989). Consider the social sharing feature in mobile applications as an example. While the addition of a single Facebook SDK enables the social sharing feature in an app, the developer can choose to increase feature richness by installing additional SDKs that pertain to the social sharing feature such as Twitter and Instagram. Conversely, if the developer chooses to add Geomoby, which adds geographical location-based services in the app, the number of distinct features and feature diversity in the app increases. For users, increasing feature richness by installing multiple SDKs in a single feature category can enhance the capabilities of that feature by broadening the selection for users or by enhancing the aesthetics of the interface (Rozendaal et al., 2009). On the other hand, increasing feature diversity may potentially make users feel uncertain about their choices related to the product use (Schwartz, 2000), and overflow of information presented from the product can distract users (Rozendaal et al., 2009).

Complexity has been studied extensively in operations management (OM) literature in various contexts and level of analysis (Jacobs and Swink, 2011). In this study, we focus on complexity at the product-level which is relatively sparse. In order to demonstrate the unique contributions of our study, we conducted an extensive re-

view of this literature. A summary of prominent studies on product complexity is presented in Table 2.2.

As is evident from Table 2.2, OM research on product complexity has primarily focused on organizational and supply side challenges associated with managing complexity (Jacobs and Swink, 2011; Gokpinar, Hopp, and Iravani, 2010; Kreye, Roehrich, and Lewis, 2015). Managing product complexity is important as it can have negative effects to a firm's operational performance such as order/unit fill rates (Closs, Nyaga, and Voss, 2010), supply chain performance (Kaski and Heikkila, 2002), and product quality (Gokpinar, Hopp, and Iravani, 2010). Even OM practices designed to reduce product complexity such as modular designs can be less effective if the inherent product complexity is too high (Vickery et al., 2016). Although this stream of literature provides important insights, it does not directly apply to our context of freemium mobile apps because the convenience of SDKs makes operational challenges in adding features less relevant. Further, even if the addition of features increase the complexity of the product and the associated developmental challenges, it is important to view features from a user perspective because they are the key influencers of user perception (Griffith, 1999) and product quality (Carpenter, Glazer, and Nakamoto, 1994). These perceptions, in turn, affects the early stage user base expansion, which determines network effects and user content generation capacities.

In this light, few service operations and marketing studies have considered market response as a function of complexity. Complex services are known to require more cognitive capacity from the customers which can lead to decreased customer satisfaction and loyalty intentions (Mikolon et al., 2015). Marketing research shows that quality evaluation is more difficult for complex products and therefore, suggests that managers focus on building stronger brand images to reinforce indirect quality cues (Hutton, 1997; Kim and Hyun, 2011). A particularly relevant study by Thompson, Hamilton, and Rust, 2005 adapts the technology acceptance framework Venkatesh

et al., 2003 and shows that the number of features of a product can impact both the usefulness and ease of the use of the product. They find an inverted U-shape impact of the number of features on the product's performance which suggests that consumers can feel fatigue from an excessive number of features. These studies focus on assessing a singular construct of product/service complexity which implicitly assumes a homogenous effect on performance. While the findings provide theoretical and managerial insights on how consumers form intentions regarding product features and why it is important to build brand images and sales representative support, they do not offer an actionable framework for producers to determine product feature designs, especially for situations with heterogeneous feature choices with differential impacts on end users.

Information Systems (IS) literature has also addressed product complexity in the context of software development. Banker, Davis, and Sandra A Slaughter, 1998 state that complex software systems require more effort in maintenance and updates. Therefore, the degree of complexity increases the required effort for making software enhancements. J Alberto Espinosa et al., 2007 determine that the benefits of software development task familiarity decrease as the structure of the software becomes more complicated. These studies fall short in assessing software performance at the feature level and do not examine the linkage between feature complexity and software market success. Furthermore, IS research in the context of mobile apps has uncovered the linkage between user ratings (Liu, Au, and Choi, 2014), portfolio strategies (Lee and Raghu, 2014), and pricing (R. Garg and Telang, 2013; Ghose and Han, 2014) on app performance. However, most of these studies focus on price-based competition models and do not address the unique lifecycle characteristics to identify key success factors for freemium apps. Currently, to the best of our knowledge, there are no existing studies that explore the impact of feature complexity on the initial success in the unique context of freemium apps. As argued earlier, initial success is critical

to the longevity and profitability of freemium apps.

This study attempts to overcome these limitations and contribute to research in this area by first analyzing demand-side market performance measures (such as number of daily active users) to estimate the impact of an app's feature complexity for free mobile apps. Second, we conceptualize complexity as a multidimensional construct and incorporate it as separate variables of interest. We collect proprietary app information at the feature SDK level, which enables a deeper investigation of complexity at the feature level. Finally, this is the first study of its kind to explore the link between feature complexity and initial success in the unique context of freemium apps.

2.2.2 Mobile app launch timing

When to launch the product into the market is an important operational decision that affects the performance of a new product. The market size, growth rate, and market need level at the time of product launch are essential factors in new product launch success (Cooper and Kleinschmidt, 1987). To identify the optimal app launch timing, anecdotal evidence has considered the effect of homogenous seasonal demand boost on launch timings. Reports show that smartphones and tablets have higher usage rates on weekdays and summer seasons compared to weekends and winter seasons (Waber, 2014). Holidays can also be an important factor as many publishers increase their ad spending during the holiday periods with eager users installing gaming apps on their new devices received as holiday gifts (Liftoff, 2017). Industry reports also confirm that in-app purchase conversion rates show a steady rise during the 12 weeks following October as apps launched in the holiday season (November to January) experience an average of 112% increase in downloads (Datta and Kajanan, 2013). Therefore, assuming homogenous customer preference across all seasons does not explain the heterogeneity in performance outcomes across apps launched in the same

Table 2.2: Selective Literature on Product Complexity

Author	Definition	Dimensions	Role	Outcome
Vickery et al., 2016	Number of BOM components and manufacturing processes	Single	MOD	New product performance
Kreye, Roehrich, and Lewis, 2015	Number and intricacy of process steps	Multi	IV	Buyer-supplier relationship
Mikolon et al., 2015	Client perceptions on complexity	Single	IV	Cognitive capacity
Closs, Nyaga, and Voss, 2010	Number of variants per component	Single	IV	Order/Unit fill rate
Gokpinar, Hopp, and Iravani, 2010	Degree of network centrality	Single	IV	Warranty claims
J Alberto Espinosa et al., 2007	Task size and structural complexity	Multi	MOD	Team performance
Thompson, Hamilton, and Rust, 2005	Number of feature components	Single	IV	Adoption intention
Sosa, Eppinger, and Rowles, 2004	Interdependence between product development activities	Single	IV	Organizational misalignment
Kaski and Heikkila, 2002	Number of physical modules and interdependency	Multi	IV	Inventory value and operating cost
Novak and Eppinger, 2001	Average of design novelty, number of moving parts, active state	Single	IV	Vertical integration
Banker, Davis, and Sandra A Slaughter, 1998	Data density, decision density, decision volatility	Multi	IV	Software enhancement effort
Griffin, 1997	Number of product functions	Single	MOD	NPD cycle time

season. Building on this thought, we explore the potential of asymmetric impact of seasonal demand boosts given the app's feature composition.

Prior OM research considers a number of factors in order to determine the optimal launch time such as capacity constraints (T.-H. Ho, Savin, and Terwiesch, 2002), competition (Calantone et al., 2010), and trade-offs between production cost and time-to-market (Savin and Terwiesch, 2005). While not much discussed in OM literature, demand is known to have a significant impact on product introduction performance. Axarloglou, 2003 showed that in the Electrical Machinery industry, market aggregate demand fluctuations at different frequencies (i.e., seasonality, business cycles) account for 35%-80% of the variability of new product introductions. The Google search trend for keyword "best mobile games" is an indicator for market interest in downloading new mobile game apps. From the plot shown in Figure 2.1, we see strong demand seasonality tied to mobile device purchases. Based on this observation, we consider market-level demand fluctuations as an important factor in determining product launch times.

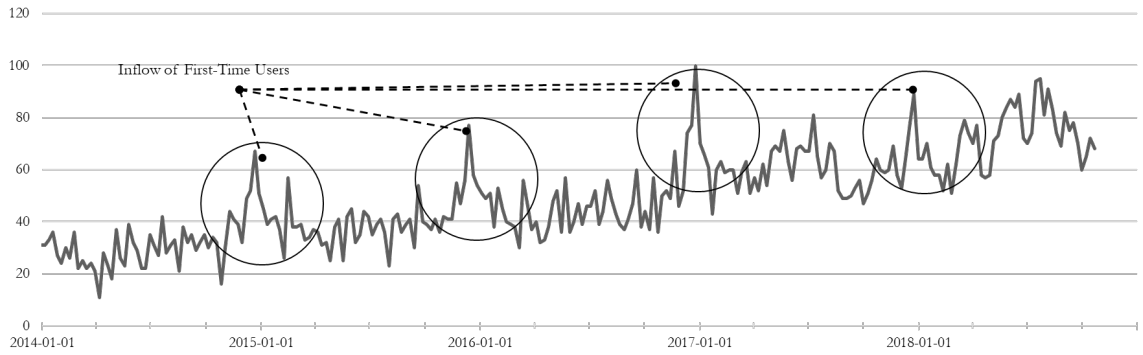


Figure 2.1: Google Search Trend for Keyword "Best Mobile Games"

Seasonality increases the potential market size which is generally considered positive, especially for products with extremely short lifecycles (Calantone et al., 2010; Krider and Weinberg, 1998). Short lifecycle products like mobile apps may not survive beyond a single season; therefore, missed opportunities from a miscalculation of product launch can be detrimental to the app's performance. If the development

process is completed during the off-season, then the developer faces a decision to launch the product immediately or wait for the next peak season. Although not all seasons are predictable, there may exist certain factors that create on-and-off seasons for mobile apps such as holidays, government actions, industry traditions, weather, social phenomena, summer and school years. Additionally, studies suggest that seasonal customers, those who use extra available time on apps, drive seasonal demand for mobile apps (Liftoff, 2017; Liss, 2017).

We also look into studies that examine the implications of seasonal demand. Closs, Nyaga, and Voss, 2010 show that seasonal demand can exacerbate the problems introduced by product complexity and capacity constraints, and further reduce service level performance. However, demand seasonality is treated as a contextual factor, which is beyond a firm's control. We treat launch timing as a strategic lever to maximize app performance. A similar approach is used by Radas and Shugan, 1998 who demonstrate the importance of the shape of the product lifecycle for timing strategies under seasonal demand. They suggest that demand seasonality can be useful in determining product launch timings. However, their study assumes homogeneous consumer preferences across demand seasons and do not consider possible interactions between demand seasonality and product characteristics. This study explores whether the effect of seasonal demand-based launch timing strategies is universally beneficial for all type of products.

In contrast to Radas and Shugan, 1998, our study argues that the benefits of increased market activity due to seasonality effects may not be universal. Even for products that are launched simultaneously during peak seasons, there is significant heterogeneity in their performances. New customers added to the market due to seasonality could have specific preferences which may not boost demand for all products. The richness and diversity of app features could play a critical role in customers' intentions to consider a specific mobile app. Extant literature does not examine these

interactions between mobile app features and market activity. We aim to address this shortcoming by examining the interactions between product characteristics and market activity. In this way, we attempt to develop a normative recommendation framework for companies to determine the optimal timing strategy given their product characteristics.

2.2.3 Research Hypotheses

Two dimensions of app feature complexity: Diversity and Richness

We argue that the impact of complexity on app market performance will materialize through two mechanisms — potential user perceptions of app quality and word-of-mouth publicity by users who have evaluated the app (Godes and Mayzlin, 2004). According to the goods and services quality classification by Nelson, 1970, search goods are products or services for which quality evaluation relies on attributes a consumer can determine before purchasing the product, and experience goods are products or services of which the quality evaluation rely on attributes that can only be discerned after purchase or during consumption. Mobile apps fall within the category of experience goods, where quality evaluation is relatively more difficult before the purchase, and therefore, indirect signals of quality such as visual cues and word-of-mouth are essential for user adoption. When a user is exposed to an app download page, the app's capability and feature sets can be inferred from the app's descriptions and posted screenshots. This allows the users to visually evaluate the quality and fit of the app with the user's usage purposes. Additionally, word of mouth and valence from user-written reviews and social interactions can further influence the app's performance. User review comments and scores are one of the critical information displayed for each app page to maximize its effectiveness. The effects of word of mouth have proven to be useful in the motion picture industry where studies have found that positive word of mouth increases attendance and ultimately,

box office performance (Duan, Gu, and Whinston, 2008).

These mechanisms should result in opposing effects of feature richness and diversity on performance, with market performance benefiting from increased feature richness while suffering from increased feature diversity. Specifically, we argue that feature richness may have a significant positive impact on the app performance. Prior research on flow (Csikszentmihalyi, 1990), cognitive absorption (Agarwal and Karahanna, 2000), and immersion (Jennett et al., 2008) have shown that perceived usefulness and ease of use of games can induce a state of continuous use. If each feature component has a non-negative, non-zero value, the addition of those features has a positive impact on the final product's perceived ease of use and usefulness (V. A. Zeithaml, 1988). Consumers perceive products with more features to be superior than other competing products considered (C. L. Brown and Carpenter, 2000). Moreover, the choice set of added features allows the app to differentiate itself from other competitors in the market (Nowlis and Simonson, 1996), and this occurs even when an added feature fails to add a significant benefit to the user (Carpenter, Glazer, and Nakamoto, 1994). Additionally, online user reviews play an important role for consumers which substitute and complement other forms of offline communications about the product quality (Chevalier and Mayzlin, 2006). Users that are content from the rich set of features may post positive reviews and spread word of mouth further increasing the user base. Therefore, we predict that feature richness will have a positive impact on the launch success of the app due to improved user perceptions about its capabilities, positive word of mouth and higher differentiation compared to competing apps.

On the other hand, we argue that feature diversity may have a significant negative impact on the app performance. Recent years have seen a proliferation of SDKs (SafeDK, 2018), making it easier for developers to add features. Although adding more features (via SDKs) to the product increases its market reach and, thus, makes

the product generally appealing to the mass market, it also increases the integration effort and leads to potential problems of crashes, viruses, malware, privacy breaches, battery drain, and lags that can undermine users' experience. For these reasons, recent industry reports express concern about having too many SDKs embedded in a single app (Shoavi, 2017). Moreover, every additional feature category requires the user to learn and search for information to achieve their goal (Nielsen, 1994). While learning new SDKs within a feature requires incremental effort that is built on the prior use of other SDKs in the same feature, acquiring knowledge about a new feature would require sizable additional effort in understanding the purpose of the new feature and supporting SDKs. Chief developer of one of the most complex games currently being sold "Magic the Gathering" points out that managing complexity âĂŞ measured by word count on the playing decks in this game - from new features and updates is one of the development team's most significant struggles that limits the entry of new players (Stoddard, 2017). Some experiments have shown that adding more features can cause "feature fatigue," which can be detrimental to users' ease of use of the product (Thompson, Hamilton, and Rust, 2005). Moreover, more features can lead to more menus and navigation to finally reach the core function that the user is looking for, which adds discomfort for the user. Specific feature categories such as in-app advertisements can be too intrusive such that it hurts the flow of user experience. This is likely to negatively influence the app's adoption via word of mouth and visual inference. Furthermore, prior research has also shown that the ease-of-use of the product can also impact the user's intentions to provide online reviews (Picazo-Vela et al., 2010). For these reasons, we hypothesize a negative relationship between feature diversity and the app performance.

Hypothesis 1. *As the feature richness of the app increases, the peak daily active users of an app during early life-cycle stage increase.*

Hypothesis 2. *As the feature diversity of the app increases, the peak daily active users of an app during early life-cycle stage decrease.*

Although we hypothesized opposing effects for feature richness and diversity, there may also exist interactions between these two constructs. If the addition of a feature in an app with a rich set of SDKs is such that the combined added value offsets the learning cost exerted on the users, the result can be performance enhancing. Prior studies on audience engagement in the context of education found that content variety positively enhanced user engagement (Webster and H. Ho, 1997). Once the learning cost is subsumed into the added value of the feature, the impact of increasing feature diversity can appeal to a broader range of users. If this is the case, feature richness is thought to have an enhancing effect on feature diversity. We test for the interaction between feature diversity and richness by proposing H3.

Hypothesis 3. *App feature richness positively moderates the impact of feature diversity on the peak daily active users of an app during the early lifecycle stage.*

Market activity in mobile apps

Having increased user activity through seasonal inflow may have positive benefits for the app. However, it is possible that features in the app may not universally appeal to all customers during all seasons. For example, in the travel industry, demand is characterized with high seasonality where customer segments are distinguishable between business customers who seek services on a regular basis and leisure customers who use services sporadically during certain times. Therefore standard revenue management practice in the hotel, airline, and search aggregators involves price and quality discrimination between customer segments (Talluri and Van Ryzin, 2006). This shift in market segments also crosses over to related mobile apps such as Airbnb and Uber where dynamic pricing is utilized to maximize revenues. In the movie industry, studies have found that DVD purchases are more affected by seasonal demands rather

than DVD rentals, which shows support for the assertion that seasonal demand can have asymmetric consumer preferences (Mukherjee and Kadiyali, 2011). Similarly, in the mobile app market, in addition to the loyal, experienced users that download and try out new apps on a regular basis, we can think of seasonal demand which adds customers who intend to utilize the extra available time on these apps. Given that these seasonal users enter the market for a short duration and have limited time to complete the games they decide to download, we speculate that these customers will be inclined towards mobile apps that are of high quality and challenging, but which have a lower learning curve.

Feature richness improves the perceived quality of the mobile app, while also potentially making the mobile game more challenging. Thus, increased feature richness should result in heightened levels of positive word of mouth being shared. Consumers that experience satisfaction and commitment from quality service, tend to share and spread that experience through positive word of mouth (T. J. Brown et al., 2005). Also, apps that provide an immersive experience can motivate the users to post positive reviews and comments on the app download page, on external user communities, and on social network media. Hence, we argue that the effect of feature richness on launch success will be further enhanced when there is a high level of market activity.

An increase in feature diversity indicates an increased number of feature categories. While learning new options within a feature requires incremental effort that is built on the prior use of that features within a category, acquiring expertise in a new feature category would require sizable additional effort in understanding the purpose of the new category and supporting features. Seasonal users who prefer mobile apps with a lower learning curve will likely download apps with lower feature diversity. Hence, we argue that apps with lower feature diversity will benefit more from seasonal increases in market activity. For these reasons, we formally state our hypothesis as follows, and test to see how the cyclical trends identified using time

series methods relates to the initial stage performance of an app. We present our research framework along with all the hypotheses in Figure 2.2.

Hypothesis 4a. *The positive effect of increasing feature richness on early stage peak daily active users will be enhanced as market activity increases.*

Hypothesis 4b. *The positive effect of decreasing feature diversity on early stage peak daily active users will be enhanced as market activity increases.*

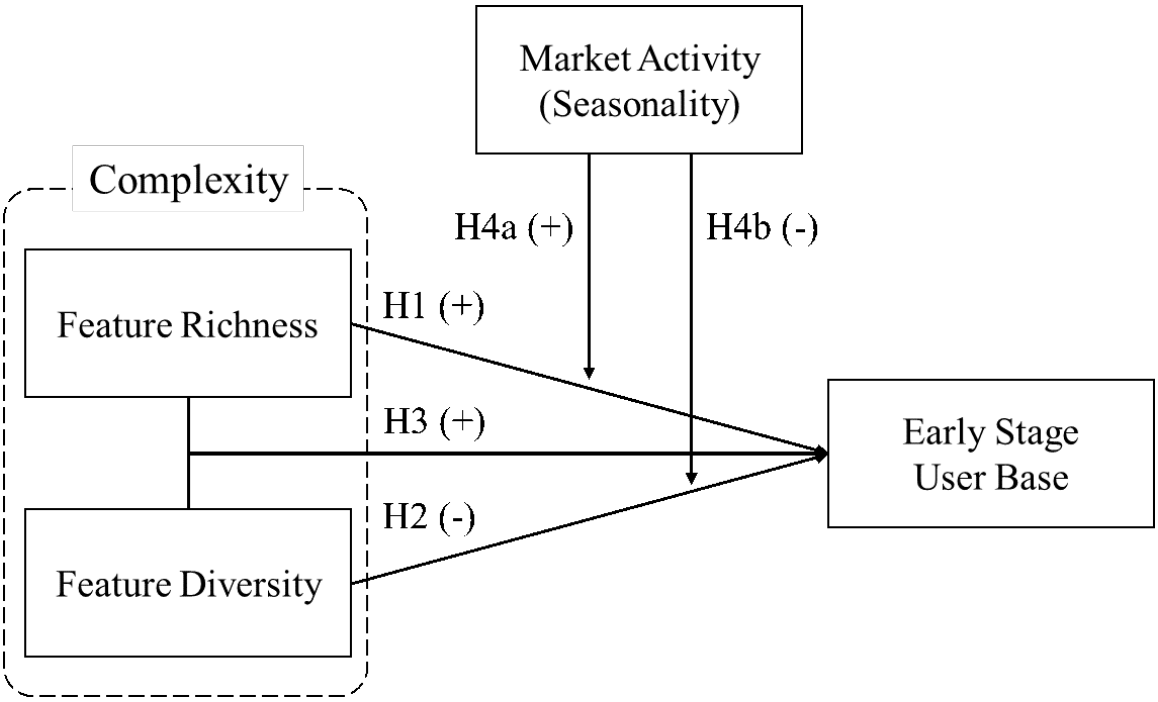


Figure 2.2: Conceptual Framework

2.3 Sample Construction and Data Description

In this section, we discuss the sample construction process of our data and describe the variables included in the analysis.

2.3.1 Sample Construction

We extract mobile app market data from the Application Programming Interface (API) server of a leading app store intelligence company. The company maintains a database on performance metrics, app characteristics, publisher characteristics of iOS and Android apps worldwide. The detailed information regarding SDK is only maintained for top 1,000 gross download apps which are updated daily. When an app enters the ranking chart at a given date, its SDK information is added to the database. At the time of data extraction, the company maintained a global (160 countries) daily panel database of approximately six million apps in the iOS and Android market over 3.5 years starting from 2015 till 2018. For all apps, we extracted general information such as file size, price, in-app purchase availability, release dates, and image file URLs for both apps and publishers at time of launch. As for app performance metrics, we collected daily/monthly active users, total revenue, and the number of daily downloads. Moreover, for approximately one million iOS ranked apps, we also collected version history and SDK installation dates. The extraction was performed using a research university's high-performance cluster computing server for parallel processing of extraction and compilation.

In this study, we focus on the top 1,000 ranked apps in the U.S. iOS app store because the iOS market provides a user experience on relatively homogeneous iPhone devices. This eliminates concerns of unobserved user device characteristics from our model estimations. The iOS platform also has extensive guidelines that SDKs must conform to. This results in a highly integrated SDK environment that provides a more homogenous experience to the users. Also, focusing on a single country allows us to reduce country-level confounds and overcome language barriers in the data. From the original three million apps, our target sample yielded daily panel observations of 1,782 mobile gaming apps in the U.S. over a period of 3.5 years. Finally, we identify our time point of initial success for each app using the method described in Section

, and convert it into cross-sectional data to conduct the analysis.

2.3.2 Variable Description

For each variable description, we include subscripts i to denote apps, j for app developers, c for the subcategory, t for initial success date, and l for app launch date. The list of variables used in the model is also summarized in Table 2.3.

Dependent variable

The dependent variable $lndau_{i,t}$ is app i's natural logged daily active users at the time of initial success t. Daily active users are the number of users that open the app at least once during a certain date. The total number of daily active users until the point of initial success captures the magnitude of the app's user base, which is a reliable indicator for the app's user base expansion performance at the early stage of product launch. The time of initial success t is identified by calculating a short-term three-day moving average and a relatively long-term ten-day moving average of daily active users, and then finding the first cross-over points. The peak is defined as the max daily active user level that occurs before the two moving averages cross-over. We consider moving averages of the time series to reduce concerns of biased peak identification due to random daily fluctuations in the data. An example of identifying the initial peak using two moving average lines is shown in Figure 2.3. Since the use of a ten-day-moving-average ignores crossovers that may occur within the first ten days of the app, we check and confirm that there are no cases where more than one crossover occurs within the first ten days since app launch. Therefore, the first intersection of the two moving average lines properly isolates the first peak and any subsequent peaks in the $lndau_{i,t}$ trajectory. To validate the local maxima, we also took the first difference of the three-day moving average of $lndau_{i,t}$ and found the point where the signs of the first difference changes from $+$ to $-$. Both methods

Table 2.3: List of Variables Used in the Model

Variable	Description
Dependent Variable	
$lndau_{i,t}$	App i's natural logged number of daily active users (i.e., users who opened the app at least once on a given date) at the time of initial success t
Independent Variables	
$diversity_{i,t}$	The total number of app i's installed SDKs feature categories at time of initial success l
$richness_{i,t}$	The total number of app i's all installed SDKs at time of initial success t
$lnmktact_{c,l}$	HP filter decomposed cyclical component of the natural logged total number of app downloads for all the apps in subcategory c at the app launch date l
$lnmkttr_{c,l}$	HP filter decomposed trend component of the natural logged total number of app downloads for all the apps in subcategory c at the app launch date l
$competition_{c,t}$	Natural logged total number of apps launched in the store for each app in subcategory c at the time of initial success t
$pubexp_{j,l}$	Cumulative number of apps developed and launched by the publisher j at the time of launch t
$screenshots_{i,l}$	App i's number of screenshots posted on the market app page at the time of launch l
$ageres_{i,l}$	Four-classification age-restriction levels (1="4+", 2="9+", 3="12+", 4="17+") for app i at time of launch l
$multicategory_{i,l}$	Number of subcategories that app i is enlisted at time of launch l
$sincelaunch_{i,t}$	The number of days passed for app i to reach the initial success at time t
$updates_{i,t}$	Cumulative number of major and minor updates implemented in app i until initial success time t
$multiplatform_{i,l}$	Indicator variable coded as 1 for app i that are launched in multiple operating platforms at the time of launch l
$appsize_{i,l}$	The app i's file size measured in bytes at the time of launch l
Instrument Variable	
$avgrich_{c,l}$	The average number of total installed SDKs in apps within the same subcategory c at the time of launch l
$avgdiv_{c,l}$	The average number of SDK categories in apps within the same subcategory c at the time of launch l
$movies_{i,l}$	The natural logged daily gross sales of top 10 box office movies at the time of app launch l

yielded 100% identical results. Also, to demonstrate the robustness of our results, we consider alternative performance measures such as number of downloads and the cumulative number of downloads at the initial success point.

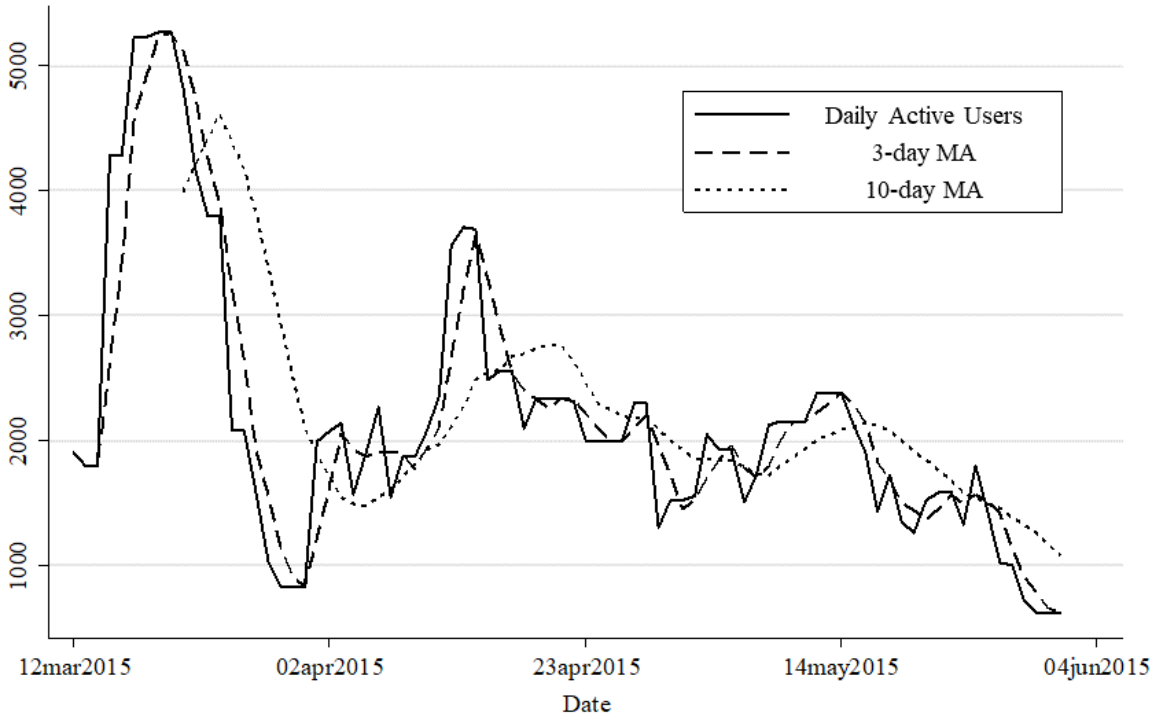


Figure 2.3: Initial Peak Identification of "Cinderella Fall" by Disney

Independent variables of interest

For independent variables, $richness_{i,t}$ is the total number of app i's installed SDKs at time of initial success t. As each SDK contributes to a feature in an app, the total SDK count captures the intensity of the features embedded within an app. For $diversity_{i,t}$, we count the number of SDK categories within an app. The mobile app market intelligence firm maintains a well-defined set of SDK categories which includes development platforms, messaging tools, social sharing, advertising networks, multimedia playback, user analytics, etc. SDKs that serve different categories are considered more distant to SDKs that serve the same feature category. Therefore, as more categories are incorporated in an app, higher degrees of heterogeneity result

among the installed SDKs. This information on SDKs is collected by downloading and reverse engineering all subject apps and analyzing their file structures. The variable $lnmktact_{c,l}$, and $lnmkttr_{c,l}$ are the cyclical and trend component of the natural logged total number of app downloads for all the apps in subcategory c at the date an app was launched l. Although the variable of interest is $lnmktact_{c,l}$, $lnmkttr_{c,l}$ is also introduced in the model to control for the increasing trend in mobile app downloads over time. We use the Hodrick-Prescott (HP) high-pass filter to separate the cyclical and trend components from the time series. The choice of the smoothing parameter depends on the granularity of the time unit, and for daily time series, it is suggested to set the smoothing parameter as $\lambda = 1600 \times (365/4)^4$ (Ravn and Uhlig, 2002). We then obtain the cyclical component by optimizing the following Equation 2.1.

$$min_{\tau_t} \left[\sum_{t=1}^{T} (y_t + \tau_t)^2 + \lambda \sum_{t=2}^{T-1} \{(\tau_{t+1} - \tau_t) - (\tau_t - \tau_{t-1})\}^2 \right] \tag{2.1}$$

Where τ_t represents the trend component at time t which is subtracted from the time series to obtain the cyclical component (Hodrick and Prescott, 1997). The trends decomposed using the HP filter are illustrated in Figure 2.4. We can observe demand surges surrounding Thanksgiving and post-holiday months such as January. These trends coincide with anecdotal evidence (Liftoff, 2017), which indicates demand seasonality in mobile games peaking around January and then decreasing until December. This boost in demand is linked to the inflow of eager users exploring and installing games for the first time on new devices they received as holiday gifts (Liftoff, 2017).

Control variables

For control variables, $competition_{c,t}$ is the natural logged total number of apps launched in a subcategory c at the time of initial success t. This controls for the increase in market competition due to new apps being launched at peak seasons. Variable

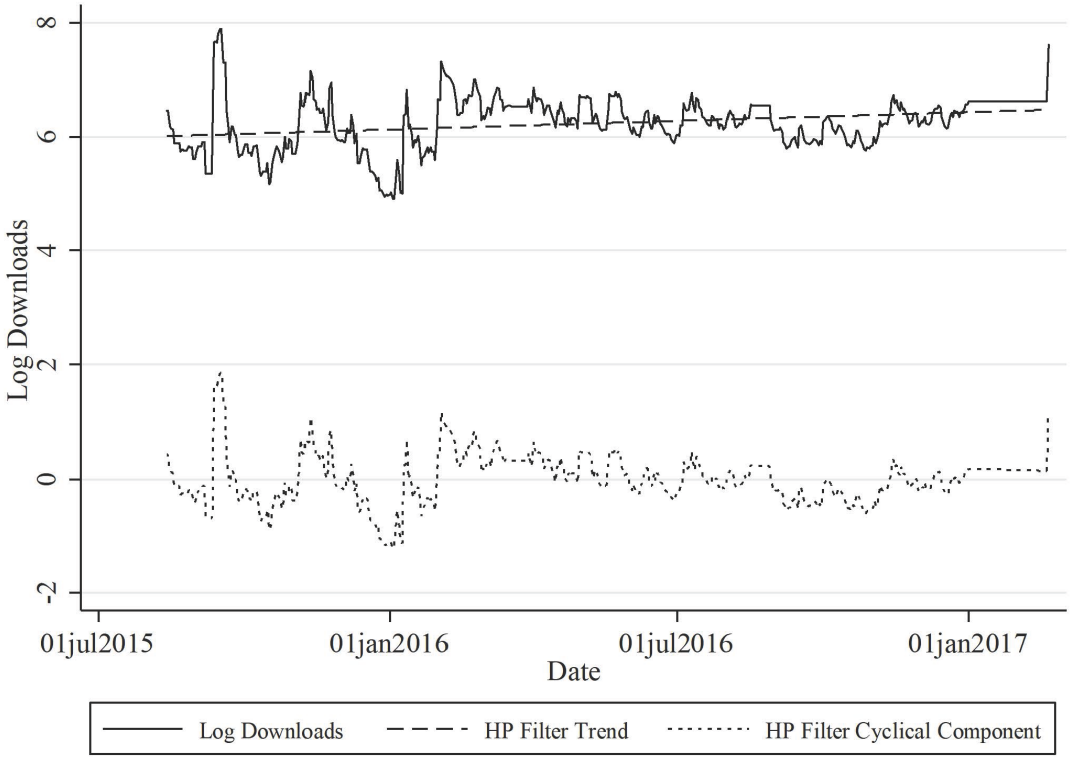


Figure 2.4: Trend and Cyclical Component of Market Activity (e.g., Arcade Category)

$pubexp_{j,t}$ is the cumulative number of apps developed and launched by the publisher j at the time of launch t. This variable controls for publisher experience and brand effects that may influence the initial performance of the app. $screenshots_{i,l}$ is app i's number of screenshots posted on the market app page at the time of launch l. Variable $ageres_{i,l}$ is the four classification age-restriction levels coded as 1 for "4+," 2 for "9+," 3 for "12+," and 4 for "17+" for app i at time of launch l, which is treated as a continuous variable in our analyses. Variable $multicategory_{i,l}$ is the number of subcategories that app i is enlisted at time of launch l. As the number of enlisted subcategories increases, the app can get exposure to a broader user pool through cross-genre listings. Variable $sincelaunch_{i,t}$ is the number of days passed for app i to reach the initial success at time t. This variable is to control for the heterogeneous download growth speed across apps. Variable $updates_{i,t}$ is the cumulative number of

major and minor updates implemented in app i until initial success time t. Variable $multiplatform_{i,l}$ is an indicator variable coded as 1 for app i that is launched in multiple operating platforms at the time of launch l to control for potential spillover effects across app stores. We also control for the app's file size measured in bytes at the time of launch l using $appsize_{i,l}$. Finally, we include subcategory fixed effects to control for potential user behavior heterogeneity. Table 2.4 shows the summary statistics and correlations of the key variables used in our model.

2.4 Econometric Model

We first acknowledge the endogeneity in SDK choice and launch timing decisions made by app developers. We present the variable used to instrument app feature complexity, and further complement the instrument by adding heteroscedasticity-based instruments (Lewbel, 2012).

2.4.1 Endogenous SDK Selection and Launch Timing

Developers' choice of installing specific SDKs and choosing the appropriate time to launch the app may depend on unobserved characteristics such as firm resources, market insights, and other development issues. If the abundance of firm resources is realized through other performance-enhancing activities such as marketing and promotion, which is omitted from our dataset and empirical model, our estimation for the variables of interest, $richness_{i,l}$, $diversity_{i,l}$, and $lnmktact_{c,l}$ can be significantly biased (Wooldridge, 2010). To address this concern of endogeneity, we rely on instruments that explain the choice of SDKs and market activity at the time of launch but is not correlated with the omitted factors. For endogenous SDK choice, we use instruments $avgrich_{c,l}$, and $avgdiv_{c,l}$ which are defined as the average richness and diversity of installed SDKs in apps within the same subcategory c at the time of launch l. The logic is similarly derived from previous works that identify instruments from

Table 2.4: Descriptive Statistics and Correlation Matrix

Variable		μ	σ	1	2	3	4	5	6	7	8	9	10	11	12	13
1	lndau	6.33	2.85													
2	diversity	4.97	2.48	-0.03												
3	richness	9.99	6.83	**0.06**	**0.83**											
4	lnmktact	0.26	0.65	0.04	0.00	0.01										
5	lnmkttr	9.55	1.80	**0.25**	**0.12**	**0.18**	0.02									
6	pubexp	4.46	6.96	**-0.10**	0.01	0.04	0.01	**0.12**								
7	competition	3.69	0.95	0.02	**0.06**	**0.13**	**0.19**	**0.67**	**0.32**							
8	screenshots	4.16	1.45	**0.28**	**0.13**	**0.21**	**0.12**	**0.35**	-0.03	**0.10**						
9	ageres	1.84	1.00	**-0.17**	**-0.16**	**-0.20**	**-0.05**	**-0.23**	**-0.12**	-0.04	**-0.25**					
10	multicategory	3.58	0.57	**-0.12**	-0.03	-0.01	-0.05	**-0.15**	**-0.05**	**-0.11**	**-0.12**	**0.12**				
11	multiplatform	0.29	0.45	**0.30**	**-0.07**	-0.02	**0.12**	**0.18**	**-0.07**	0.00	**0.23**	**-0.11**	-0.04			
12	appsize(mb)	146.54	202.1	**0.24**	-0.04	0.00	0.03	**0.20**	-0.03	**0.07**	**0.17**	0.00	-0.02	**0.17**		
13	updates	0.08	0.43	**0.20**	-0.01	0.00	-0.04	**0.07**	**-0.06**	**0.06**	**0.07**	0.00	**-0.06**	**0.10**	**0.10**	
14	sincelaunch	23.13	25.55	**0.25**	**-0.08**	0.03	0.00	**0.15**	**-0.15**	0.04	**0.22**	-0.04	**-0.12**	**0.16**	**0.11**	**0.14**

Note. Bold denotes significance at p<.05

average characteristics of products supplied by other firms within the same market (Berry, Levinsohn, and Pakes, 1995), or studies that use level of market power or competitive pressure within the market (Berry and Jia, 2010). Nonetheless, mobile app developers' decision on including certain SDKs can be affected by competition in the market because reverse engineering of downloaded apps and identification of generic SDK components within them is relatively easy for software products.

To instrument market activity levels at the time of app launch, we use an instrument of $movies_{i,l}$, which is defined as the natural logged gross sales of top 10 box office movies at the time of launch l. Movies and mobile games are both experiential goods that provide entertainment. The demand fluctuation in the movie industry can be a proxy for the level of attention towards entertainment goods and availability of time for potential mobile game users. This level of attention is also known to be high around vacations and holiday seasons (Calantone et al., 2010), and therefore is correlated to the market activity levels in mobile app stores. At the same time, the performance of movies does not directly affect the performance of mobile applications, which makes box office sales an ideal instrument for our situation.

Moreover, in subsequent models where $richness_{i,l}$, $diversity_{i,l}$, and $lnmktact_{c,l}$ interacts with exogenous variables, we additionally include the interactions between our instruments and the exogenous moderator as instruments for both the endogenous variable and the endogenous interaction term (Bun and Harrison, 2018). Because we lack additional external instruments which make the model exactly identified, we supplement the instrument with additionally generated instruments as simple functions of the model's data to improve the efficiency of the estimator and allow over-identification tests for model assumptions.

2.4.2 Heteroscedastic Covariance-Restricted IV Regression

We evaluate SDK implementation and product launch timing decisions on a software product's market performance at the app-unit level of analysis. For app i, subcategory c, and publisher j at time of launch l, and time of initial success t, we formulate the system of equations as follows.

$$\begin{aligned} lndau_{i,t} = \beta_0 + \beta_1 richness_{i,l} + \beta_2 diversity_{i,l} + \beta_3 lnmktact_{c,l} \\ + B\Gamma + \alpha_c + \epsilon \end{aligned} \tag{2.2}$$

$$\begin{aligned} richness_{i,l} = \beta_{10} + \beta_{11} avgrich_{c,l} + \beta_{12} avgdiv_{c,l} + \beta_{13} movies_{i,l} \\ + \Theta_1\Gamma + \Omega_1 Z + \alpha_c + \tau_1 \end{aligned} \tag{2.3}$$

$$\begin{aligned} diversity_{i,l} = \beta_{20} + \beta_{21} avgrich_{c,l} + \beta_{22} avgdiv_{c,l} + \beta_{23} movies_{i,l} \\ + \Theta_2\Gamma + \Omega_2 Z + \alpha_c + \tau_2 \end{aligned} \tag{2.4}$$

$$\begin{aligned} lnmktact_{c,l} = \beta_{30} + \beta_{31} avgrich_{c,l} + \beta_{32} avgdiv_{c,l} + \beta_{33} movies_{i,l} \\ + \Theta_3\Gamma + \Omega_3 Z + \alpha_c + \tau_3 \end{aligned} \tag{2.5}$$

Where, B is a vector of estimated coefficients for control covariates in the vector Γ in Equation 2.2. Z is a vector of constructed instruments in addition to $avgrich_{c,l}$, $avgdiv_{c,l}$, and $movies_{i,l}$ in Equation 2.3, 2.4, and 2.5. Θ and Ω are estimated coefficients for control covariates Γ and constructed instruments Z. α_c represents subcategory fixed effects. Lewbel's method relaxes the assumption that covariates in vector X should be strictly exogenous and allows them to be correlated at higher orders, such that $cov\,(X, \epsilon^2) \neq 0$. Then a set of instruments Z can be generated by demeaning the existing covariates in vector X included in the model and multiplying the residuals from the first-stage regression, τ as in the following Equation 2.6.

$$Z = (X - \bar{X}) \cdot \tau \tag{2.6}$$

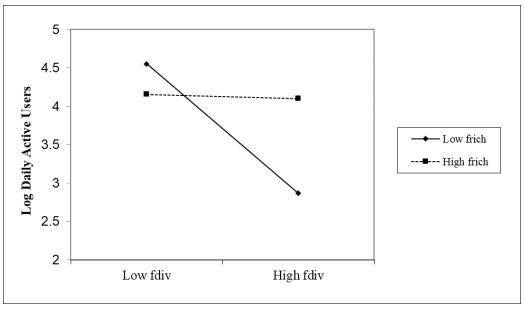
Although the resulting generated instruments can be less reliable then externally identified instruments, they can still capture the underlying common unobserved fac-

tors, given that $E[X\epsilon] = 0$, $E[X\tau] = 0$, and $cov(Z, \epsilon\tau) = 0$. Moreover, the strength of the instrument is proportional to $cov(Z, \tau)$, which corresponds to the degree of heteroscedasticity of τ with respect to Z (Lewbel, 2012). Therefore, to ensure the strength of our generated regressors, we perform the Pagan-Hall test (Pagan and Hall, 1983) and confirm significant heterogeneity in our data ($146.356, p < 0.01$). For interactions terms involving the endogenous variable, we include interactions between the endogenous variable with exogenous variables and include additional instruments to the model. All models are estimated with robust standard errors clustered at the publisher level.

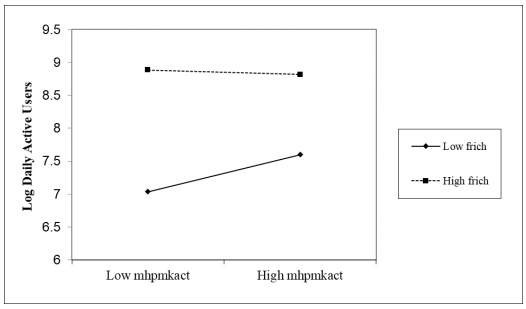
We implement a two-step GMM estimator instead of the two-stage OLS estimator which is known to be more efficient in the presence of heteroscedasticity (Wooldridge, 2010). Under-identification test using the Kleibergen-Paap rk LM statistic shows that our models are adequately identified ($p < 0.05$) (Kleibergen and Paap, 2006). The Kleibergen-Paap rk Wald F statistic are all above the Stock-Yogo weak identification test critical values of 10% maximal relative bias and size (Stock and Yogo, 2005), which supports the strength of our instruments. We test the over identifying restrictions (the unobserved error process and our instruments are orthogonal), using the Hansen J statistic (Hansen, 1982). We do not find significant correlation across all models ($p > 0.05$). Overall, the test results jointly support the validity of our instruments.

2.5 Empirical Results

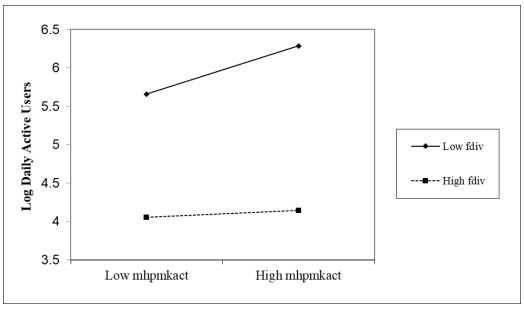
In this section, we present our results from estimating the instrumental variable regression model and demonstrate robustness to several alternative specifications.



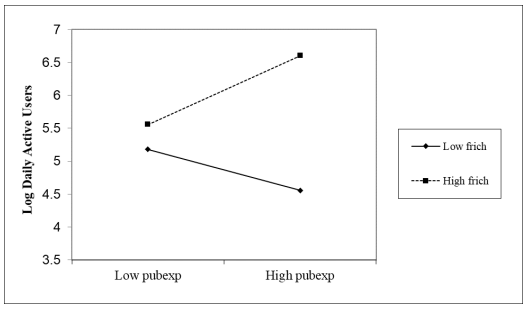
(a) Richness×Diversity



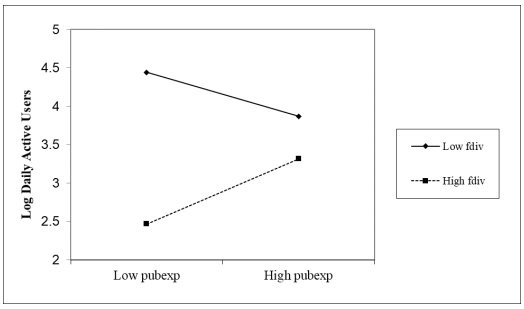
(b) Richness×Market Activity



(c) Diversity×Market Activity



(d) Richness×Experience



(e) Diversity×Experience

Figure 2.5: Interaction Plots

2.5.1 Estimation Results

The results of the estimates are shown in Table 2.5. The first column shows results without any control variables. With the added set of control covariates in Model (2), the results of our focal independent variables estimates are qualitatively consistent. We also present the interaction plots in Figure 2.5.

In Model (2), the first order term estimates of our focal independent variables, feature richness and diversity are presented. We find that as the number of SDKs implemented in the app increases by one unit, the number of daily active users at the

Table 2.5: Main IV Regression Results

	(1)	(2)	(3)	(4)	(5)
$richness_{i,l}$	0.022	0.068***	−0.083***	0.119***	0.116***
	(0.069)	(0.025)	(0.032)	(0.022)	(0.021)
$diversity_{i,l}$	−0.206	−0.231***	−0.404***	−0.352***	−0.348***
	(0.183)	(0.081)	(0.071)	(0.068)	(0.059)
$lnmktact_{c,l}$	−0.653**	0.422***	0.364***	0.544***	0.680***
	(0.304)	(0.119)	(0.110)	(0.133)	(0.194)
$lnmkttrend_{c,l}$	0.457***	0.509***	0.441***	0.294**	0.318**
	(0.065)	(0.161)	(0.144)	(0.135)	(0.151)
$pubexp_{j,l}$		−0.011	−0.004	−0.007	−0.008
		(0.008)	(0.007)	(0.007)	(0.007)
$competition_{c,t}$		−0.863***	−0.756***	−0.681***	−0.698***
		(0.172)	(0.155)	(0.144)	(0.162)
$screenshots_{i,l}$		0.242***	0.246***	0.185***	0.172***
		(0.048)	(0.048)	(0.044)	(0.045)
$ageres_{i,l}$		−0.183**	−0.232***	−0.155**	−0.185**
		(0.088)	(0.082)	(0.079)	(0.083)
$multicategory_{i,l}$		−0.166	−0.135	−0.268**	−0.264**
		(0.136)	(0.123)	(0.119)	(0.120)
$multiplatform_{i,l}$		1.022***	1.028***	0.944***	0.927***
		(0.173)	(0.153)	(0.162)	(0.159)
$appsize_{i,l}$		0.002***	0.002***	0.002***	0.002***
		(0.000)	(0.000)	(0.000)	(0.000)
$updates_{i,t}$		1.072***	1.040***	1.080***	1.027***
		(0.125)	(0.113)	(0.126)	(0.124)
$sincelaunch_{i,t}$		0.012***	0.013***	0.012***	0.013***
		(0.002)	(0.002)	(0.002)	(0.002)
$richness_{i,l} \times diversity_{i,l}$			0.023***		
			(0.003)		
$lnmkact_{c,l} \times richness_{i,l}$				−0.035***	
				(0.010)	
$lnmkact_{c,l} \times diversity_{i,l}$					−0.082***
					(0.032)
Constant	3.046***	4.418***	5.602***	6.832***	6.677***
	(0.663)	(1.384)	(1.247)	(1.206)	(1.255)
Subcategory FE	NO	YES	YES	YES	YES
Observations	1,782	1,782	1,782	1,782	1,782
R-squared	0.032	0.277	0.285	0.275	0.275
Number of Publishers	711	711	711	711	711

Robust standard errors clustered by publisher in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

initial peak increases by 6.8% on average $(0.068, p < 0.01)$. On the other hand, a one unit increase in SDK categories in the app leads to an 23.1% decrease in the number of daily active users at the initial peak $(-0.231, p < 0.01)$. These results provide support to our hypotheses H1 and H2. Although not hypothesized, the impact of launching the app when market activity is 1% higher is associated with a 42.2% increase in number of daily active users at the initial peak $(0.422, p < 0.01)$. This emphasizes the importance of a well-timed launch of mobile applications.

Next, we introduce interactions into the model to estimate the interaction effects between the focal independent variables. As shown in Model (3), we find a significant positive interaction between feature richness and diversity $(0.023, p < 0.01)$. The marginal effect of richness conditional on levels of diversity as shown in Figure 2.5(a) suggests that the negative effect of high diversity is mitigated when there are high levels of feature richness. This confirms the enhancing effect such that the learning cost can be mitigated by added value from SDK components. Thus, our hypotheses H3 is supported.

For richness and market activity interactions, we find a significant negative interaction as shown in Model (4) $(-0.035, p < 0.01)$. Interestingly, this result is in the opposite direction from our hypothesis H4a. The significant negative interaction suggests that seasonal customers do not reward feature richness of apps. As shown in Figure 2.5(b), only apps with relatively lower feature richness are able to enjoy a significant increase in early-stage user base. Between diversity and market activity, we find a significant negative interaction as shown in Model (5) $(-0.082, p < 0.01)$. The interaction plot, as shown in Figure 2.5(c), suggests a slight performance boost of low feature diversity apps when launched during peak seasons, and a performance loss for high feature diversity apps when launched during peak seasons. Again, results show that seasonal users do not prefer high feature diversity apps. This supports our hypothesis H4b.

For control variables, we find a significant positive effect of number of screenshots $(0.242, p < 0.01)$, multiplatform $(1.022, p < 0.01)$, app size $(0.002, p < 0.01)$, and number of previous updates $(1.072, p < 0.01)$. The estimated effects are qualitatively consistent regarding direction and strength as reported in the prior literature (Ghose and Han, 2014), and competition shows negative directionality $(-0.863, p < 0.01)$, which is consistent with our expectations.

2.5.2 Post-hoc Analysis

From our main analysis, we find a consistently negative impact of SDK diversity. Therefore, a natural question arises, *"How do firms overcome the negative impact of feature diversity?" "What is driving the negative feature diversity effect?"* To answer this question, we conduct two post-hoc analyses. First, we tap into the literature on organizational learning as applied in the context of software development (Fong Boh, Sandra A. Slaughter, and J. Alberto Espinosa, 2007; Narayanan, Balasubramanian, and Swaminathan, 2009). Because the decision of choosing the optimal set of SDKs can be a complex problem, it is possible that firms get better through repetition and experience accumulation. Therefore, we interact the publisher prior development experience and feature diversity to see whether firms can turn the diversity impact into a positive one through refined decision making.

Using the identical IV regression specification, we incorporate two interactions, feature diversity and experience, and feature richness and experience. The estimation results are presented in Table 2.6.

We find that experience positively moderates both feature richness (Model (6): $0.006, p < 0.01$) and feature diversity (Model (7): $0.014, p < 0.01$). Plots of these interactions are shown in Figure 5d and 5e. Feature diversity shows a negative impact at low levels of publisher experience. As publishers gain experience from prior app launches, this negative impact of diversity is mitigated. Moreover, publishers that

Table 2.6: Post-Hoc IV Regression Results

	(6)	(7)	(8)
$richness_{i,l}$	0.053 * *	0.079 * **	0.090 * **
	(0.022)	(0.017)	(0.023)
$diversity_{i,l}$	−0.282 * **	−0.328 * **	
	(0.059)	(0.051)	
$lnmktact_{c,l}$	0.436 * **	0.440 * **	0.458 * **
	(0.104)	(0.108)	(0.110)
$lnmkttrend_{c,l}$	0.544 * **	0.540 * **	0.567 * **
	(0.147)	(0.149)	(0.154)
$pubexp_{j,l}$	−0.050 * **	−0.062 * **	−0.006
	(0.009)	(0.012)	(0.007)
$competition_{c,t}$	−0.926 * **	−0.925 * **	−0.931 * **
	(0.162)	(0.163)	(0.163)
$screenshots_{i,l}$	0.193 * **	0.205 * **	0.234 * **
	(0.044)	(0.043)	(0.045)
$ageres_{i,l}$	−0.191 * *	−0.171 * *	−0.153*
	(0.078)	(0.076)	(0.082)
$multicategory_{i,l}$	−0.189	−0.225*	−0.337 * **
	(0.120)	(0.118)	(0.121)
$multiplatform_{i,l}$	0.963 * **	0.915 * **	0.889 * **
	(0.157)	(0.156)	(0.165)
$appsize_{i,l}$	0.002 * **	0.002 * **	0.002 * **
	(0.000)	(0.000)	(0.000)
$updates_{i,t}$	1.163 * **	1.157 * **	1.099 * **
	(0.117)	(0.120)	(0.120)
$sincelaunch_{i,t}$	0.012 * **	0.012 * **	0.012 * **
	(0.002)	(0.002)	(0.002)
$pubexp_{j,l} \times richness_{j,l}$	0.006 * **		
	(0.001)		
$pubexp_{j,l} \times diversity_{j,l}$		0.014 * **	
		(0.003)	
$devfeat_{i,l}$			−0.172
			(0.110)
$monetization_{i,l}$			−0.357 * **
			(0.088)
Constant	4.878 * **	5.099 * **	4.834 * **
	(1.221)	(1.226)	(1.304)
Subcategory FE	YES	YES	YES
Observations	1,782	1,782	1,782
R-squared	0.282	0.284	0.276
Number of Publishers	711	711	711

Robust standard errors clustered by publisher in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

launch low diversity apps even after accumulating prior experience may see decreased performance. On the other hand, highly experienced firms see stronger benefits from launching feature-rich apps. Again, if a firm with experience launches a low richness app, the app may yield lower performance levels.

Next, in order to answer the second question, we divided the feature diversity variable into two sub diversity variables of development feature diversity and monetization feature diversity. Development features are groups of SDKs essential for app development such as development tools, crash reporting, data hubs, and testing. On the other hand, monetization features are SDK groups that enable the developer to build revenue streams by gaining an in-depth understanding of user behavior, sending targeted promotions, providing convenient in-app purchase methods, and embed in-app advertisements. We re-estimate the IV regression with the separated feature diversity variables to see whether the negative impact of diversity pertains to monetization features.

Estimation results for the second posthoc test are shown in Table 2.6. We find a significant negative effect of monetization features ($-0.357, p < 0.01$) and insignificant effect of development tool features ($-0.172, p > 0.10$). These results suggest that the negative impact of feature diversity mostly comes from monetization features embedded in the app. Monetization features such as in-app advertisements are known to be very intrusive and hinder the immersion of the user. Additional permissions and data requirements from user data mining features may increase the data traffic and lag the app performance. However, these features are necessary for the developer to generate revenue streams. We confirm a trade-off relationship between user experience and developer monetization. It is important for the developer to know the severity of this trade-off and carefully balance both-sides to achieve sustainable service.

2.5.3 Robustness Checks

We demonstrate the robustness of our results by performing the analyses based on 1) alternative dependent variables, 2) alternative independent variables, 3) alternative estimation procedures, and 4) alternative samples.

For alternative dependent variables, we considered the logged number of downloads at the time of peak downloads and logged number of cumulative downloads at the time of peak. We find qualitatively consistent results from both alternative dependent variables. Next, we use alternative operationalizations for our focal independent variables of market activity and feature richness. Instead of the HP filter, we use the Christiano-Fitzgerald filter to calculate the cyclicality in market demand (Christiano and Fitzgerald, 2003). Instead of the total number of SDKs for the feature richness variable, we use the maximum number of SDKs across the SDK feature categories to measure feature richness. For both alternative operationalizations, the results were consistent with our main model results. For alternative estimation procedures, we formulate an OLS regression which does not account for the endogeneity. Finally, for alternative samples, we first account for apps that are launched by high profile publishers and second, account for apps that are developed with holiday-specific themes. We discard the top 5% apps regarding number of downloads and re-estimate our models to see whether our results were influenced by so-called "blockbuster" apps. For holiday specific apps, we identify 20 apps that contain any holiday-related terms (e.g., Christmas, Santa, Halloween, Easter, Xmas, New Year) in their titles, and dropped them from our sample. We present the results of our robustness checks in Appendix A. The reported results are qualitatively consistent with our main model results. These results collectively show support for the robustness of our results.

2.6 Discussion and Conclusions

2.6.1 Implications for Theory

Our investigation makes significant theoretical contributions to research streams in product complexity and market seasonality. First, we delineate the effect of complex product feature designs from a downstream user perspective. Rather than focusing on how complexity imposes product development challenges, we assessed whether the choice of adding more layers of complexity to the product is rewarded by the users. Even if complexity adds challenges to the development process, it can be inevitable if users prefer richer and more diverse experience. To do this, we adopt a demand-side performance metric of initial peak magnitude in daily active users, number of downloads, and cumulative downloads. This demand-side assessment is especially important because freemium mobile apps largely rely on network effects and user content generation. We find that feature set composition significantly influences the early stage user base expansion of an app.

Second, we identify product feature richness and diversity as a critical non-price competition factor in the context of mobile applications. While price is often a critical product and market indicator that drives economic theories, factors that are identified in this study provide valuable insight for explaining performance outcome of product systems when they are competing on non-price factors. Optimizing the feature set is especially important in our context as the extremely short product life cycle does not allow firms to experiment and study market reactions after the product is launched. Therefore, we contribute to the research stream in mobile apps by assessing mobile app performance at the feature-level, which becomes more relevant as freemium emerges as the dominant business model.

Third, we find asymmetric effects of the different dimensions of complexity, which helps us establish the conceptual differences between the dimensions. Prior literature

on the demand side assessment of product complexity (Kim and Hyun, 2011; Mikolon et al., 2015; Thompson, Hamilton, and Rust, 2005) mostly view complexity as a unidimensional construct or hypothesize a unified direction of multiple dimensions of complexity, failing to delineate the differences between each complexity dimension. We conceptualize the two complexity dimensions as feature richness and diversity and estimate their relative impact on the apps' performance. Findings from this study support a positive effect of complexity when components cohesively contribute to a certain feature, thereby adding richness. On the other hand, when components are scattered across a variety of feature categories, the complexity negatively affects the app's performance.

Fourth, we show how feature complexity of a product interacts with market seasonality. Specifically, we find that apps with less feature richness and diversity benefit more from a peak season launch. This finding helps us explain the heterogeneous performance outcomes of mobile apps competing during peak demand seasons. Research in economics that examines business cycles and marketing literature that examines demand seasonality have endorsed the idea of optimizing market entry timing strategies based on demand patterns. In contrast to the naÃŕve belief that more market potential is always good, we argue that it is important to examine the demand seasonality and product feature interactions. In contrast to our initial reasoning that the effect of feature richness and diversity would be more salient to the seasonal users, we find that the complexity of features reduces the attractiveness of the app to the seasonal customers. The additional inflow of users during peak seasons represent a type of users that are somewhat constrained by resources or attention spans such that product adoption occurs mostly during a specific time of year. Therefore, seasonality in demand itself can occur for a specific type of consumer segment which requires the app to be exceptionally easy to understand and use. These users do not reward the developers for richer and more diverse features. The negativity in learning

cost resulting from feature complexity dominates in these user segments, and simpler apps benefit from the seasonality based-timing. This study provides insight into an Operations-Marketing interface issue by demonstrating a significant interaction between market demand patterns and product features.

Finally, through two post-hoc analyses, we dive deeper into the underlying mechanisms of the negative impact of feature diversity. We show that app developers accumulate knowledge from prior launch experience such that they can make better decisions on SDK selection and optimize the benefits. Interestingly, by observing the interaction plots, we find that the market penalizes firms with reputation and experience if they launch low richness and low diversity apps. An explanation for this finding is that the experience variable is also capturing the firm reputation in the market. Results show that as a firm grows their reputation, possibly users in the market may expect more diverse experience and richer features in the newly launched apps. A firm that does not innovate and still maintains the low richness and low diversity may eventually suffer from reduced performance. In sum, the results suggest two things. First, publishers indeed learn over time from prior development and launch experiences and make better decisions regarding SDK implementations. From a broader picture, this shows that while managing complexity is a challenge for publishers, they improve their decision-making regarding product complexity from prior experience and excel over time. Second, there is a market pull for constant technology adoption and innovation regarding app features such that a firm that does not offer novel features in their newly launched apps may quickly lose its place in the market. We also find that the negative impact of feature diversity mostly comes from monetization features. This shows that there is a trade-off relationship between user experience and publisher revenue sources. Carefully balancing the two opposing forces poses difficult challenges for developers in this business.

2.6.2 Implications for Practice

Practitioners can benefit from the findings of this study in a number of ways. First, we have addressed a critical managerial decision in the context of mobile app development with regard to SDK choice. Now that the number of SDKs available in the market is growing exponentially, picking and choosing the right SDKs and the resulting feature set is becoming an essential problem for app developers. Our findings suggest that it is crucial to consider SDK choice from a perspective of adding more richness or diversity to the app's features. It is vital for managers to know that a diverse feature set that lacks richness can backfire and lead to reduced performance. Second, the significant interaction between market activity, feature richness and feature diversity suggest that managers should be careful in assuming that the market segment is homogenous between the peak and off-peak season. If there is a cost to postponing product launch after development completion to potentially take advantage of launching in the peak-season, this wait may not be justified, especially if the app is complex. On the other hand, apps with relatively simpler features and a straightforward value proposition can benefit more from a well-timed product launch. Fourth, the results suggest that publishers should be cautious in expanding the feature diversity of apps. It is advisable that the feature expansion takes place after accumulating several product launch experiences. Prior product launch and managing experience allows the developer to accumulate knowledge about the user preference and behaviors, which can be valuable in optimizing new feature category experience. Finally, developers should be aware of the trade-off relationships between adding features that enhance user experience versus those focused on monetization. Although these features may be tempting, a careful balance between the two will be essential for sustaining a healthy app service.

2.6.3 Limitations and Conclusions

Our first limitation is that the scope of the study pertains to iOS gaming apps only. This reduction in scope allowed us to reduce concerns related to unobserved influences from mobile device characteristics and user demographics. However, we believe that the gaming context is an extreme case regarding competition intensity and short innovation cycles, which the mobile app economy generally shares in differing degrees. Therefore, our findings on complexity and market activity should apply to other app categories as well that use modularized software development kits. Second, we do not have a more detailed performance measure that captures the actual usage of the app. The actual duration of use would be ideal to capture user engagement with the mobile app. Future research can look into user-level behavior data to strengthen the link between app features and performance outcomes.

Overall, this study sheds light on both theory and practice on the emerging trend in mobile app ecosystems. Our conceptualization focusing on the differences of mobile app lifecycles opens novel research avenues yet to be explored. Asymmetric effects of the feature complexity dimensions and their interactions with demand patterns are the primary findings of this study.

Chapter 3

Software Maintenance Strategies for Free Mobile Applications

Abstract

Nowadays, mobile apps are converging towards the freemium business model which shows extremely short lifecycles with almost instantaneous demand saturation. To retain and acquire new users after an app is launched, app developers releases enhancement updates. Enhancement updates exert substantial investment from the developers. Therefore, it is important to understand the impact of these enhancements, and identify contextual factors that further reinforce the effectiveness of these efforts. In contrast to prior software maintenance literature that focuses on optimizing the cost and efforts of software maintenance, we focus on the proactive use of enhancement efforts to effectively stimulate demand and increase the longevity of the mobile app. Specifically, through a difference-in-differences estimation with Bass model predictions as the base case, we estimate the average treatment effect of releasing enhancement updates in an app. Moreover, we explore contextual factors such as update schedule regularity, lifecycle stages, and market activity levels at the time of update to explain why certain enhancement updates can be more effective than others even within a single app. The dataset is extracted from a proprietary application programming interface (API) which contains daily app performance and file structure information of 433 free iOS gaming apps with 4,052 updates in the U.S. over 3.5 years. We classify enhancement updates using tf-idf naive Bayes text clas-

sifiers. Findings confirm significant positive effects of enhancement updates on the app download performance. Further, keeping a regular schedule, releasing updates right after the initial peak in adoptions, and high market activity levels have positive moderation effects on the enhancement update effect. Post-hoc analyses using semi-parametric and parametric survival analyses reveal that our findings are consistent in extending the app lifecycle. Our study provides both academic as well as managerial insights on how enhancement updates should be released when the firm is concerned with proactively stimulating mobile app adoption.
Keywords: Mobile apps, software maintenance, schedule regularity, app lifecycle, market activity

3.1 Introduction

Software maintenance refers to the modification of software after implementation to correct errors, to improve performance, or to adapt to a changed environment (E Burton Swanson, 1976). Software maintenance is long recognized by researchers as a costly, but a crucial process that entails around 50-80% of a firm's IT budget (Nosek and Palvia, 1990). For this reason, research has predominantly focused on understanding determinants of maintenance efforts (Banker and Sandra A Slaughter, 1997; Banker, Davis, and Sandra A Slaughter, 1998) or minimize cost (M. S. Krishnan, Mukhopadhyay, and Kriebel, 2004; Arora, Caulkins, and Telang, 2006; Kulkarni et al., 2009; Ji et al., 2011). Such software maintenance efforts are equally important for mobile applications where the cost of software maintenance generally amounts up to 20% of the entire app development cost (Moore, 2019).

Software maintenance is commonly characterized as a group of three activities. These activities are corrective maintenance (performed in response to the occurrence of system failures), adaptive maintenance (performed in anticipation of changes within the data or processing environment), and perfective maintenance (performed

to eliminate inefficiencies and enhance performance) (Lientz, E. Burton Swanson, and Tompkins, 1978; E Burton Swanson, 1976). As commonly done in practice and research, we bundle perfective and adaptive maintenance under the term *enhancement updates* (Ji et al., 2011). Additionally, organizations need to perform corrective maintenance throughout the system lifetime. We refer to corrective maintenance as simply *maintenance updates*. Dealing with maintenance activities with clear requirements, user requests, and cost estimates, in other words, the *reactive* software maintenance can be optimized through various batching and resource allocation policies. However, much less is known about the decision process of organizing software maintenance *proactively* (i.e., using enhancement activities to acquire new users and increase revenue).

By focusing on the mobile gaming industry context, this research explores the effects of one type of software maintenance effort, enhancement activities. The mobile gaming category is characterized by the highest competition intensity and revenue generation capability among all app categories. Developers in this category need to release new contents to constantly maintain user engagement. New content updates can lead to significant increases in revenue and lifetime of the app. For example, one of the most successful apps in the gaming category, Clash of Clans, has accumulated 97 updates since its launch in 2012. The company rolled out an update every three weeks on average, primarily to increase user engagement through new content roll outs and bug fixes. With higher engagement of its user base, this app generates over $1.5 million per day. Finally, because the gaming app average lifetime is the shortest among all app categories (Gordon, 2018), we can observe the entire lifetime of a sizable number of apps even within our study period. Therefore, mobile games provides a nice empirical setting to study the factors that determine the effectiveness of enhancement updates. Through this study, we attempt to answer the following research question: *Why are some enhancement updates more effective than others?*

How should mobile app developers schedule their enhancement updates? Our focus is on estimating the effect of introducing an enhancement update in a mobile app and further exploring multiple factors that can further reinforce this update effect.

The type of software maintenance updates and its rollout timing is a critical decision for mobile app developers, not only because they are expensive, but also because they can be effective in bringing in new users and retaining current users. Over time, the dominant business model for mobile apps, is gravitating towards the freemium business model where the revenue streams are generated from in-app purchases and advertisements (Taube, 2013). These revenue streams accrue slowly over time, and increase proportionately with the user base size. Therefore, the developer needs to quickly accumulate the user base size and retain that user base to break-even. Failing to address issues in the system effectively, or failing to provide more content promptly, may deter users away from the app quickly due to their short attention spans (Brauer, 2014). Especially, enhancement updates in mobile apps are often tied to various advertising activities (Norton and Bass, 1987), which play a vital role in increasing the user base in later stages of the app's life cycle.

We collect unique data from a proprietary application programming interface server of a leading app store intelligence firm. The database maintains daily observations of the app, publisher characteristics, market performance metrics, and most of all, version history, update date, and update log texts in separate endpoints. Our observation window ranges from January 2015 to December 2017. We deploy text frequency-inverse document frequency (tf-idf) text analysis with naive Bayes classifier on the update logs using supervised machine learning to classify the update types. Then, we estimate a difference-in-differences analysis using predictions of each app download time series from a Bass diffusion model (Bass, 1969) as the control group. Similar to an event study method, we restrict our sample to ± 2 days around the enhancement activity to reduce concerns from unobserved confounds. Our final sample

consists of 433 apps with 4,052 updates. In addition to estimating the enhancement activity effects, we estimate the effect of deviations from keeping a regular update schedule, updates made during certain life cycle stages, and updates made during varying market activity levels. Regular update schedule is operationalized as a rolling standard deviation metric of inter-update times. We define three life cycle stages of an app. Stage one is from launch date to date of initial peak in daily downloads, stage two is from date of initial peak in daily downloads to date when the daily downloads decrease to 50% of the initial peak, and stage three is from date of 50% of the initial peak to the date of app service termination. Market activity is operationalized as the aggregate total daily downloads in a mobile game subcategory, which consists of mobile game genres. For robustness checks, we deploy semi-parametric and parametric survival analysis models and alternative variable specifications to demonstrate the robustness of our findings.

We find that software enhancement activities are associated with a 5.23% increase in daily downloads in a two-day after treatment window. Further, a 1% increase in inter-update time variability leads to a 2% decrease in this content update effect. Regarding life cycle stages, we find that enhancement efforts made in the stage right after the initial peak in daily downloads are associated with a 44.7% increase in daily downloads compared to enhancements made before the initial peak. Finally, we find that releasing enhancement updates when the market activity level is 1% higher, the enhancement update effect increases by 9.77%. In a post-hoc analysis, we find that certain weekdays such as Thursdays and Fridays are preferable in releasing enhancement updates.

Our study makes significant theoretical contributions to software maintenance literature. First, recent analytical models attempt to jointly consider demand and supply side constraints to optimize software maintenance schedules, product launch times, and termination decisions. The lack of empirical evidence forces these models

to rely on simplistic assumptions about the demand change arising due to updates. This study provides the crucial inputs regarding the demand and additional parameters to consider when building such optimization models. Second, we argue that traditional software maintenance frameworks do not thoroughly explain the effects of enhancement updates. Therefore, in contrast with prior research that mainly considers supply-side cost implications of software maintenance, we propose a proactive view of software maintenance that aims to maximize the benefits of enhancement updates. Third, we contribute to literature on mobile apps by understanding the effects of software maintenance activities in the context of mobile apps, which is an essential piece currently missing in the literature.

Our study has significant managerial implications for mobile app developers. Through a quasi-experimental analysis, we obtain estimates of enhancement updates with minimum bias. By exploring additional parameters as moderators for this enhancement update effect, we propose three types of updating strategies that can further increase the benefit of these updates. We find schedule regularity to have relatively small moderation effects on enhancements. However, due to its known benefits from traditional software maintenance frameworks that aim at minimizing costs, many app developers adopt this updating policy. On the other hand, we find lifecycle-based enhancement and market activity-based enhancement to be viable alternative strategies. We find that even if the pursuit of these strategies accompanies disruptions to a regular update schedule, their moderation effects are significant in magnitude, and therefore should be considered by app developers when developing their update schedules.

The rest of the chapter is organized as follows. In the next section, we present a survey of the literature in this area. In section 3.3, we describe the sample construction process and data used for this study. In section 3.4, we develop the econometric model to estimate the effects of interest related to enhancement updates and timing

decisions that affect the updates. In section 3.5, we present the estimation results followed by a post-hoc analysis, which further explores micro timing strategies related to the weekday of the update. We also discuss the results of robustness checks. Finally, in section 3.6, we discuss the implications of our findings and point to future work avenues.

3.2 Literature Review

In what follows, we examine the role of software maintenance in mobile apps and discuss how variability in update schedules can moderate the effectiveness of software maintenance efforts.

3.2.1 Software Maintenance in Mobile Apps

Software maintenance is a critical task for a mobile app in increasing revenue and extending the life cycle. Software maintenance activities take up a large portion of the software developers' resources and is even considered more important than the initial development of a software (Lientz, E. Burton Swanson, and Tompkins, 1978). For large information systems (IS), the maintenance cost is estimated as high as 50-80% of the total IS budget (Nosek and Palvia, 1990). For smaller scale mobile applications, still the portion of maintenance is known to amount up to at least 20% of the app development budget (Moore, 2019). From a user's perspective, a well-executed update can increase the value of the app significantly, leading to higher satisfaction. While there is a strong need, managing software maintenance activities effectively are known to be difficult.

The literature on software maintenance provides various classifications to software maintenance activities. Earlier work in this area categorizes maintenance activities into three categories, namely adaptive maintenance, perfective maintenance, and corrective maintenance. Adaptive maintenance refers to changes in the software

to cope with environmental changes. Perfective maintenance refers to implementations of new or changed user requirements which enhances the functionality of the software. Corrective maintenance refers to fixes and patches for issues that undermine the software's functionality (E Burton Swanson, 1976). Subsequent work puts the maintenance activities into two classes of enhancement activities or maintenance activities (Lientz, E. Burton Swanson, and Tompkins, 1978). This classification is more clean cut, as adaptive maintenance can be dealt with by perfective or corrective maintenance requirements. Kitchenham et al., 1999 also adopts this typology and further provides three sub-classes for enhancement implementations such as change in existing requirements, change to accomodate new system requirements, and implementation enhancements. More recently, a typology by Chapin et al., 2001 provides a decision tree model to classify software maintenance activities into 12 update typologies. Depending on the answer to three questions, "was the software changed?" "was the source code changed?" "was the function of the software changed?" the maintenance activities could be clustered into changes in the support interface, documentation, business rules, and software properties. Although this framework is detailed, the classification relies on in-depth information provided by the software developer. Also, these differences are not directly distinguishable by the users. Therefore, this study adopts the simple two-class software maintenance typology (Lientz, E. Burton Swanson, and Tompkins, 1978; Kitchenham et al., 1999) which is comprised of enhancement activities and maintenance activities to understand the updates being made in mobile apps that have more distinguishable characteristics from a user perspective.

A recurring argument in software maintenance literature is the high cost of maintenance efforts. The cost minimization approach puts more emphasis on managing the activities related to software maintenance. For the effective management of this cost, a large body of literature is dedicated to the supply side issues in soft-

ware maintenance. Banker and Sandra A Slaughter, 1997 develops a DEA model to demonstrate scale economies in software maintenance. They find that batching minor maintenance requests into larger planned releases can alleviate software maintenance costs. Subsequently, Banker, Davis, and Sandra A Slaughter, 1998 shows that usage of software development practices such as code generators and packaged software may affect the software complexity , which is associated with higher software maintenance efforts. M. S. Krishnan, Mukhopadhyay, and Kriebel, 2004 develops a cost-minimizing dynamic optimization model to derive the optimal major update policy. Arora, Caulkins, and Telang, 2006 considers the trade-off relationship between time-to-market and after-sales software failures to determine the optimal market entry time and maintenance effort level. (Kulkarni et al., 2009) considers the total cost of software maintenance in a queuing system to determine the optimal order of batch sizes and service rates. Narayanan, Balasubramanian, and Swaminathan, 2009 studies the effect of having a balance between variety and specialization in the developer team's experience on a software firm's maintenance effort. Recently, Ji et al., 2011 develops an optimal development effort, initial feature choice, and service termination policy by considering the total discounted profit for a software system throughout its lifetime. However, the system profit is derived from the number of features in the system rather than its market valuation by the users. Collectively, prior literature treats software maintenance as a reactive task, where its cost needs to be minimized.

However, studies acknowledge that the traditional approach of software maintenance may no longer apply to the fast-changing marketing trend and technology evolution alone (Bennett and Rajlich, 2000). Specifically, mobile apps have a different characteristic and environment (Salmre, 2005) which puts more emphasis on the proactive use of enhancement updates. First, mobile apps are restricted by the capability of the device that is running the application. Device capability restrictions include screen size, battery run time, and storage space. Second, on average,

we see more frequent updates made in mobile apps because their sizes are relatively small (i.e., several thousand lines of code) which makes it easier to implement updates. Third, mobile apps are accessed anytime and anywhere. This means that users access apps at higher frequencies, but in shorter sessions. This requires the app to minimize delays and optimize their performances. Finally, updates in mobile apps are generally associated with substantial advertising behavior. Unlike IT system software maintenance activities, mobile app maintenance activities are more similar to launching a successive generation of an existing product. Moreover, each successive generation of a product aims to obtain sales by expanding the market through enhanced features (Norton and Bass, 1987). Mehra, Seidmann, and Mojumder, 2014 develop a model for packaged software life-cycles, and makes an assumption that marketing activities associated with the upgrade reaches all possible new users they could attract. For mobile apps, once an update is implemented, advertisements containing information about new features and contents are deployed afterwards via company websites, ad networks, and social network services.

Because of these differences, mobile app developers must account for some constraints when introducing new features and contents through software maintenance activities. First, software maintenance activities should be conducted promptly, minimizing disruptions to the users' experience. Performance issues, slow start times, poor network connection, and server downtime due to maintenance may all deter users who have short attention spans and low tolerance for accessibility issues. Second, to ensure quick and responsive app performance while meeting various device limitations, the application size needs to be continuously optimized. For example, holiday special editions for an app can be applied only for a limited time and removed for a subsequent update. Third, mobile app developers should constantly plan ahead and release new versions of the application to attract new users and keep existing users (Greer and Ruhe, 2004). Meeting these constraints is not an easy task.

3.2.2 Moderators of Software Maintenance Effectiveness

The importance of a well-planned maintenance schedule is well recognized in physical goods manufacturing. Proper maintenance will help improve the lifetime of the equipment and avoid any unplanned maintenance tasks. The concept of preventive maintenance has proven that keeping a regular schedule of performing maintenance tasks even without system failures can (1) decrease equipment downtime and number of major repairs needed, (2) conserve assets in better conditions and eliminate premature replacement of machinery and equipment, (3) reduce overtime costs and maintenance labor cost by working on a scheduled basis instead of a crash basis, and (4) improve safety and quality conditions (Krajewski, Ritzman, and Malhotra, 2018). These benefits are obtained because the workers who are directly impacted by the maintenance tasks can prepare for the upcoming maintenance schedule beforehand and organize their tasks to minimize the loss from the disruption (Jonsson, 2000).

The importance of keeping regular maintenance schedules are greater for products and services where product failure and service downtime impacts the users' experience directly. We argue that the sense of predictability created from the regular schedules can help the users of a product/service in a similar fashion. From a users' perspective, preventive maintenance allows users to prepare for upcoming disruptions and organize personal activities to minimize the impact. S. Garg et al., 1998 endorses the adoption of preventive maintenance for software systems because the demand for high reliability and availability is escalating. Therefore, failure in software systems can arise from both unreliable performance and lack of content availability. While majority of software maintenance literature focuses on sustaining the stability and reliability of the system, failures from content availability is mostly overlooked. For enhancement updates, the aim of preventive software maintenance should be to provide new features and contents on a regular schedule to prevent users from depleting the content and churn.

In the context of mobile apps, an app developer that keeps a regular update schedule would divide the app lifetime into fixed-term maintenance sprints that typically has a duration of two or three weeks. The content of each maintenance is determined by considering user expectations, business value and implementation risks. An example is the Facebook app with the most recent version number reaching 214.0 by March 28; Facebook has consistently introduced updates around every 1-2 weeks with few exceptions. Most of these updates include generic descriptions about the updates being made as follows. "We update the app regularly so we can make it better for you. Get the latest version for all of the available Facebook features. This version includes several bug fixes and performance improvements."

However, despite the benefits of maintaining a regular fixed schedule, there may exist other factors that influence the update schedules. Instead of maintaining a fixed maintenance schedule, firms can focus their enhancement efforts on specific points in the app's life cycle to maximize returns. This demand sensing can be made by observing the trajectory of their app's performance or by acquiring information about the overall market dynamics.

Observance of the app's adoption trends over the lifecycle may provide useful information for formulating strategies including timing of enhancement updates (Anderson and C. P. Zeithaml, 1984). Prior literature states that the most fundamental variable in determining an appropriate business strategy is the stage of the product lifecycle (Hofer, 1975). Literature mentions four stages that consists a product's lifecycle, namely, introduction, growth, maturity, and decline. However, for digital products such as mobile apps, adoption occurs instantaneously such that there is no clear distinction between introduction and growth stages. Also, once peaked in adoption, apps have difficulty in sustaining the user base because of the short attention span of mobile users. Therefore, once an app reaches the peak in adoption, the constant decline in adoption follows right after. Relying on these observations, we

define the lifecycle stages of a mobile app as three stages comprised of stage1 (launch to initial peak in adoption), stage2 (initial peak in adoption to 50% retention), and stage3 (50% peak in adoption to termination). We included a hypothetical second stage in a seemingly two-stage lifecycle curve because the initial peak in adoption may serve as a strong cue for developers to deploy aggressive user acquisition and retention strategies. By investigating the effect of enhancement efforts, especially at this stage, may provide additional insights on the effectiveness of efforts deployed in this critical period in the app lifecycle.

Moreover, the level of overall market dynamics may be useful in timing the enhancement updates as well. Here, we define market activity as the aggregate download for the entire app subcategory. This market activity may have predictable surges due to seasonal in-flow of users in the market. Prior literature acknowledges that seasonality increases the potential market size, which is generally considered positive, especially for products with extremely short lifecycles (Calantone et al., 2010; Krider and Weinberg, 1998). Although not all seasons are predictable, there may exist certain factors that create on-and-off seasons for mobile apps such as holidays, government actions, industry traditions, weather, social phenomena, summer and school years. Additionally, industry reports suggest that seasonal customers, those who use extra available time on apps, drive seasonal demand for mobile apps (Liftoff, 2017; Liss, 2017). Exploiting this predictable surge in demand, many app developers launch holiday specific theme packs, merchandise, and in-app events to attract the additional seasonal demand.

These could be factors that perhaps a firm may further enhance the update effectiveness despite the disruption in update schedule regularities.

3.3 Sample Construction and Data Description

In this section, we discuss the sample construction process of our data and describe the variables included in the analysis.

3.3.1 Sample Construction

The data used for this study comes from a proprietary Application Programming Interface (API) server of a leading app store intelligence company. Not only does the firm maintain daily observations on the app, publisher, and market performance metrics, but they also collect information on version history, update date, and update log text files for the apps contained in the database. We compile a sample of U.S. iOS mobile gaming apps starting from January 2015 to December 2017. After excluding apps that have missing observations or gaps in the time series, we obtained a sample of 4,129 apps which contains information on 42,772 updates. The data extraction was performed using the university's high-performance cluster computing server (HPC) for parallel processing of extraction and database manipulations. Certain metrics included in our empirical model such as the standard deviation of inter-update times requires each app to have at least three or more updates. After dropping apps that have less than three updates, we are left with 2,049 apps. Further, we run a Bass model non-linear OLS estimation on each app time series. This further dropped apps that failed to converge. After these data manipulations, we obtained a final sample of 433 apps with 4,052 updates.

The mobile gaming app category serves as a nice research setting because gaming apps are known to have the shortest average life cycle among all mobile app categories and have the highest update frequency. With our limited three-year observation period, mobile games allow us to study the update effects in varying frequencies across the entire life cycle for a fair amount of apps within our sample.

3.3.2 Variable Description

The unit of analysis in this study is individual mobile app updates. Therefore, for each variable description, we denote subscripts i to denote the updates, j for apps, k for developer, c for subcategory, and t for day. The gaming app represents a mobile app category and subcategories are the 18 game genres such as strategy, role-playing, action, shooting, family, puzzle, etc. The list of variables is also summarized in Table 3.1.

Dependent Variable

The dependent variable $lndown_{ijt}$ is the natural logged number of daily downloads for update i in-app j at time t. The number of downloads is highly correlated with the user base size in the app. Revenue generation for freemium apps through in-app purchase and in-app advertising also rely on the number of users in the app, which makes the number of downloads a fine measure to gauge an app's performance.

Independent Variables of Interest

We rely on a difference-in-differences (DID) design to estimate the effect of content updates. The DID estimator is constructed as an interaction between a set of binary indicators to denote the before the update and after update periods ($after_t$), and to denote the treatment and control group association of the observation ($treat_j$).

Once we have estimated the effect of content updates on the app's downloads, we further interact the DID estimator with a set of moderators to estimate the effectiveness of certain updating contexts. To estimate the enhancement update effect while keeping a regular schedule, we include the rolling standard deviation of inter-update times ($upsd_{ijt}$). As the number of updates accumulates and new pieces of inter-update time information is added, the standard deviation is recalculated based on the additional observation. One caveat is that we have to limit our sample to

Table 3.1: List of Variables Used in the Model

Variable	Description
Dependent Variable	
$lndown_{ijt}$	Natural logged number of daily downloads for update i in app j at time t
Independent Variables	
$after_t$	Binary variable coded as 0 for before update periods (i.e., 2 days) and 1 for after update periods (i.e., 3 days)
$treat_j$	Binary variable coded as 0 for control group and 1 for treatment group
$upsd_{ijt}$	Rolling standard deviation of inter-update times updated at each update i and held constant until the next update
$LCSTAGE_{ijt}$	A set of dummy variables indicating the life cycle stages (stage1: launch-initial peak in downloads, stage2: initial peak-50% retention, stage3: 50% retention-service termination
$mkttr_{ct}$	Natural logged total number of app daily downloads for all the apps in mobile game subcategory c at time t
hhi_{ct}	Sum of squared market shares (i.e., daily download shares) in all apps within a mobile game subcategory c at time t
$rating_{jt}$	Average of five star user rating of app j until time t
$sincelaunch_{ijt}$	The number of days passed for update i at time t
$weekday_t$	A set of dummy variables indicating the day in a week (0: Sunday - 6: Saturday)
$month_t$	A set of dummy variables indicating the month in a year (0: January - 11: December)

apps that have introduced at least three updates or more to be able to calculate this metric. To estimate the enhancement update effects at different lifecycle stages, we include a vector of dummy variables denoting the life cycle stage around the time of the update ($LCSTAGE_{ijt}$). We define three life cycle stages of an app — stage one from launch date to date of initial peak in daily downloads, stage two from date of initial peak in daily downloads to date when the daily downloads decrease to 50% of the initial peak, and stage three from date of 50% of the initial peak to the date of app service termination. In our sample, the developer removes the app from the app market when the service is terminated, and this results as a dropout at the end of the time series in our dataset. To estimate the enhancement update effect at varying market activity levels, we include the aggregate logged daily downloads in the entire app subcategory ($mkttr_{ct}$).

Control Variables

For controls, we include the Herfindahl-Hirschman Index (hhi_{ct}) of a gaming app subcategory to control for the competition intensity in the subcategory. We include the average five-star rating score for the app at time t ($rating_{jt}$). This variable partially controls for any unobserved changes other than the content update in the quality of the app, which may in turn influence the number of downloads. We include the count of days since the app was launched in the market up to the day the content update is being made ($sincelaunch_{ijt}$). This partially controls for the maturity of the app, which may affect the users' degree of familiarity of the contents and changes in an update and the developer's degree of experience or stability of the system through managing the app over time. We also include time fixed effects as dummy variables to denote the day of the week ($weekday_t$) and the month in a year ($month_t$). These time fixed effects control for unobserved time correlated shocks in the app market such as seasonal behaviors. Table 3.2 shows the summary statistics and correlations

of the key variables used in our model.

3.4 Econometric Model

In this research, we assess (1) the effect of a content update in a mobile app and (2) the moderation effects of three different update contextual factors on the content update. To make an unbiased estimation, we need to benchmark the changes in the app's performance against a control group which did not introduce a content update. To this end, we formulate a DID model to estimate the effects of interest. While constructing a DID sample, we face challenges in (1) identifying and classifying content updates and (2) constructing the control group.

3.4.1 Classifying Content Updates

In software development, developers follow a convention of numbering the versions of the software using semantic numbering. Semantic numbering refers to a three-digit numbering system (i.e., Major.Minor.Patch) that developers use to classify the amount of content being changed to the original code based on the degree of risk and number of function points. However, many developers happen to use their numbering system rather than following this convention. Therefore, a simple classification based on the version numbering may not be able to categorize all the updates in our sample. Among the 42,772 updates in the initial dataset, 24,250 updates are major content updates, 15,383 updates are minor bug fixes and patches, and 3,139 updates are unclassified. To classify the updates into content updates and bug-fix/patches, we deploy a supervised machine learning algorithm to analyze the update logs and automatically classify each update. We rely on the scikit-learn library in python to develop a machine learning-based classification model. This simple approach is sufficient in classifying the updates as the text information mostly consists of direct information without sentiments. To maximize the accuracy of the classification al-

Table 3.2: Descriptive Statistics and Correlation Matrix

	Variable	μ	σ	Min	Max	1	2	3	4	5	6	7	8	9	10
1	lndown	4.54	2.66	0.00	11.96										
2	after	0.60	0.49	0.00	1.00	0.00									
3	treat	0.50	0.50	0.00	1.00	**0.26**	0.00								
4	upsd	13.33	22.11	0.00	225.74	**-0.04**	**0.07**	0.00							
5	hhi	0.05	0.08	0.00	1.00	**0.03**	**-0.01**	0.00	**-0.05**						
6	rating	4.36	0.50	2.00	5.00	**0.08**	0.00	0.00	**0.04**	0.00					
7	mkttr	11.17	1.21	3.50	13.50	**0.22**	0.01	0.00	**0.04**	**-0.30**	**-0.02**				
8	sincelaunch	209.64	162.89	1.00	841.00	**-0.14**	0.01	0.00	**0.41**	**-0.15**	**0.06**	**0.06**			
9	lcstage1	0.32	0.47	0.00	1.00	**-0.03**	-0.01	0.00	**-0.11**	**0.11**	**0.06**	**-0.14**	**-0.35**		
10	lcstage2	0.06	0.24	0.00	1.00	**0.14**	0.00	0.00	**-0.04**	**-0.02**	0.01	**0.11**	**-0.10**	**-0.17**	
11	lcstage3	0.62	0.49	0.00	1.00	**-0.04**	0.01	0.00	**0.13**	**-0.09**	**-0.06**	**0.08**	**0.38**	**-0.88**	**-0.32**

Note. Bold denotes significance at $p < .05$

gorithm, we apply the following steps to the original dataset that contains 101,059 sentences. The update logs contain an average of 2.4 sentences. Considering that prior text analysis tasks required a minimum of 3,000 to 6,000 sentences, we have a sufficient number of text data for the classification algorithm. First, we pre-process the text data included in the update logs. We separate all the words within the text (tokenization), and then set all the words in lower case. Next, we shorten the words into their root stems (lemmatization/stemming). Then, we remove English stop words (e.g., a the, and, not, in, on, etc.) and punctuation. For the list of English stop words, we rely on a list retrieved from python's natural language toolkit library (NLTK) (http://www.nltk.org) which is commonly used in text mining tasks. Second, we construct a vector of word counts for each update log (count vectorization). Third, we apply a weight to each word using the text frequency-inverse document frequency (tf-idf) approach. This weighting scheme is applied to over 83% of text-based recommendation systems in digital libraries. For term i in update log j, the tf-idf weight $w_{i,j}$ is calculated as follows.

$$w_{i,j} = f_{i,j} \times log(\frac{N}{g_i}) \tag{3.1}$$

Where, $f_{i,j}$ is the number of occurrences of term i in update log j, N is the total number of update logs, and g_i is the number of documents that contain the term i. The inverse document frequency is a measure of how much information each word provides. If a term frequently appears across all documents, then the term does not provide much information in classifying the documents. Next, we split the data that follows the semantic version numbering scheme into a training dataset and test dataset in a ratio of 70:30. We fit a naive Bayes classification model to predict the classification outcomes. The classification algorithm is based on Bayes' Theorem assuming that predictors are independent. Consider a document classification problem, where a document is broken down into a bag of words (Harris, 1954) where the probability that the ith word of a given document occurs in a document class C as

follows.

$$p(w_i \mid C) \tag{3.2}$$

Then the probability that a given document D contains all of the words w_i, given a class C is

$$p(D \mid C) = \prod_i p(w_i \mid C) \tag{3.3}$$

According to Bayes Theorem,

$$p(C \mid D) = \frac{p(C)p(D \mid C)}{p(D)} \tag{3.4}$$

In our case, we have two mutually exclusive document classes, E and $\neg E$ (i.e., Enhancement update and maintenance update). Such that $p(E \mid D) + p(\neg E \mid D) = 1$. Then the probability that a document D containing all of the words w_i, given a class E and $\neg E$ is

$$p(D \mid E) = \prod_i p(w_i \mid E) \tag{3.5}$$

$$p(D \mid \neg E) = \prod_i p(w_i \mid \neg E) \tag{3.6}$$

The Bayes' Theorem yields the a statement of probability in terms of likelihood as follows.

$$p(E \mid D) = \frac{p(E)}{p(D)} \prod_i p(w_i \mid E) \tag{3.7}$$

$$p(\neg E \mid D) = \frac{p(\neg E)}{p(d)} \prod_i p(w_i \mid \neg E) \tag{3.8}$$

Dividing Equation 3.7 by 3.8 gives a likelihood ratio as follows.

$$\frac{p(E \mid D)}{p(\neg E \mid D)} = \frac{P(E)}{p(\neg E)} \prod_i \frac{p(w_i \mid E)}{p(w_i \mid \neg E)} \tag{3.9}$$

Taking the logarithm of Equation 3.9 yields the log-likelihood ratio.

$$\ln \frac{p(E \mid D)}{p(\neg E \mid D)} = \ln \frac{P(E)}{p(\neg E)} + \sum_i \ln \frac{p(w_i \mid E)}{p(w_i \mid \neg E)} \tag{3.10}$$

The document of update logs can be classified as enhancement updates if $p(E \mid D) > p(\neg E \mid D)$ or $\ln \frac{p(E|D)}{p(w_i|\neg E} > 0$ and maintenance updates otherwise.

The Naive Bayes classifier is easy and fast to implement, and it is also capable in dealing with multiple classes (i.e., $C > 2$). When the assumption of independence holds, it is know to perform better than other models with less training data. For numerical predictors such as counts of w_i, the model assumes normal distribution of the variables. As the model assumption of independence and normal distribution of predictors can be a strong assumption, we test for alternative models and compare the accuracy. A 5-fold cross validation result yielded a 73% accuracy for the native Bayes classification model. We chose this model for unclassified update predictions because it yielded the highest accuracy compared to alternative models such as logistic regression (70%), support vector machines (68%), and K-means clustering (65%). The final classified sample resulted in 26,349 major updates and 16,423 minor bug fix and patches. We merged this version history dataset with the app performance dataset to obtain our final sample.

3.4.2 Control Group Construction

The content updates in our sample occur at different time points, and having multiple updates within a single app makes identification of a proper control group difficult. To overcome this issue, we rely on the event study literature stream which estimates abnormal returns of corporate events on stock price time series by extrapolating the time series using a linear prediction. To predict the adoption patterns of mobile apps, we rely on the Bass diffusion model (reference) which is formulated as follows (Bass, 1969).

$$F(t) = \frac{1 - e^{-(p+q)t}}{1 + \frac{q}{p} e^{-(p+q)t}} \tag{3.11}$$

where $F(t)$ is the portion of the potential market that has adopted the mobile app by time t, p is the coefficient of innovation, and q is the coefficient of imitation. For

each app, we fit the Bass model curve to the daily download time series and obtain these parameters. We then use the fitted model to predict the daily downloads after the content update. The choice of the Bass model to construct the baseline prediction is based on three reasons. First, we do not have an a priory theoretical causal model to explain the trajectory of mobile app downloads for freemium apps. Second, due to the unique non-linear shape of the mobile app adoption pattern, we find that the Bass model shows highest fit to observed data points compared to other time series methods such as moving averages or exponential smoothing techniques. Third, the Bass model contains parameters that represent innovators and imitator user segments, which represent the importance of network effects and the essence of why people download mobile apps. Finally, we can construct the control group observations from the identical observation and therefore, satisfy the parallel path assumption during pre-treatment periods(Autor, 2003). We recover the parameter values from the time series using a non-linear OLS estimation. In order to reduce the computation time, we split the data into 14 partitions of 150 apps and ran 14 parallel processors on the high-performance cluster computing (HPC) server. During this process, some app time series fail to converge within 1,000 iterations and therefore, are dropped from our subsequent analyses. However, for the apps that converged, we obtained an average R^2 of 0.995, which shows an overall good fit of the model (Figure 3.1). After excluding the apps that the Bass model estimation failed to converge, we are left with a final sample size of 433 apps with 4,052 updates.

3.4.3 Difference-In-Differences Analysis

Following the analysis in event study models (Hendricks and Singhal, 2005), we limit our observation sample to two days before the update and three days after the update including the day of the update to reduce concerns of unobserved confounding events in the app. Therefore, for each update, we have a total of 5 days. To assess the effect

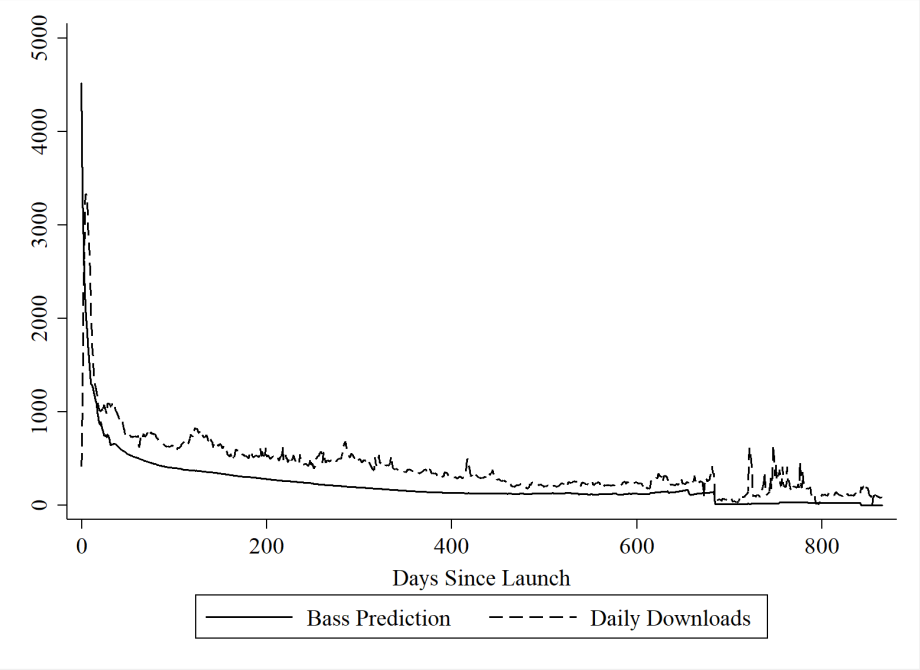


Figure 3.1: Average Logged Daily Downloads with Bass Model Predictions

of content updates on the performance of mobile apps, we formulate a difference-in-differences regression, where the main dependent variable is the performance measure $lndown_{ijt}$. The independent variables are the binary treatment indicator $treat_j$, the binary indicator for the post treatment periods $after_t$, and the interaction term of these two indicators. In addition, we include control covariates to account for unobserved firm, app, and market characteristics. More specifically, we estimate the following regression for update i, app j, at time t.

$$lndown_{ijt} = \beta_0 + \beta_1 after_t + \beta_2 treat \cdot after_{ijt} + \delta Z_{ijt} + \tau_m + \tau_d + \alpha_i + \epsilon_{ijt} \quad (3.12)$$

where the coefficient β_2 of the interaction term is our key DID estimator of interest. The main effect of $treat_j$ is dropped from the model because it cannot be identified when update fixed effects α_i are included in the model. Z_{ijt} is a vector of control covariates (i.e., hhi_{ct}, $rating_{jt}$, and $sincelaunch_{ijt}$). τ_m and τ_d are sets of the month and weekday time fixed effects that control for seasonality. We estimate the model with standard errors clustered by apps.

The identification of the treatment effect depends on the satisfaction of the stable unit treatment value assumptions (SUTVA) (Imbens and Rubin, 2017). SUTVA consists of two components. First, the treatment applied to one unit does not affect the outcome of another unit, and second, there is only a single version of each treatment level. For mobile app updates, each update event is contained within the update. Even for competition effects, this may not affect the downloads of another app as users can download multiple apps and keep them in their devices. Regarding the possibility of spillovers from prior updates, content updates take place when the developer thinks that the users have depleted the current stock of contents within the app. Also, we specify a short observation window for the before and after updates which are even shorter than an aggressive content update cycle of one week. Therefore, we believe that spillovers from other treatments are not a concern for content updates. Next, regarding the consistency of the treatment effects, we have deployed a machine learning classification model to isolate out the major content updates and minor bug fixes and patches. Further, we account for update and app fixed effects and rely on a within-transformation to estimate the content update effects. Although there may be some variations in the number of content updates, the essence of the content update representing major changes to the code and imposing a risk to the application remains consistent across all content updates within a single app. This is a reasonable assumption as developers have to manage the enhancement content development within constrained resources and tight release schedules. Also, because the app performance needs to be optimized with consistent app size, there is a limit to the extent of changes that can be made in a single enhancement update.

3.4.4 Update Strategies

Upon estimating the content update effect via DID regression, we proceed to estimating the moderating effects of the proposed update strategies. First, we estimate the

effect of a enhancement under schedule regularity by interacting $upsd_{ijt}$, the rolling standard deviation of inter-update times with the DID estimator. Specifically, we estimate the following regression for update i, app j, at time t.

$$\begin{aligned} lndown_{ijt} =& \beta_0 + \beta_1 after_t + \beta_2 after \cdot upsd_{ijt} + \beta_3 treat \cdot upsd_{ijt} \\ &+ \beta_4 treat \cdot after_{ijt} + \beta_5 after \cdot treat \cdot upsd_{ijt} \\ &+ \delta Z_{ijt} + \tau_m + \tau_d + \alpha_i + \epsilon_{ijt} \end{aligned} \tag{3.13}$$

where the coefficient of the three-way interaction β_5 is our estimator of interest. The first order term of $upsd_{ijt}$ is added to the vector of control covariates Z_{ijt}.

To estimate the moderating effects of lifecycle stages, we incorporate a vector of life cycle stage binary indicator variables as $LCSTAGE_{ijt}$, and interact it with the DID estimator. Specifically, we estimate the following regression for update i, app j, at time t.

$$\begin{aligned} lndown_{ijt} =& \beta_0 + \beta_1 after_t + \beta_2 after \cdot LCSTAGE_{ijt} \\ &+ \beta_3 treat \cdot LCSTAGE_{ijt} + \beta_4 treat \cdot after_{ijt} \\ &+ \beta_5 after \cdot treat \cdot LCSTAGE_{ijt} \\ &+ \delta Z_{ijt} + \tau_m + \tau_d + \alpha_i + \epsilon_{ijt} \end{aligned} \tag{3.14}$$

where the coefficient of the three-way interaction β_5 is our estimator of interest. The first order term of $LCSTAGE_{ijt}$ is added to the vector of control covariates Z_{ijt}.

To estimate the moderating effects of market activity levels, we introduce $mkttr_{ct}$, the total daily downloads in a mobile game genre subcategory and the interaction between the DID estimator. Similarly, we estimate the following regression for update i, app j, at time t.

$$\begin{aligned} lndown_{ijt} =& \beta_0 + \beta_1 after_t + \beta_2 after \cdot mkttr_{ct} + \beta_3 treat \cdot mkttr_{ijt} \\ &+ \beta_4 treat \cdot after_{ijt} + \beta_5 after \cdot treat \cdot mkttr_{ijt} \\ &+ \delta Z_{ijt} + \tau_m + \tau_d + \alpha_i + \epsilon_{ijt} \end{aligned} \tag{3.15}$$

where the coefficient of the three-way interaction β_5 is our estimator of interest. The first order term of $mkttr_{ct}$ is added to the vector of control covariates Z_{ijt}.

3.5 Empirical Results

3.5.1 Difference-in-Differences Analysis

In this section, we summarize and report our findings as shown in Table 3.3.

First, we report the estimates of Equation 3.12 in model (1), Table 3.3. We find that updating a mobile app has a significant and positive impact on logged daily downloads ($\beta = 0.051, p < 0.01$). This coefficient shows that introducing a content update leads to an average of 5.23% ($(e^{0.051} - 1) \times 100$) increase in daily downloads. Second, we report the estimates of Equation 3.13 in model (2). We find that the standard deviation of inter-update times has a negative and significant interaction with the DID estimator ($\beta = -0.002, p < 0.05$). The elasticity of the standard deviation of inter-update time shows that a 1% increase in the inter-update time standard deviation leads to a 1.96 percentage point decrease in content update effect. Third, we report the estimates of Equation 3.14 in model (3). We find multiple significant interactions between the DID estimators and the life cycle stage variables, both for two-way and three-way interactions. We first calculate the marginal effects of content updates made at the second stage and third stage compared to the base level of content updates made in the first stage of the life cycle. We find that the marginal effect of a content update made in the second stage of the life cycle leads to a 44.7% increase in logged daily downloads compared to content updates made in the first stage. Also, content updates made in the third stage of the life cycle leads to a 68.0% decrease in logged daily downloads compared to content updates made in the first stage. Finally, we report the estimates of Equation 3.15 in model (4). We find a positive significant interaction effect of market activity on content updates ($\beta = 0.061, p < 0.01$). The calculated elasticity shows that a 1% increase in market

Table 3.3: Difference-in-Differences Estimation Results

DV: $lndown_{ijt}$	(1)	(2)	(3)	(4)
$after_t$	−0.011	−0.019	0.017	−0.001
	(0.046)	(0.046)	(0.046)	(0.059)
$after_t \times treat_{ij}$	0.051 * **	0.076 * **	0.073 * **	−0.640 * **
	(0.019)	(0.022)	(0.025)	(0.195)
$upsd_{ijt}$	−0.000	0.000	−0.000	−0.000
	(0.000)	(0.000)	(0.000)	(0.000)
hhi_{ct}	−0.140	−0.140	−0.141	−0.149*
	(0.087)	(0.087)	(0.087)	(0.089)
$upcount_{ijt}$	−0.029	−0.030	−0.038	−0.042
	(0.045)	(0.045)	(0.045)	(0.055)
$rating_{jt}$	−0.160*	−0.160*	−0.162*	−0.156*
	(0.085)	(0.085)	(0.084)	(0.085)
$mkttr_{ct}$	0.422 * **	0.422 * **	0.422 * **	−0.001
	(0.025)	(0.025)	(0.025)	(0.003)
$sincelaunch_{ijt}$	0.002	0.002	0.003	0.003
	(0.004)	(0.004)	(0.004)	(0.004)
$lcstage2_{ijt}$	0.136*	0.136*	−0.254 * *	0.118
	(0.082)	(0.082)	(0.107)	(0.080)
$lcstage3_{ijt}$	−0.229 * *	−0.228 * *	0.231 * **	−0.243 * *
	(0.098)	(0.098)	(0.087)	(0.098)
$after_t \times upsd_{ijt}$		0.001		
		(0.000)		
$treat_{ij} \times upsd_{ijt}$		−0.001		
		(0.001)		
$after_t \times treat_{ij} \times upsd_{ijt}$		−0.002 * *		
		(0.001)		
$after_t \times lcstage2_{ijt}$			−0.013	
			(0.013)	
$after_t \times lcstage3_{ijt}$			−0.037 * **	
			(0.012)	
$treat_{ij} \times lcstage2_{ijt}$			0.871 * **	
			(0.273)	
$treat_{ij} \times lcstage3_{ijt}$			−0.868 * **	
			(0.257)	
$after_t \times treat_{ij} \times lcstage2_{ijt}$			−0.157 * **	
			(0.041)	
$after_t \times treat_{ij} \times lcstage3_{ijt}$			−0.005	
			(0.038)	
$after_t \times mkttr_{ct}$				0.001
				(0.002)
$treat_{ij} \times mkttr_{ct}$				0.814 * **
				(0.049)
$after_t \times treat_{ij} \times mkttr_{ct}$				0.061 * **
				(0.017)
Constant	0.470	0.474	0.523	0.704
	(0.891)	(0.893)	(0.889)	(0.926)
Update FE	YES	YES	YES	YES
Month FE	YES	YES	YES	YES
Weekday FE	YES	YES	YES	YES
Observations	20,254	20,254	20,254	20,254
R-squared	0.207	0.207	0.215	0.411
Number of updateid	4,052	4,052	4,052	4,052

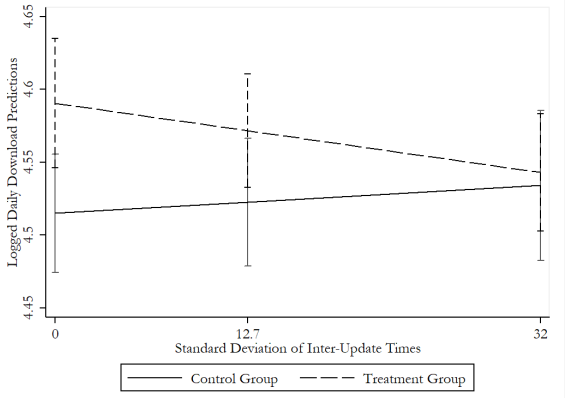
Robust standard errors clustered by app in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

activity increases the content update effect by 87.4%.

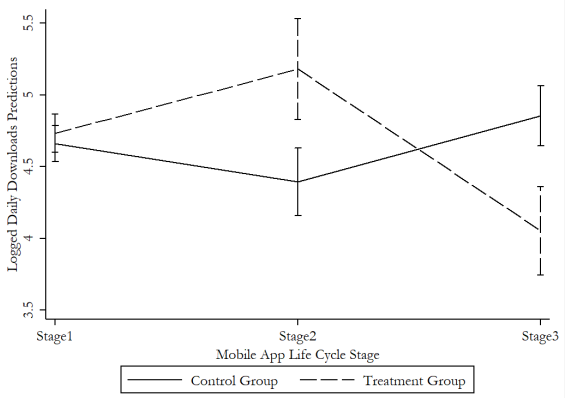
We also plot the predicted levels of the logged daily downloads by treatment / control group (See Figure 3.2). We predict the outcome levels at mean and ±1 standard deviation values (i.e., High/Low) for the continuous variables, $upsd_{ijt}$, $mkttr_{ct}$, and the three stages for the discrete $LCSTAGE_{ijt}$ variable. As we can see in Figure 3.2(a), at low levels of the inter-update time standard deviation, the update effect creates significant differences in daily downloads ($\chi^2 = 12.28, p < 0.01$). While this difference is still significant at mean levels of the inter-update time standard deviation, the gap closes as update regularity decreases and is no longer significant at high levels of the inter-update time standard deviation ($\chi^2 = 0.09, p > 0.10$). This shows that the benefits of content updates can be nullified at high levels of irregular update times. Regarding updates at different life cycle stages (Figure 3.2(b)), we find that content updates made in the second stage show the strongest impact on the app's performance compared to the control group performance ($\chi^2 = 9.18, p < 0.01$). We also find that the content update effect does not yield significant differences in the first stage of the life cycle. Moreover, interestingly, we find that a content update made in the third stage decreases the performance of the app compared to the control group ($\chi^2 = 9.79, p < 0.01$). Finally, in Figure 3.2(c), we find that content updates generates a significant positive boost for the app's performance at all levels of market activities (Low-Mean-High). However, as the level of market activity increases, the content update effect increases proportionally as well.

3.5.2 Post-hoc Analysis

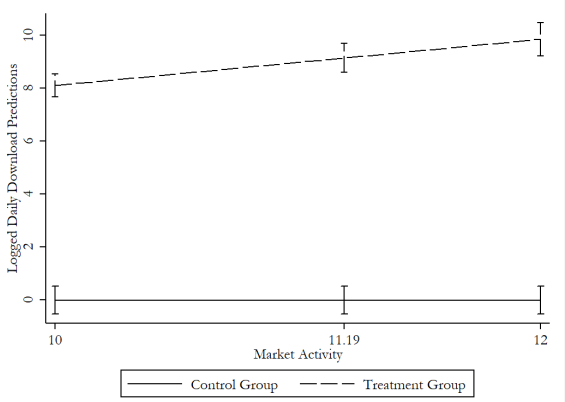
In this section, we answer additional questions that we have identified during the main analysis. Specifically, by looking at the average inter-update times across all the apps in our sample, we see that the average update interval is around one month (28 days). Even for apps that aggressively introduce content updates, they require at



(a) after×treat×upsd



(b) after×treat×LCSTAGE



(c) after×treat×mkttr

Figure 3.2: Logged Daily Download Predictions by Treatment

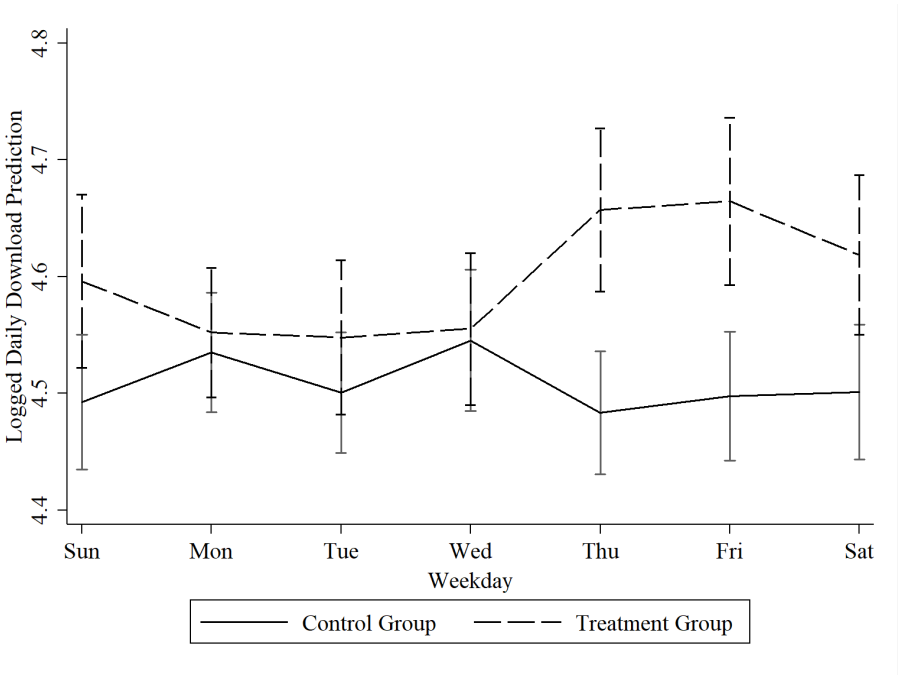


Figure 3.3: Logged Daily Download Predictions by Treatment×Weekday

least one week to bundle new contents together and implement them in the currently serviced app. Therefore, for a developer, it would be interesting to know in that inter-update time, when the best time would be to introduce an update. To answer this question, we take a look at a single week and identify the weekday which is the best time to introduce an update. Our estimation is done by interacting the update DID variable with the weekday fixed effects. The results are summarized in Table 3.4, and the linear predictions are plotted in Figure 3.3.

As we can see from the results in Table 3.4 and Figure 3.3, compared with the control group, the treatment group experiences the content update effect in varying degrees across the days in a week. Especially, there is a significant boost in the update effect, especially when the content update is introduced in either Thursday or Friday. The update effect becomes non-existent when it is introduced on Monday through Wednesday. When a developer releases enhancements, it is therefore also important to take the day of the week into consideration to further enhance the benefits from a content update.

Table 3.4: Content Update and Weekday Interaction Analysis

DV: $lndown_{ijt}$	Coefficient	Standard Error
$after_t$	-0.013	(0.058)
$after_t \times treat_{ij}$	$0.104**$	(0.044)
$weekday(Mon)_t$	0.010	(0.012)
$weekday(Tue)_t$	$-0.039**$	(0.014)
$weekday(Wed)_t$	-0.001	(0.019)
$weekday(Thu)_t$	$-0.077**$	(0.019)
$weekday(Fri)_t$	$-0.024**$	(0.012)
$weekday(Sat)_t$	-0.005	(0.009)
$after_t \times weekday(Mon)_t$	0.033	(0.021)
$after_t \times weekday(Tue)_t$	$0.047**$	(0.022)
$after_t \times weekday(Wed)_t$	$0.053**$	(0.025)
$after_t \times weekday(Thu)_t$	$0.068**$	(0.023)
$after_t \times weekday(Fri)_t$	$0.029*$	(0.016)
$after_t \times weekday(Sat)_t$	0.014	(0.012)
$treat_{ij} \times weekday(Mon)_t$	0.007	(0.033)
$treat_{ij} \times weekday(Tue)_t$	0.057	(0.043)
$treat_{ij} \times weekday(Wed)_t$	0.063	(0.051)
$treat_{ij} \times weekday(Thu)_t$	$0.193**$	(0.055)
$treat_{ij} \times weekday(Fri)_t$	$0.100**$	(0.045)
$treat_{ij} \times weekday(Sat)_t$	0.031	(0.028)
$after_t \times treat_{ij} \times weekday(Mon)_t$	$-0.093*$	(0.056)
$after_t \times treat_{ij} \times weekday(Tue)_t$	$-0.114*$	(0.065)
$after_t \times treat_{ij} \times weekday(Wed)_t$	$-0.156**$	(0.069)
$after_t \times treat_{ij} \times weekday(Thu)_t$	$-0.123**$	(0.058)
$after_t \times treat_{ij} \times weekday(Fri)_t$	-0.037	(0.056)
$after_t \times treat_{ij} \times weekday(Sat)_t$	-0.017	(0.040)
Constant	0.778	(0.889)
Update FE	YES	
Weekday FE	YES	
Month FE	YES	
Observations	20,254	
R-squared	0.212	
Number of updateid	4,052	

Control covariates suppressed for brevity
Robust standard errors clustered by app in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Next, we focus on the negative effect of enhancement updates in the third stage of the app's lifecycle. One explanation for the negative enhancement update effect in the last stage of the life cycle is that developers changed their update objectives from adding value to the users into revenue extraction. Milking is a common strategy for declining products and services, which allows the firm to secure capital for future investments. Such milking strategies are not explicitly mentioned in update logs. Therefore, we identify a proxy variable that can capture the shift in strategic objectives. A common milking strategy observed in mobile games is introducing various in-app advertisements that can further increase the revenue stream of the developer. However, because these in-app advertisements can disrupt the immersion and flow of the users' experience, developers are cautious in introducing them in early stages of the app's lifecycle. We introduce in-app advertising revenue as an additional variable that proxies the monetization intentions of the developer. We estimate a four-way interaction between the before/after, treatment, lifecycle stage, and adrevenue variable and plot the predicted daily downloads in Figure 3.4. The addition of the advertisement variable allows us to estimate the lifecycle stage effects while holding advertising revenues at the mean. We find that the enhancement update effects for the first and second stage remains consistent with the main model findings. However, the third stage negative impact of enhancement update effect becomes insignificant ($F = 2.12, p > 0.1$). This supports our explanation that third stage updates may be focused on revenue extraction.

3.5.3 Robustness Checks

To demonstrate the robustness of our findings, we (1) address the possibility of an alternative explanation for the lifecycle stage estimation arising from user base scale effect arguments, (2) address the potential serial correlation bias, (3) conduct a placebo test around the before/after period, (4) specify an alternative fixed effects model to

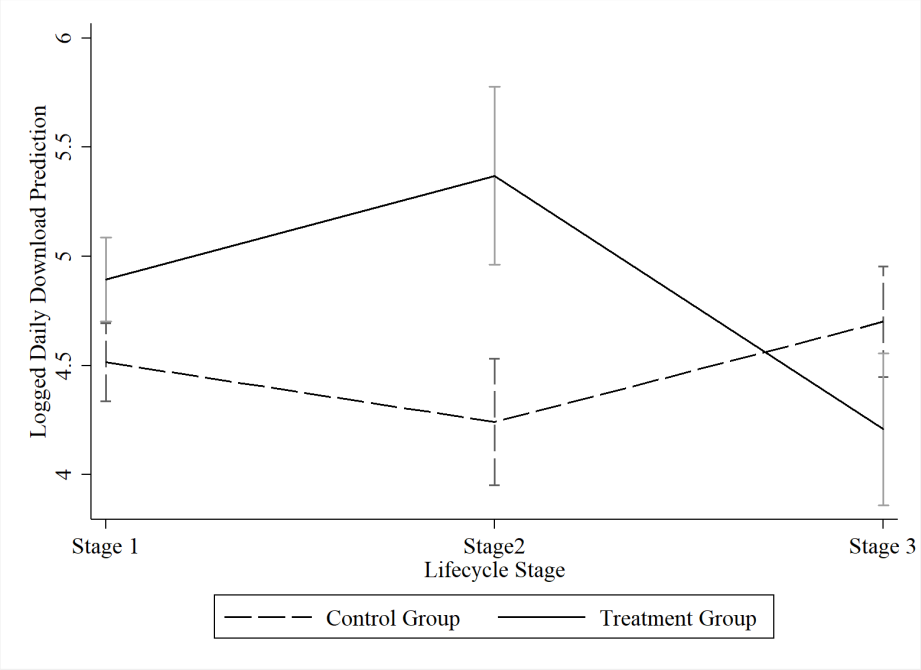


Figure 3.4: Logged Daily Download Predictions with Ad Revenue

deal with concerns of the short time panel, (5) estimate the effect of content updates on an alternative dependent variable using survival analysis, and (6) provide an estimate for a more granular split of the life cycle stages (i.e., 5-stage model). The results of the alternative models are presented in Appendix B.

First, an alternative explanation may exist regarding our findings from the lifecycle stage interaction effect. The enhancement effects can also be stronger because we simply have more users during the second stage. Then the enhancement update reinforcing effect may be driven by the user base scale rather than the lifecycle stage. To see whether this is the case, we estimate a panel quantile regression model with 100 bootstrap replications that does not control for lifecycle stages, but instead, estimates the enhancement update effect at 10 percent quantiles of the daily downloads. The result shows (Figure 3.5) insignificant enhancement update effects across all quantiles which suggests that the significant interaction effect is not driven by the size of the user base.

Second, we address a potential bias that may arise from serial correlation in our

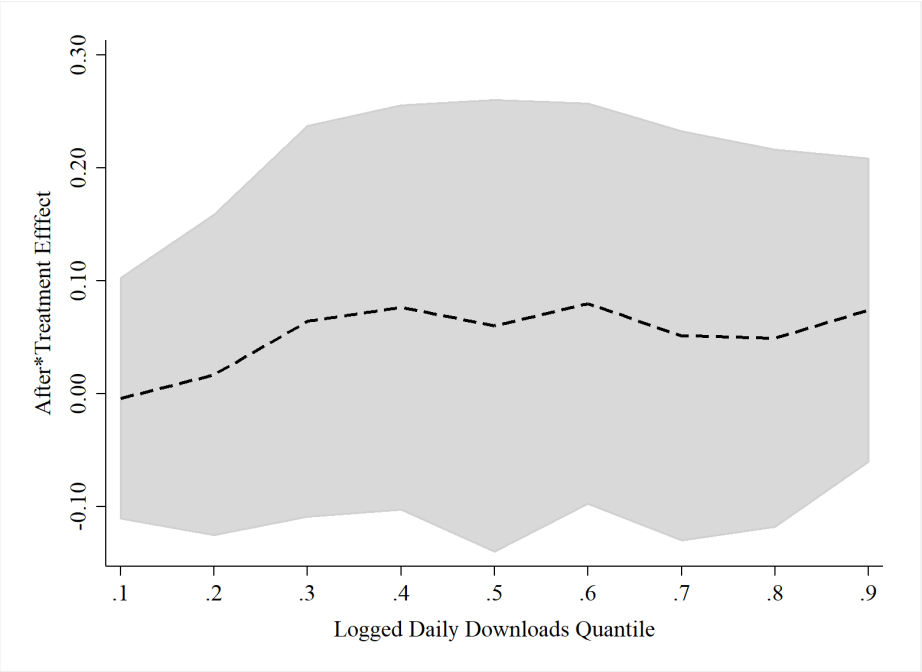


Figure 3.5: Quantile Regression Plot

data. It is known that DID designs are susceptible to serial correlation bias, and one treatment for this matter is to convert the before/after periods into a singular observation by averaging the multiple periods (Bertrand, Duflo, and Mullainathan, 2004). After averaging the before and after periods, we estimate the enhancement update effect and find consistent results which support our main model findings ($\beta = 0.052, p < 0.01$).

Third, to ensure that our DID model is formulated properly with the appropriate control covariates, and is capturing the enhancement update effect at the precise treatment time, we conduct a placebo test. For this test, we falsely assume that the enhancement update effect took place one day before the actual treatment period, and estimate the DID effect with only the two before treatment periods. We find insignificant enhancement update effects which supports the fact that our DID design is not picking up spurious effects ($\beta = 0.004, p > 0.1$).

Fourth, we specify and estimate an alternative fixed effects model using app fixed effects instead of update fixed effects to address potential concerns from Nickell bias

(Nickell, 1981). If the time horizon is short, fixed effect transformations could increase the correlation between autocorrelated variables and induce endogeneity bias to our estimates. Therefore, we formulate an OLS model that controls for app level fixed effects. Compared to the update fixed effects which are derived from a short five-day panel, app level fixed effects are derived from a long time series as we have multiple updates within each app. Estimation results show qualitatively consistent results and support the robustness of our findings.

Fifth, we estimate the effect of enhancement updates on an alternative dependent variable using survival analysis. Instead of the before/after change in daily downloads, we estimate whether the frequency of content updates significantly increase the entire life cycle of an app by reducing the hazard ratio. Since we cannot assume the effect duration of each content update, we operationalize an independent variable that counts the number of content updates that have been made at each lifecycle stage. We then estimate the effect of update frequencies at the respective lifecycle stages, and the effect of update regularity in a Cox Proportional Hazard model. Additionally, we estimate parametric models that assume certain hazard probability distributions with random effects to account for unobserved app heterogeneity. We fit the two most commonly used distributions such as the exponential and Weibull distribution. Findings show qualitatively consistent results as content updates focused in earlier stages in the app life cycle (i.e., first and second stage) can significantly reduce the hazard ratio, and increased variability in inter-update times may significantly increase the hazard ratio. With our main model analysis, these results jointly suggest that early stage content updates with regular schedules can enhance the performance and also extend the life cycle of the mobile app.

Finally, we estimate the effect of enhancement updates at various lifecycle stages using a five-stage life cycle stage specification. Here the five stages are defined as time since launch till the initial peak in adoptions (stage one), the time between the initial

peak till 75th percentile of the peak in daily downloads (stage two), 75th percentile till 50th percentile of the peak in daily downloads (stage three), 50th percentile till 25th percentile in daily downloads (stage four), and 25th percentile till app termination (stage five). Estimation results consistently support our main model findings that content updates in the second stage have the most substantial positive impact on the app's performance, and which declines to a negative impact in later stages in the app life cycle.

3.6 Discussion and Conclusions

To gain a better understanding of enhancement updates, we estimate its effect on app performance measures such as daily downloads and longevity of the app. The estimated effect shows that developers can increase the app's daily downloads on an average of 5.2%. Having an increase in daily downloads throughout the enhancement update can lead to substantial increases in the app user base size in the long-term. While the details of the changes implemented in each enhancement is not available in our data, we assume that these changes are quite comparable between the enhancement updates within the app panel because developers are tightly constrained by their resources and release schedules. Therefore, we believe the estimated update effects are consistent across updates being made within an app. This estimate of the enhancement can serve as important input parameters when developers or future researchers attempt to develop optimized models that jointly consider the software maintenance costs and the demand benefits.

From the estimations and findings in this study, we explore three contextual factors that may influence the effectiveness of the enhancement updates. Findings may prove to be helpful to mobile app developers in further increasing the effectiveness of the updates. The three factors are (1) update schedule regularity, (2) lifecycle stage, and (3) market activity.

3.6.1 Update Schedule Regularity

We estimated the benefits of keeping a regular update schedule from a mobile app user perspective. While the benefits of keeping a regular maintenance schedule may be more prominent on the supply-side in allocating resources and lowering costs, we also find significant positive benefits from the demand-side. When updates are introduced to users regularly, this can minimize the disruptions and frustrations from service downtime. Regular updates can also signal the users that there will be another update to expect in the next update cycle and create a sense of anticipation. Besides direct marketing communications initiated by the developer (i.e., advertisements), another important communication channel is the indirect user-generated content channel where users run individual broadcasts about the app (i.e., youtube, twitch, etc.). These users require contents to show and tell their viewers to maintain the broadcast channels and earn revenue from advertisements. A regular update schedule means that these indirect channels also have less difficulty in finding additional content to broadcast regularly. Because the estimated effect is relatively small compared to the enhancement effect itself, small deviations from the regular schedule may not impact the performance of the app much. However, for significant deviations, which means that the updates are carried out on a more random and sporadic schedule, the accumulated effects may not be negligible.

3.6.2 Lifecycle Stages

Based on our estimates of the enhancement update effects in the three life cycle stages, we find that updates made in the second stage showed strongest impact, followed by the first stage, with the third stage updates showing a negative impact. When compared with the control group, the first stage enhancement updates do not show significant changes in demand. This is because the first stage is when the app is freshly launched into the market. Advertisements about the app are still around the

core features and functions of the app, and users have yet to comprehend and consume all the content that the app provides. The second stage takes place right after the app has reached an initial peak in daily downloads, and the rate of user growth starts to decline. From a Bass diffusion model perspective, the initial peak in adoption tells us that the app has reached its full potential in reaching out to innovators in the market. Innovators are users that decided to adopt the product without peer influences. This segment of users may also serve as influencers that initiate indirect marketing channels and promote apps that provided them with a satisfying experience. In this stage, the core app content may show high rates of completion which means that the app gets promoted and user awareness increases based on additional enhancement updates because core features are no longer appealing to the remaining potential adopters in the market.

One interesting finding is that the update effect becomes significantly negative as the app reaches later stages in the life cycle. One possible explanation is that although the content load and effort are comparable among the enhancement updates, it is possible that the objectives behind the update may have changed in later stages of the app's life cycle. As the downloads and revenue trends show clear signs that the app can no longer appeal to the users, developers may start introducing end-of-life revenue extracting contents in the app which may further accelerate the decay of app adoption. Such end-of-life milking strategies may evoke negative publicity towards the app and deter potential users from downloading the app.

Through robustness checks that rely on a more granular split of the life cycle, these main findings are consistently supported. We can see that the second stage in a five-stage life cycle remains strongly significant, which means that the region that significantly reinforces enhancement updates can be a very tight opportunity window. Therefore, we suggest that publishers should keep a keen eye on the performance trajectory of their apps and make good use of information that the life cycle signals.

Because the moderation effect is relatively larger than the regularity effect, it would make sense for developers to even go out of their regular schedules and focus their maintenance efforts on particular life cycle stages.

3.6.3 Market Activity

The interaction between market activity and enhancement update shows how the effectiveness of enhancement updates change at various market activity levels. The market activity variable captures the general interest of the population towards mobile apps. There may exist predictable spikes in market activity, especially around holiday seasons. Reports show that smart phones and tablets have higher usage rates on weekdays and summer seasons compared to weekends and winter seasons (Waber, 2014). Holidays can also be an important factor, as many publishers increase their ad spending during the holiday periods with eager users installing gaming apps on their new devices received as holiday gifts (Liftoff, 2017). To take advantage of these seasonal demands, many apps introduce holiday-specific events and enhancement updates. For example, a popular farming game "Hay Day" changes the entire theme and background of the app during Christmas every year. The developer changes the background music and adds limited holiday merchandise in the store for users to purchase.

The digital marketplace for mobile apps provides various advantages to the players in the market. One of those advantages is that firms can extract information about the market structure and monitor their competitors for benchmarking. For this reason, currently, there are many market intelligence firms that provide information and consulting services for app developers. Releasing enhancement updates close to high market activity levels would require developers to acquire additional information about the market, and use that information to plan their enhancement update schedules.

3.6.4 Conclusion and Future Research

In this study, we estimate the effect of enhancement update on freemium mobile app download performance and longevity. Additionally, via exploring the moderation effects of update schedule regularity, life cycle stage, and market activity, we show the feasibility of three update strategies.

In interpreting the findings, we acknowledge some limitations. First, the Bass model predictions were obtained using the entire time series of an app. A more rigorous approach would be using only data points preceding the time of update and incorporating a temporal gap between the estimation window and the prediction window. Applying this procedure would drastically reduce our sample because updates with few estimation data points would fail to converge. This reduction in sample size would occur for mostly updates occurring early in the app's life cycle, and would introduce a sample selection bias. We believe our procedure may have generated an upward biased prediction and led to conservative estimates of the enhancement updates. Therefore, in practice, enhancement updates may exhibit larger effects on the app's performance. Second, our sample is limited in terms of app category. However, as previously mentioned, the gaming category is the leading category in terms of revenue generation and technology implementation. Third, while software maintenance is a general task required for all software products, our focus is on mobile apps which relies on the freemium business model. However, the software industry is transitioning to a Software as a Service (SaaS) model which entails other similar revenue generation models such as subscription services and free trials. Therefore, we believe that the applicability of findings from the mobile app context will slowly expand to the software industry over time.

Future research can explore interactions between enhancement update strategies and pricing models for software. This includes in-app purchasing option pricing and other types of software that relies on subscription and physical distribution that

involves pricing decisions. Also, future research can analyze the content of each enhancement update in detail and identify how enhancement updates evolve over the app's lifetime. This can help us understand why specific enhancement updates can lead to adverse outcomes. Finally, future research can investigate the link between enhancement updates and a developer's advertising behavior. This study assumes that the two activities are closely tied, but in practice, there may be cases where the two activities show discrepancies.

Chapter 4

Conclusion

4.1 Introduction

This chapter covers the discussion of the study's results, and expands on its contribution to research and practice. We then present opportunities for future research and conclusions.

4.2 Discussion

This dissertation aimed to understand the interactions between the market, developer, and users surrounding the mobile app industry. Through the studies in Chapter 2 and 3, we identified success strategies for mobile app developers during mobile app development/deployment and operations phase in the software development lifecycle. Specifically, we answered the following research questions.

- What is the impact of feature design on the early stage user base expansion of a mobile app?
- How does the launch timing of the app affect the customers' perceptions toward app feature design?
- Why are some enhancement updates more effective than others?
- How should mobile app developers schedule their enhancement updates?

Results from our study show that when designing mobile apps, offering a focused rich feature set is better than offering a diverse set of features. Specifically, seasonal

users prefer relatively simple apps. Therefore, feature rich and diverse apps may not see much benefits from a market activity-based launch timing strategy. We also estimate the effect of enhancement updates after the app is launched. We find that the enhancement update effect is dependent on the update schedule regularity, life cycle stage, and market activity levels at the time of the update.

4.3 Contributions

4.3.1 Theoretical Implications

Our studies make several theoretical contributions to research streams in product complexity, market seasonality, and software maintenance. First, we estimate the effect of complex product feature designs from a downstream user perspective. Rather than focusing on how complexity imposes product development challenges, we assess whether the choice of adding more layers of complexity to the product is rewarded by the users. Even if complexity adds challenges to the development process, it may be inevitable if users prefer richer and more diverse experience. To do this, we adopt a demand-side performance metric of initial peak magnitude in daily active users, number of downloads, and cumulative downloads. This demand-side assessment is especially important because freemium mobile apps largely rely on network effects and user content generation. We find that feature set composition significantly influences the early stage user base expansion of an app.

Second, we identify product feature richness and diversity as a critical non-price competition factor in the context of mobile applications. While price is often a critical product and market indicator that drives economic theories, factors that are identified in this study provide valuable insight for explaining performance outcome of product systems when they are competing on non-price factors. Optimizing the feature set is especially important in our context as the extremely short product life cycle do not allow firms to experiment and study market reactions after the product

is launched. Therefore, we contribute to the research stream in mobile apps by assessing mobile app performance at the feature-level, which becomes more relevant as freemium emerges as the dominant business model.

Third, we find asymmetric effects of the different dimensions of complexity, which helps us establish the conceptual differences between the dimensions. Prior literature on the demand side assessment of product complexity (Kim and Hyun, 2011; Mikolon et al., 2015; Thompson, Hamilton, and Rust, 2005) mostly view complexity as a unidimensional construct or hypothesize a unified direction of multiple dimensions of complexity, while failing to delineate the differences between each complexity dimension. We conceptualize the two complexity dimensions as feature richness and diversity, and estimate their relative impact on the apps' performance. Findings from this study support a positive effect of complexity when components cohesively contribute to a certain feature, thereby adding richness. On the other hand, when components are scattered across a variety of feature categories, the complexity negatively affects the app's performance.

Fourth, we show how feature complexity of a product interacts with market seasonality. Specifically, we find that apps with less feature richness and diversity benefit more from a peak season launch. This finding helps us explain the heterogeneous performance outcomes of mobile apps competing during peak demand seasons. Research in economics that examines business cycles and marketing literature that examines demand seasonality have endorsed the idea of optimizing market entry timing strategies based on demand patterns. In contrast to the naive belief that more market potential is always good, we argue that it is important to examine the demand seasonality and product feature interactions. In contrast to our initial reasoning that the effect of feature richness and diversity would be more salient to the seasonal users, we find that the complexity of features reduces the attractiveness of the app to the seasonal customers. The additional inflow of users during peak seasons represent a

type of users who are somewhat constrained by resources or attention spans such that product adoption occurs mostly during a specific time of year. Therefore, seasonality in demand itself can occur for a specific type of consumer segment, which in turn requires the app to be exceptionally easy to understand and use. These users do not reward the developers for richer and more diverse features. The negativity in learning cost resulting from feature complexity dominates in these user segments, and simpler apps benefit from the seasonality based-timing. This study provides insight into an Operations-Marketing interface issue by demonstrating a significant interaction between market demand patterns and product features.

Fifth, through two post-hoc analyses, we dive deeper into the underlying mechanisms of the negative impact of feature diversity. We show that app developers accumulate knowledge from prior launch experience such that they can make better decisions on SDK selection and optimize the benefits. Interestingly, by observing the interaction plots, we find that the market penalizes firms with reputation and experience if they launch low richness and low diversity apps. An explanation for this finding is that the experience variable is also capturing the firm reputation in the market. Results show that as a firm grows their reputation, possibly users in the market may expect more diverse experience and richer features in the newly launched apps. A firm that does not innovate and still maintains the low richness and low diversity may eventually suffer from reduced performance. In sum, the results suggest two things. First, publishers indeed learn over time from prior development and launch experiences and make better decisions regarding SDK implementations. From a broader picture, this shows that while managing complexity is a challenge for publishers, they improve their decision-making regarding product complexity from prior experience and excel over time. Second, there is a market pull for constant technology adoption and innovation regarding app features, whereby a firm that does not offer novel features in their newly launched apps may quickly lose its place in the market.

We also find that the negative impact of feature diversity comes mostly from monetization features, which shows the trade-off relationship between user experience and publisher revenue sources. Carefully balancing the two opposing forces poses difficult challenges for developers in this business.

Finally, to gain a better understanding of enhancement updates, we estimate its effect on app performance measures such as daily downloads and longevity of the app. This estimate of the enhancement can serve as important input parameters when developers or future researchers attempt to develop optimized models that jointly consider the software maintenance costs and the demand benefits. The tests regarding the contextual variables helps us understand how the effectiveness enhancement updates change under varying conditions. Specifically, we show that update schedule regularity, lifecycle stage of the release, and market activity levels at the time of release to have significant moderating effect on enhancement updates. While prior literature dominantly discusses software maintenance as a reactive cost minimizing task, we suggest a perspective that proactively uses software maintenance as a source of competitive advantage.

4.3.2 Managerial Implications

Practitioners can benefit from the findings of this study in a number of ways. First, we have addressed a critical managerial decision in the context of mobile app development with regard to SDK choice. Now that the number of SDKs available in the market is growing exponentially, picking and choosing the right SDKs and the resulting feature set is becoming an essential problem for app developers. Our findings suggest that it is crucial to consider SDK choice from a perspective of adding more richness or diversity to the app's features. It is vital for managers to know that a diverse feature set that lacks richness can backfire and lead to reduced performance.

Second, the significant interaction between market activity, feature richness and

feature diversity suggest that managers should be careful in assuming that the market segment is homogenous between the peak and off-peak season. If there is a cost to postponing product launch after development completion to potentially take advantage of launching in the peak-season, this wait may not be justified, especially if the app is complex. On the other hand, apps with relatively simpler features and a straightforward value proposition can benefit more from a well-timed product launch. Fourth, the results suggest that publishers should be cautious in expanding the feature diversity of apps. It is advisable that the feature expansion takes place after accumulating several product launch experiences. Prior product launch and managing experience allows the developer to accumulate knowledge about the user preference and behaviors, which can be valuable in optimizing new feature category experience. Finally, developers should be aware of the trade-off relationships between adding features that enhance user experience versus those focused on monetization. Although these features may be tempting, a careful balance between the two will be essential for sustaining a healthy app service.

Third, we provide estimates on the effectiveness of three types of update strategies. Specifically, we propose three enhancement update strategies that may prove to be helpful to mobile app developers in further increasing the effectiveness of the updates. The three strategies are (1) regularity-based software maintenance, (2) lifecycle-based software maintenance, and (3) market activity-based software maintenance. While the benefits of keeping a regular maintenance schedule may be more prominent on the supply-side in allocating resources and lowering costs, we also find significant positive benefits from the demand-side. When updates are introduced to users regularly, this can minimize the disruptions and frustrations from service downtime. Regular updates can also signal the users that there will be another update to expect in the next update cycle and create a sense of anticipation. Based on our estimates of the enhancement update effects in the three life cycle stages, we find that updates made

in the second stage showed strongest impacts, followed by the first stage, and third stage updates showing a negative impact. Through robustness checks that rely on a more granular split of the life cycle, these main findings are consistently supported. We can see that the second stage in a five-stage life cycle remains strongly significant, which means that the region that significantly reinforces enhancement updates can be a very tight opportunity window. Therefore, we suggest that publishers should keep a keen eye on the performance trajectory of their apps and make good use of information that the life cycle signals. Because the moderation effect is relatively larger than the regularity effect, it would make sense for developers to even go out of their regular schedules and focus their maintenance efforts on particular life cycle stages. The interaction between market activity and enhancement update tests the feasibility of a market activity-based update strategy. One way to take advantage of market activity is by attending to predictable seasonal demands. Many apps introduce holiday-specific events and enhancement updates. For example, a popular farming game, Hay Day, changes the entire theme and background of the app during Christmas every year. The developer changes the background music and adds limited holiday merchandise in the store for users to purchase. The digital marketplace for mobile apps provides various advantages to the players in the market. One of those advantages is that firms can extract information about the market structure and monitor their competitors for benchmarking. For this reason, currently there are many market intelligence firms that provide information and consulting services for app developers. The market activity-based maintenance strategy would require developers to acquire additional information about the market and use that information to plan their enhancement update schedules.

4.4 Limitations

In interpreting the findings, we acknowledge some limitations. First, the scope of the study pertains to iOS gaming apps only. This reduction in scope allowed us to reduce concerns related to unobserved influences from mobile device characteristics and user demographics. However, we believe that the gaming context is an extreme case regarding competition intensity and short innovation cycles, which the mobile app economy generally shares in differing degrees. Therefore, our findings on complexity and market activity should apply to other app categories as well that use modularized software development kits.

Second, we do not have a more detailed performance measure that captures the actual usage of the app. The actual duration of use would be ideal to capture user engagement with the mobile app.

Third, the Bass model predictions were obtained using the entire time series of an app. A more rigorous approach would be using data points preceding the time of update and incorporating a temporal gap between the estimation window and the prediction window. Applying this procedure would drastically reduce our sample because updates with few estimation data points would fail to converge. This reduction in sample size would happen mostly for updates occurring early in the app's life cycle, and would introduce a sample selection bias. We believe our procedure may have generated an upward biased prediction and led to conservative estimates of the enhancement updates. Therefore, in practice, enhancement updates may exhibit larger effects on the app's performance.

Fourth, while software maintenance is a general task required for all software products, our focus is on mobile apps which relies on the freemium business model. This may limit the generalizability of our findings. However, the software industry is transitioning to a Software as a Service (SaaS) model which entails other similar revenue generation models such as subscription services and free trials. Therefore,

we believe that the applicability of findings from the mobile app context will expand to the software industry slowly over time.

4.5 Recommendation for Future Work

This research opens up new avenues for future research. Future research can look into user-level behavior data to strengthen the link between app features and performance outcomes. The user-level behavior includes in-app purchasing behavior and in-app content generation behavior. Moreover, future research can explore interactions between enhancement update strategies and pricing models for software. This includes in-app purchasing option pricing and other types of software that relies on subscription and physical distribution that involves pricing decisions. Also, future research can analyze the content of each enhancement update in detail and identify how enhancement updates evolve over the app's lifetime. This can help us understand why specific enhancement updates can lead to adverse outcomes. Finally, future research can investigate the link between enhancement updates and a developer's advertising behavior. This study assumes that the two activities are closely tied, but in practice, there may be cases where the two activities show discrepancies.

4.6 Conclusions

Overall, this study sheds light on both theory and practice on the emerging trend in mobile app ecosystems. Our conceptualization focusing on the differences of mobile app lifecycles opens novel research avenues yet to be explored. Asymmetric effects of the feature complexity dimensions and their interactions with demand patterns are the primary findings of this study. We also estimate the effect of enhancement updates on freemium mobile app download performance and longevity. Additionally, via exploring the moderation effects of update schedule regularity, life cycle stage, and market activity, we show the feasibility of three update strategies. In sum, we

believe finding from our study help mobile app developers in formulating effective app development, deployment, and updating strategies. Innovations in the mobile app industry have the potential to change the way we do things for the better and can touch upon lives where traditional businesses failed to do so.

Bibliography

[1] Ritu Agarwal and Elena Karahanna. "Time flies when you're having fun: Cognitive absorption and beliefs about information technology usage". In: *MIS quarterly* (2000), pp. 665–694. ISSN: 0276-7783.

[2] Carl R Anderson and Carl P Zeithaml. "Stage of the product life cycle, business strategy, and business performance". In: *Academy of Management journal* 27.1 (1984), pp. 5–24.

[3] Appsee. *The ultimate SDK guide for mobile apps*. Appsee, 2018, pp. 1–44.

[4] Ashish Arora, Jonathan P Caulkins, and Rahul Telang. "Research note—Sell first, fix later: Impact of patching on software quality". In: *Management Science* 52.3 (2006), pp. 465–471.

[5] Kankanhalli Atreyi, Jonathan Hua Ye, and Hock Hai Teo. "Comparing potential and actual innovators: an empirical study of mobile data service innovation". In: *MIS Quarterly* 39.3 (2015), pp. 667–682. DOI: 10.25300/MISQ/2015/39.3.07.

[6] Terrence August, Duy Dao, and Hyoduk Shin. "Optimal Timing of Sequential Distribution: The Impact of Congestion Externalities and Day-and-Date Strategies". In: *Marketing Science* 34.5 (2015), pp. 755–774. ISSN: 0732-2399. DOI: 10.1287/mksc.2015.0936. URL: http://pubsonline.informs.org/doi/10.1287/mksc.2015.0936.

[7] David H. Autor. "Outsourcing at Will: The Contribution of Unjust Dismissal Doctrine to the Growth of Employment Outsourcing". In: *Journal of Labor Economics* 21.1 (2003), pp. 1–42.

[8] Kostas Axarloglou. “The cyclicality of new product introductions”. In: *The Journal of Business* 76.1 (2003), pp. 29–48. ISSN: 0021-9398.

[9] Carliss Young Baldwin and Kim B Clark. *Design rules: The power of modularity*. Vol. 1. MIT press, 2000.

[10] Rajiv D Banker, Gordon B Davis, and Sandra A Slaughter. “Software development practices, software complexity, and software maintenance performance: A field study”. In: *Management science* 44.4 (1998), pp. 433–450. ISSN: 0025-1909.

[11] Rajiv D Banker and Sandra A Slaughter. “A field study of scale economies in software maintenance”. In: *Management science* 43.12 (1997), pp. 1709–1725.

[12] Frank M Bass. “A new product growth for model consumer durables”. In: *Management science* 15.5 (1969), pp. 215–227.

[13] Ornella Benedettini and Andy Neely. “Complexity in services: an interpretative framework”. In: *23rd Annual Conference of the Production and Operations Management Society (POMS)*. 2012, pp. 1–11.

[14] Keith H Bennett and Vaclav T Rajlich. “Software maintenance and evolution: a roadmap”. In: *Proceedings of the Conference on the Future of Software Engineering*. ACM. 2000, pp. 73–87.

[15] Steven Berry and Panle Jia. “Tracing the woes: An empirical analysis of the airline industry”. In: *American Economic Journal: Microeconomics* 2.3 (2010), pp. 1–43. ISSN: 1945-7669.

[16] Steven Berry, James Levinsohn, and Ariel Pakes. “Automobile prices in market equilibrium”. In: *Econometrica: Journal of the Econometric Society* (1995), pp. 841–890. ISSN: 0012-9682.

[17] Marianne Bertrand, Esther Duflo, and Sendhil Mullainathan. "How much should we trust differences-in-differences estimates?" In: *The Quarterly journal of economics* 119.1 (2004), pp. 249–275.

[18] Chris Brauer. *The App Attention Span*. 2014. URL: https://www.appdynamics.com/media/uploaded-files/1425406960/app-attention-span-research-report-1.pdf (visited on 05/07/2018).

[19] Christina L Brown and Gregory S Carpenter. "Why is the trivial important? A reasons-based account for the effects of trivial attributes on choice". In: *Journal of Consumer Research* 26.4 (2000), pp. 372–385. ISSN: 1537-5277.

[20] Shona L Brown and Kathleen M Eisenhardt. "The art of continuous change: Linking complexity theory and time-paced evolution in relentlessly shifting organizations". In: *Administrative science quarterly* (1997), pp. 1–34.

[21] Tom J Brown et al. "Spreading the word: Investigating antecedents of consumers' positive word-of-mouth intentions and behaviors in a retailing context". In: *Journal of the Academy of Marketing Science* 33.2 (2005), pp. 123–138. ISSN: 0092-0703.

[22] Maurice J. G. Bun and Teresa D. Harrison. "OLS and IV estimation of regression models including endogenous interaction terms". In: *Econometric Reviews* (Jan. 2018), pp. 1–14. ISSN: 0747-4938. DOI: 10.1080/07474938.2018.1427486. URL: https://www.tandfonline.com/doi/full/10.1080/07474938.2018.1427486.

[23] Roger J. Calantone et al. "The Effects of Competition in Short Product Life-Cycle Markets: The Case of Motion Pictures". In: *Journal of Product Innovation Management* 27.3 (May 2010), pp. 349–361. ISSN: 07376782. DOI: 10.1111/j.1540-5885.2010.00721.x. URL: http://doi.wiley.com/10.1111/j.1540-5885.2010.00721.x.

[24] Gregory S Carpenter, Rashi Glazer, and Kent Nakamoto. “Meaningful brands from meaningless differentiation: The dependence on irrelevant attributes”. In: *Journal of Marketing Research* (1994), pp. 339–350. ISSN: 0022-2437.

[25] Ned Chapin et al. “Types of software evolution and software maintenance”. In: *Journal of software maintenance and evolution: Research and Practice* 13.1 (2001), pp. 3–30.

[26] Sam Cheney. *The Data Behind 10 Years of the iOS App Store*. 2018. URL: `https://www.appannie.com/en/insights/market-data/data-behind-10-years-ios-app-store/` (visited on 05/07/2018).

[27] Judith A Chevalier and Dina Mayzlin. “The effect of word of mouth on sales: Online book reviews”. In: *Journal of marketing research* 43.3 (2006), pp. 345–354. ISSN: 0022-2437.

[28] Lawrence J Christiano and Terry J Fitzgerald. “The band pass filter”. In: *international economic review* 44.2 (2003), pp. 435–465. ISSN: 0020-6598.

[29] David J Closs, Gilbert N Nyaga, and M Douglas Voss. “The differential impact of product complexity, inventory level, and configuration capacity on unit and order fill rate performance”. In: *Journal of Operations Management* 28.1 (2010), pp. 47–57. ISSN: 0272-6963.

[30] Clutch. *Top Mobile App Development Companies*. 2018. URL: `https://clutch.co/directory/mobile-application-developers` (visited on 11/15/2018).

[31] Robert G Cooper and Elko J Kleinschmidt. “New products: what separates winners from losers?” In: *Journal of Product Innovation Management: AN INTERNATIONAL PUBLICATION OF THE PRODUCT*

DEVELOPMENT & MANAGEMENT ASSOCIATION 4.3 (1987), pp. 169–184. ISSN: 0737-6782.

[32] Mihaly Csikszentmihalyi. *Flow: The psychology of optimal performance*. 1990.

[33] Isabelle Dalmasso et al. "Survey, comparison and evaluation of cross platform mobile application development tools". In: *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*. IEEE, 2013, pp. 323–328. ISBN: 1467324809.

[34] Diya Datta and Sangaralingam Kajanan. "Do app launch times impact their subsequent commercial success? an analytical approach". In: *2013 International Conference on Cloud Computing and Big Data (CloudCom-Asia)*. IEEE, 2013, pp. 205–210. ISBN: 1479928305.

[35] Richard A D'aveni. *Hypercompetition*. Simon and Schuster, 2010.

[36] Artyom Dogtiev. *How Much Does App Development Cost?* 2018. URL: `http://www.businessofapps.com/guide/app-development-cost/` (visited on 07/20/2018).

[37] Larry Downes and Paul Nunes. *Big bang disruption: Strategy in the age of devastating innovation*. Penguin, 2014. ISBN: 1591846900.

[38] Wenjing Duan, Bin Gu, and Andrew B Whinston. "The dynamics of online word-of-mouth and product sales—An empirical investigation of the movie industry". In: *Journal of retailing* 84.2 (2008), pp. 233–242. ISSN: 0022-4359.

[39] J Alberto Espinosa et al. "Familiarity, complexity, and team performance in geographically distributed software development". In: *Organization science* 18.4 (2007), pp. 613–630. ISSN: 1047-7039.

[40] Wai Fong Boh, Sandra A. Slaughter, and J. Alberto Espinosa. "Learning from Experience in Software Development: A Multilevel Analysis". In: *Management Science* 53.8 (2007), pp. 1315–1331. ISSN: 0025-1909. DOI: 10.1287/mnsc.1060.0687. URL: http://pubsonline.informs.org/doi/abs/10.1287/mnsc.1060.0687.

[41] Rajiv Garg and Rahul Telang. "Inferring App Demand from Publicly Available Data". In: *MIS Quarterly* 37.4 (2013), pp. 1253–1264. ISSN: 0276-7783. DOI: 10.25300/MISQ/2013/37.4.12.

[42] Sachin Garg et al. "Analysis of preventive maintenance in transactions based software systems". In: *IEEE transactions on Computers* 47.1 (1998), pp. 96–107.

[43] Anindya Ghose and Sang Han. "Estimating demand for mobile applications in the new economy". In: *Management Science* 60.6 (2014), pp. 1470–1488. ISSN: 00251909. DOI: 10.1287/mnsc.2014.1945. URL: http://www.scopus.com/inward/record.url?eid=2-s2.0-84902252476%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=.

[44] David Godes and Dina Mayzlin. "Using Online Conversations to Study Word-of-Mouth Communication". In: *Marketing Science* 23.4 (Nov. 2004), pp. 545–560. ISSN: 0732-2399. DOI: 10.1287/mksc.1040.0071. URL: http://pubsonline.informs.org/doi/abs/10.1287/mksc.1040.0071.

[45] Bilal Gokpinar, Wallace J Hopp, and Seyed M R Iravani. "The impact of misalignment of organizational structure and product architecture on quality in complex product development". In: *Management science* 56.3 (2010), pp. 468–484. ISSN: 0025-1909.

[46] Mary Ellen Gordon. *Benchmarking the Half-Life and Decay of Mobile Apps.* 2018. URL:

http://flurrymobile.tumblr.com/post/115191376315/bench-smarking-the-half-life-and-decay-of-mobile (visited on 07/15/2018).

[47] Des Greer and Guenther Ruhe. "Software release planning: an evolutionary and iterative approach". In: *Information and software technology* 46.4 (2004), pp. 243–253.

[48] Abbie Griffin. "The effect of project and process characteristics on product development cycle time". In: *Journal of Marketing Research* (1997), pp. 24–35. ISSN: 0022-2437.

[49] Term L Griffith. "Technology features as triggers for sensemaking". In: *Academy of Management review* 24.3 (1999), pp. 472–488. ISSN: 0363-7425.

[50] Sven Grundberg. *Angry Birds Space Hits 10 Million Downloads*. Mar. 2012. URL: https://on.wsj.com/1BDGX0b.

[51] Lars Peter Hansen. "Large sample properties of generalized method of moments estimators". In: *Econometrica: Journal of the Econometric Society* (1982), pp. 1029–1054. ISSN: 0012-9682.

[52] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[53] Kevin B Hendricks and Vinod R Singhal. "Association between supply chain glitches and operating performance". In: *Management science* 51.5 (2005), pp. 695–711.

[54] Teck-Hua Ho, Sergei Savin, and Christian Terwiesch. "Managing demand and sales dynamics in new product diffusion under supply constraint". In: *Management Science* 48.2 (2002), pp. 187–206. ISSN: 0025-1909.

[55] Robert J Hodrick and Edward C Prescott. "Postwar US business cycles: an empirical investigation". In: *Journal of Money, credit, and Banking* (1997), pp. 1–16. ISSN: 0022-2879.

[56] Charles W Hofer. “Toward a Contingency Theory of Business Strategy”. In: *Academy of Management Journal* 18.4 (1975), pp. 784–810.

[57] James G Hutton. “A study of brand equity in an organizational-buying context”. In: *Journal of Product & Brand Management* 6.6 (1997), pp. 428–439. ISSN: 1061-0421.

[58] Guido W Imbens and Donald B Rubin. “Rubin causal model”. In: *The new palgrave dictionary of economics* (2017), pp. 1–10.

[59] Mark A Jacobs and Morgan Swink. “Product portfolio architectural complexity and operational performance: Incorporating the roles of learning and fixed assets”. In: *Journal of Operations Management* 29.7-8 (2011), pp. 677–691. ISSN: 0272-6963.

[60] Charlene Jennett et al. “Measuring and defining the experience of immersion in games”. In: *International journal of human-computer studies* 66.9 (2008), pp. 641–661. ISSN: 1071-5819.

[61] Yonghua Ji et al. “Optimal enhancement and lifetime of software systems: A control theoretic analysis”. In: *Production and Operations Management* 20.6 (2011), pp. 889–904.

[62] Patrik Jonsson. “Towards an holistic understanding of disruptions in Operations Management”. In: *Journal of operations management* 18.6 (2000), pp. 701–718.

[63] Timo Kaski and Jussi Heikkila. “Measuring product structures to improve demand-supply chain efficiency”. In: *International Journal of Technology Management* 23.6 (2002), pp. 578–598. ISSN: 0267-5730.

[64] Stuart A Kauffman and Edward D Weinberger. “The NK model of rugged fitness landscapes and its application to maturation of the immune response”. In: *Journal of theoretical biology* 141.2 (1989), pp. 211–245. ISSN: 0022-5193.

[65] Ji-Hern Kim and Yong J Hyun. “A model to investigate the influence of marketing-mix efforts and corporate image on brand equity in the IT software sector”. In: *Industrial marketing management* 40.3 (2011), pp. 424–438. ISSN: 0019-8501.

[66] Barbara A Kitchenham et al. “Towards an ontology of software maintenance”. In: *Journal of Software Maintenance: Research and Practice* 11.6 (1999), pp. 365–389.

[67] Frank Kleibergen and Richard Paap. “Generalized reduced rank tests using the singular value decomposition”. In: *Journal of econometrics* 133.1 (2006), pp. 97–126. ISSN: 0304-4076.

[68] Lee J Krajewski, Larry P Ritzman, and Manoj K Malhotra. *Operations Management: Processes and Supply Chains*. Pearson, 2018.

[69] Melanie E Kreye, Jens K Roehrich, and Michael A Lewis. “Servitising manufacturers: the impact of service complexity and contractual and relational capabilities”. In: *Production Planning & Control* 26.14-15 (2015), pp. 1233–1246. ISSN: 0953-7287.

[70] Robert E Krider and Charles B Weinberg. “Competitive dynamics and the introduction of new products: The motion picture timing game”. In: *Journal of Marketing Research* (1998), pp. 1–15. ISSN: 0022-2437.

[71] Mayuram S Krishnan, Tridas Mukhopadhyay, and Charles H Kriebel. “A decision model for software maintenance”. In: *Information Systems Research* 15.4 (2004), pp. 396–412.

[72] Vidyadhar G Kulkarni et al. “Optimal allocation of effort to software maintenance: A queuing theory approach”. In: *Production and Operations Management* 18.5 (2009), pp. 506–515.

[73] Gunwoong Lee and T. S. Raghu. "Determinants of Mobile Apps' Success: Evidence from the App Store Market". In: *Journal of Management Information Systems* 31.2 (2014), pp. 133–170. ISSN: 0742-1222. DOI: 10.2753/MIS0742-1222310206. arXiv: arXiv:1011.1669v3. URL: http://www.tandfonline.com/doi/full/10.2753/MIS0742-1222310206.

[74] Arthur Lewbel. "Using heteroscedasticity to identify and estimate mismeasured and endogenous regressor models". In: *Journal of Business & Economic Statistics* 30.1 (Jan. 2012), pp. 67–80. ISSN: 0735-0015. DOI: 10.1080/07350015.2012.643126. URL: https://www.tandfonline.com/doi/full/10.1080/07350015.2012.643126.

[75] Bennet P Lientz, E. Burton Swanson, and Gail E Tompkins. "Characteristics of application software maintenance". In: *Communications of the ACM* 21.6 (1978), pp. 466–471.

[76] Liftoff. *Mobile Gaming Apps Report: User acquisition trends and benchmark*. Liftoff, 2017, pp. 1–20.

[77] Omer Liss. *The Hidden Seasonality of Social Gaming*. 2017. URL: https://www.optimove.com/blog/hidden-seasonality-social-gaming (visited on 11/12/2018).

[78] Charles Zhechao Liu, Yoris A. Au, and Hoon Seok Choi. "Effects of Freemium Strategy in the Mobile App Market: An Empirical Study of Google Play". In: *Journal of Management Information Systems* 31.3 (July 2014), pp. 326–354. ISSN: 1557928X. DOI: 10.1080/07421222.2014.995564. URL: http://www.tandfonline.com/doi/abs/10.1080/07421222.2014.995564.

[79] Amit Mehra, Abraham Seidmann, and Probal Mojumder. "Product life-cycle management of packaged software". In: *Production and Operations Management* 23.3 (2014), pp. 366–378.

[80] Marc H Meyer and Kathleen Foley Curley. "An applied framework for classifying the complexity of knowledge-based systems". In: *Mis Quarterly* (1991), pp. 455–472. ISSN: 0276-7783.

[81] Sven Mikolon et al. "The complex role of complexity: how service providers can mitigate negative effects of perceived service complexity when selling professional services". In: *Journal of Service Research* 18.4 (2015), pp. 513–528. ISSN: 1094-6705.

[82] Tyler Moore. *What's the Cost to Maintain an App?* 2019. URL: https://www.app-press.com/blog/whats-the-cost-to-maintain-an-app (visited on 03/29/2019).

[83] Anirban Mukherjee and Vrinda Kadiyali. "Modeling multichannel home video demand in the US motion picture industry". In: *Journal of Marketing Research* 48.6 (2011), pp. 985–995. ISSN: 0022-2437.

[84] Sriram Narayanan, Sridhar Balasubramanian, and Jayashankar M. Swaminathan. "A Matter of Balance: Specialization, Task Variety, and Individual Learning in a Software Maintenance Environment". In: *Management Science* 55.11 (2009), pp. 1861–1876. ISSN: 0025-1909. DOI: 10.1287/mnsc.1090.1057. URL: http://pubsonline.informs.org/doi/abs/10.1287/mnsc.1090.1057.

[85] Phillip Nelson. "Information and consumer behavior". In: *Journal of political economy* 78.2 (1970), pp. 311–329. ISSN: 0022-3808.

[86] Newzoo. *Newzoo Global App Store Intelligence Q4 2016*. 2016. URL: https://newzoo.com/insights/articles/global-mobile-market-report-app-market-to-gross-44-8bn-this-year/ (visited on 06/20/2018).

[87] Stephen Nickell. “Biases in dynamic models with fixed effects”. In: *Econometrica* 49.6 (1981), pp. 1417–1426. ISSN: 0012-9682.

[88] Jakob Nielsen. *Usability engineering*. Elsevier, 1994. ISBN: 0080520294.

[89] John A Norton and Frank M Bass. “A diffusion theory model of adoption and substitution for successive generations of high-technology products”. In: *Management science* 33.9 (1987), pp. 1069–1086.

[90] John T Nosek and Prashant Palvia. “Software maintenance management: changes in the last decade”. In: *Journal of Software Maintenance: Research and Practice* 2.3 (1990), pp. 157–174.

[91] Sharon Novak and Steven D Eppinger. “Sourcing by design: Product complexity and the supply chain”. In: *Management science* 47.1 (2001), pp. 189–204. ISSN: 0025-1909.

[92] Stephen M Nowlis and Itamar Simonson. “The effect of new product features on brand choice”. In: *Journal of marketing research* (1996), pp. 36–46. ISSN: 0022-2437.

[93] Adrian Rodney Pagan and Anthony David Hall. “Diagnostic tests as residual analysis”. In: *Econometric Reviews* 2.2 (1983), pp. 159–218. ISSN: 0747-4938.

[94] Riley Panko. *How Small Businesses Build Mobile Apps*. 2018. URL: `https://themanifest.com/app-development/how-small-businesses-build-mobile-apps` (visited on 11/12/2018).

[95] Sarah Perez. *Paid Apps On The Decline: 90% Of iOS Apps Are Free, Up From 80-84% During 2010-2012, Says Flurry*. 2013. URL: `https://techcrunch.com/2013/07/18/paid-apps-on-the-decline-90-of-ios-apps-are-free-up-from-80-84-during-2010-2012-says-flurry/` (visited on 01/15/2018).

[96] Sergio Picazo-Vela et al. “Why provide an online review? An extended theory of planned behavior and the role of Big-Five personality traits”. In: *Computers in Human Behavior* 26.4 (2010), pp. 685–696. ISSN: 0747-5632.

[97] Sonja Radas and Steven M Shugan. “Seasonal marketing and timing new product introductions”. In: *Journal of Marketing Research* (1998), pp. 296–315. ISSN: 0022-2437.

[98] Karthik Ramachandran and Vish Krishnan. “Design architecture and introduction timing for rapidly improving industrial products”. In: *Manufacturing & Service Operations Management* 10.1 (2008), pp. 149–171. ISSN: 1523-4614.

[99] Morten O Ravn and Harald Uhlig. “On adjusting the Hodrick-Prescott filter for the frequency of observations”. In: *Review of economics and statistics* 84.2 (2002), pp. 371–376. ISSN: 0034-6535.

[100] Marco C Rozendaal et al. “Game feature and expertise effects on experienced richness, control and engagement in game play”. In: *AI & society* 24.2 (2009), pp. 123–133. ISSN: 0951-5666.

[101] SafeDK. *Mobile SDKs Data Trends in the Android Market*. Tech. rep. SafeDK, 2018, pp. 1–33.

[102] Ivo Salmre. *Writing mobile code: Essential software engineering for building mobile applications*. Addison-Wesley Professional, 2005.

[103] Sergei Savin and Christian Terwiesch. “Optimal product launch times in a duopoly: Balancing life-cycle revenues with product cost”. In: *Operations Research* 53.1 (2005), pp. 26–47. ISSN: 0030-364X.

[104] Richard Schmalensee. “Antitrust issues in Schumpeterian industries”. In: *American Economic Review* 90.2 (2000), pp. 192–196.

[105] Barry Schwartz. "Self-determination: The tyranny of freedom." In: *American psychologist* 55.1 (2000), p. 79. ISSN: 1557987041.

[106] Orly Shoavi. *How Top Mobile Apps Fight the SDK Fatigue and the Effect on their Business Results*. 2017. URL: http://blog.safedk.com/sdk-economy/sdk-fatigue-sdks-control-mobile/ (visited on 11/12/2018).

[107] Manuel E Sosa, Steven D Eppinger, and Craig M Rowles. "The misalignment of product architecture and organizational structure in complex product development". In: *Management science* 50.12 (2004), pp. 1674–1689. ISSN: 0025-1909.

[108] Statista. *Mobile gaming app revenue 2015-2020*. 2018. URL: https://www.statista.com/statistics/511639/global-mobile-game-app-revenue/ (visited on 11/12/2018).

[109] Statista. *Mobile Gaming Industry - Statistics & Facts*. 2018. URL: https://www.statista.com/topics/1906/mobile-gaming%20/ (visited on 11/12/2018).

[110] James H Stock and Motohiro Yogo. "Testing for weak instruments in linear IV regression". In: *Identification and inference for econometric models: Essays in honor of Thomas Rothenberg* (2005).

[111] Sam Stoddard. *Complexity Creep*. 2017. URL: https://magic.wizards.com/en/articles/archive/latest-developments/complexity-creep-2017-02-24 (visited on 11/12/2018).

[112] Karsten Strauss. *The $2.4 Million-Per-Day Company: Supercell*. 2013. URL: https://www.forbes.com/sites/karstenstrauss/2013/04/18/the-2-4-million-per-day-company-supercell/%7B%5C#%7D23b37cdf6fc1 (visited on 07/20/2018).

[113] E Burton Swanson. "The dimensions of maintenance". In: *Proceedings of the 2nd international conference on Software engineering*. IEEE Computer Society Press. 1976, pp. 492–497.

[114] Kalyan T Talluri and Garrett J Van Ryzin. *The theory and practice of revenue management*. Vol. 68. Springer Science & Business Media, 2006. ISBN: 0387273913.

[115] Aaron Taube. *People Spend Way More On Purchases In Free Apps Than They Do Downloading Paid Apps*. Dec. 2013. URL: https://www.businessinsider.com/in-app-purchases-dominate-revenue-share-2013-12.

[116] Debora Viana Thompson, Rebecca W Hamilton, and Roland T Rust. "Feature fatigue: When product capabilities become too much of a good thing". In: *Journal of marketing research* 42.4 (2005), pp. 431–442. ISSN: 0022-2437.

[117] Eray Tuzin et al. "Adopting integrated application lifecycle management within a large-scale software company: An action research approach". In: *Journal of Systems and Software* 149 (2019), pp. 63–82.

[118] Viswanath Venkatesh et al. "User Acceptance of Information Technology: Toward a Unified View". In: *MIS Quarterly* 27.3 (2003), pp. 425–478.

[119] Shawnee K. Vickery et al. "Product Modularity, Process Modularity, and New Product Introduction Performance: Does Complexity Matter?" In: *Production and Operations Management* 25.4 (Apr. 2016), pp. 751–770. ISSN: 10591478. DOI: 10.1111/poms.12495. URL: http://doi.wiley.com/10.1111/poms.12495.

[120] Andrew Waber. *The Seasonality of Mobile Device Usage: Warmer Weather Tempers Tech*. 2014. URL: https://marketingland.com/seasonality-

mobile-device-usage-warmer-weather-tempers-tech-95937 (visited on 11/12/2018).

[121] Qing Wang and Nick von Tunzelmann. "Complexity and the functions of the firm: breadth and depth". In: *Research Policy* 29.7-8 (2000), pp. 805–818. ISSN: 0048-7333.

[122] Jane Webster and Hayes Ho. "Audience engagement in multimedia presentations". In: *ACM SIGMIS Database: the DATABASE for Advances in Information Systems* 28.2 (1997), pp. 63–77. ISSN: 0095-0033.

[123] Jeffrey M. Wooldridge. *Econometric analysis of cross section and panel data*. Cambridge: MIT Press, 2010.

[124] Valarie A Zeithaml. "Consumer perceptions of price, quality, and value: a means-end model and synthesis of evidence". In: *The Journal of marketing* (1988), pp. 2–22. ISSN: 0022-2429.

Appendix A

Essay 1 Robustness Check Results

We present additional results using alternative variable operationalizations and model specifications.

Table A.1: Alternative DV IV Regression Results

	lndownloads				lncdd			
$diversity_{i,l}$	−0.160***	−0.316***	−0.171***	−0.175***	−0.143*	−0.352***	−0.207***	−0.195***
	(0.059)	(0.057)	(0.046)	(0.055)	(0.073)	(0.065)	(0.057)	(0.065)
$richness_{i,l}$	0.046 * *	−0.060 * *	0.057***	0.060***	0.056 * *	−0.029	0.083***	0.085***
	(0.018)	(0.025)	(0.016)	(0.018)	(0.022)	(0.027)	(0.019)	(0.019)
$lnmkttrend_{c,l}$	0.287***	0.205 * *	0.669***	0.627***	0.257 * *	0.222*	0.633***	0.620***
	(0.103)	(0.101)	(0.144)	(0.120)	(0.130)	(0.125)	(0.187)	(0.154)
$lnmktact_{c,l}$	0.231 * *	0.166	0.192*	0.222 * *	0.248	0.180	0.156	0.145
	(0.115)	(0.114)	(0.109)	(0.111)	(0.155)	(0.151)	(0.145)	(0.151)
$pubexp_{j,l}$	−0.011*	−0.009	−0.008	−0.007	−0.023***	−0.022***	−0.019***	−0.016***
	(0.006)	(0.006)	(0.006)	(0.006)	(0.007)	(0.006)	(0.006)	(0.006)
$competition_{c,t}$	−0.173	−0.070	−0.186	−0.231*	−0.020	0.087	0.029	0.010
	(0.133)	(0.130)	(0.124)	(0.127)	(0.173)	(0.169)	(0.162)	(0.166)
$screenshots_{i,l}$	0.195***	0.190***	0.159***	0.179***	0.246***	0.211***	0.209***	0.231***
	(0.039)	(0.037)	(0.036)	(0.036)	(0.047)	(0.045)	(0.043)	(0.043)
$ageres_{i,l}$	−0.182***	−0.191***	−0.191***	−0.201***	−0.137*	−0.179***	−0.163 * *	−0.136 * *
	(0.063)	(0.059)	(0.059)	(0.059)	(0.070)	(0.065)	(0.067)	(0.066)
$multicategory_{i,l}$	−0.227 * *	−0.175*	−0.218 * *	−0.208 * *	−0.146	−0.160	−0.144	−0.161
	(0.099)	(0.092)	(0.092)	(0.092)	(0.114)	(0.107)	(0.107)	(0.106)
$multiplatform_{i,l}$	0.553***	0.487***	0.518***	0.546***	0.641***	0.534***	0.595***	0.565***
	(0.147)	(0.129)	(0.138)	(0.138)	(0.162)	(0.149)	(0.153)	(0.151)
$appsize_{i,l}$	0.001***	0.001***	0.001***	0.001***	0.001***	0.001***	0.001***	0.001***
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
$updates_{i,t}$	0.932***	0.891***	0.934***	0.949***	1.690***	1.668***	1.673***	1.705***
	(0.086)	(0.080)	(0.082)	(0.083)	(0.099)	(0.093)	(0.094)	(0.094)
$sincelaunch_{i,t}$	−0.001	−0.002	−0.001	−0.001	0.006 * *	0.005 * *	0.005 * *	0.006 * *
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$diversity_{i,l} \times richness_{i,l}$		0.018***				0.017***		
		(0.003)				(0.003)		
$lnmkact_{c,l} \times diversity_{i,l}$			−0.094***				−0.095***	
			(0.022)				(0.028)	
$lnmkact_{c,l} \times richness_{i,l}$				−0.045***				−0.050***
				(0.008)				(0.010)
Constant	4.028***	5.094***	4.599***	4.300***	3.342 * *	4.884***	4.443***	4.519***
	(1.015)	(0.965)	(0.908)	(0.933)	(1.310)	(1.232)	(1.163)	(1.218)
Subcategory FE	YES	YES	YES	YES	YES	YES	YES	YES
Observations	1,782	1,782	1,782	1,782	1,782	1,782	1,782	1,782
R-squared	0.259	0.267	0.261	0.263	0.303	0.303	0.304	0.305

Robust standard errors clustered by publisher in parentheses (*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Table A.2: Alternative Variable Operationalization Results

	CF Filter				Max Richness			
$diversity_{i,l}$	−0.257***	−0.380***	−0.222***	−0.338***	−0.164***	−0.292***	−0.172***	−0.141***
	(0.079)	(0.070)	(0.061)	(0.070)	(0.054)	(0.070)	(0.037)	(0.047)
$richness_{i,l}$	0.072***	−0.039	0.097***	0.122***	0.069*	−0.145**	0.140***	0.098**
	(0.025)	(0.033)	(0.021)	(0.024)	(0.042)	(0.066)	(0.038)	(0.039)
$lnmkttrend_{c,l}$	−0.096	−0.120	0.166	−0.116	0.438***	0.381***	0.792***	0.541***
	(0.159)	(0.147)	(0.165)	(0.154)	(0.112)	(0.103)	(0.192)	(0.137)
$lnmktact_{c,l}$	0.452***	0.472***	0.438***	0.400***	0.551***	0.462***	0.371**	0.427***
	(0.116)	(0.106)	(0.110)	(0.112)	(0.157)	(0.144)	(0.149)	(0.149)
$pubexp_{j,l}$	−0.018**	−0.012*	−0.021***	−0.016**	−0.013*	−0.009	−0.007	−0.008
	(0.007)	(0.007)	(0.007)	(0.007)	(0.008)	(0.007)	(0.007)	(0.007)
$competition_{c,t}$	−0.329*	−0.250	−0.305*	−0.223	−0.874***	−0.761***	−0.717***	−0.781***
	(0.171)	(0.156)	(0.159)	(0.157)	(0.168)	(0.155)	(0.160)	(0.158)
$screenshots_{i,l}$	0.230***	0.225***	0.228***	0.210***	0.248***	0.254***	0.177***	0.209***
	(0.050)	(0.048)	(0.046)	(0.047)	(0.048)	(0.048)	(0.044)	(0.045)
$ageres_{i,l}$	−0.189**	−0.222***	−0.180**	−0.166**	−0.157*	−0.204**	−0.170**	−0.136*
	(0.088)	(0.082)	(0.084)	(0.083)	(0.087)	(0.083)	(0.082)	(0.078)
$multicategory_{i,l}$	−0.215	−0.240*	−0.223*	−0.181	−0.164	−0.099	−0.260**	−0.220*
	(0.136)	(0.125)	(0.126)	(0.127)	(0.136)	(0.129)	(0.120)	(0.127)
$multiplatform_{i,l}$	0.879***	0.917***	0.815***	0.845***	0.957***	0.978***	0.850***	0.904***
	(0.167)	(0.152)	(0.159)	(0.157)	(0.178)	(0.158)	(0.161)	(0.169)
$appsize_{i,l}$	0.002***	0.002***	0.002***	0.002***	0.002***	0.002***	0.002***	0.002***
	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)	(0.000)
$updates_{i,t}$	1.108***	1.064***	1.177***	1.142***	1.100***	1.043***	1.075***	1.119***
	(0.140)	(0.127)	(0.116)	(0.118)	(0.127)	(0.117)	(0.126)	(0.125)
$sincelaunch_{i,t}$	0.012***	0.012***	0.011***	0.011***	0.013***	0.014***	0.014***	0.013***
	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)	(0.002)
$diversity_{i,l} \times richness_{i,l}$		0.017***				0.043***		
		(0.003)				(0.010)		
$lnmkact_{c,l} \times diversity_{i,l}$			−0.048***				−0.091***	
			(0.016)				(0.032)	
$lnmkact_{c,l} \times richness_{i,l}$				−0.009*				−0.080**
				(0.005)				(0.031)
Constant	4.304***	4.593***	3.975***	4.539***	3.970***	4.888***	5.795***	5.070***
	(1.187)	(1.082)	(1.072)	(1.086)	(1.355)	(1.267)	(1.250)	(1.305)
Subcategory FE	YES	YES	YES	YES	YES	YES	YES	YES
Observations	1,782	1,782	1,782	1,782	1,782	1,782	1,782	1,782
R-squared	0.287	0.298	0.287	0.285	0.269	0.278	0.273	0.274

Robust standard errors clustered by publisher in parentheses (*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Table A.3: OLS Regression Results

	(1)	(2)	(3)	(4)
$diversity_{i,l}$	−0.195***	−0.295***	−0.181***	−0.194***
	(0.070)	(0.070)	(0.068)	(0.070)
$richness_{i,l}$	0.069 **	−0.045	0.069 **	0.072***
	(0.027)	(0.039)	(0.027)	(0.027)
$lnmkttrend_{c,l}$	0.302	0.297	0.305	0.302
	(0.190)	(0.190)	(0.193)	(0.192)
$lnmktact_{c,l}$	0.132	0.127	0.409	0.231
	(0.139)	(0.138)	(0.267)	(0.208)
$pubexp_{j,l}$	−0.017	−0.017	−0.017	−0.017
	(0.019)	(0.018)	(0.019)	(0.019)
$competition_{c,t}$	−0.602***	−0.553***	−0.608***	−0.603***
	(0.208)	(0.204)	(0.211)	(0.209)
$screenshots_{i,l}$	0.202***	0.210***	0.200***	0.202***
	(0.064)	(0.065)	(0.064)	(0.064)
$ageres_{i,l}$	−0.272***	−0.266***	−0.274***	−0.272***
	(0.101)	(0.099)	(0.101)	(0.101)
$multicategory_{i,l}$	−0.381 **	−0.345*	−0.375 **	−0.376 **
	(0.188)	(0.182)	(0.188)	(0.187)
$multiplatform_{i,l}$	0.907***	0.908***	0.906***	0.908***
	(0.238)	(0.227)	(0.239)	(0.238)
$appsize_{i,l}$	0.002***	0.002***	0.002***	0.002***
	(0.000)	(0.000)	(0.000)	(0.000)
$updates_{i,t}$	0.939***	0.921***	0.942***	0.940***
	(0.213)	(0.204)	(0.213)	(0.213)
$sincelaunch_{i,t}$	0.012***	0.013***	0.012***	0.012***
	(0.003)	(0.003)	(0.003)	(0.003)
$diversity_{i,l} * richness_{i,l}$		0.015***		
		(0.004)		
$lnmkact_{c,l} * diversity_{i,l}$			−0.056	
			(0.042)	
$lnmkact_{c,l} * richness_{i,l}$				−0.011
				(0.015)
Constant	6.894***	7.248***	6.795***	6.853***
	(1.622)	(1.606)	(1.633)	(1.630)
Subcategory FE	YES	YES	YES	YES
Observations	1,782	1,782	1,782	1,782
R-squared	0.290	0.299	0.291	0.290
Number of Publishers	711	711	711	711

Robust standard errors clustered by publisher in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Table A.4: IV Regression Results on Truncated Sample

	(1)	(2)	(3)	(4)
$diversity_{i,l}$	−0.209***	−0.401***	−0.373***	−0.357***
	(0.079)	(0.069)	(0.065)	(0.057)
$richness_{i,l}$	0.060 * *	−0.104***	0.120***	0.117***
	(0.024)	(0.031)	(0.021)	(0.021)
$lnmkttrend_{c,l}$	0.373***	0.265 * *	0.467***	0.680***
	(0.119)	(0.110)	(0.136)	(0.193)
$lnmktact_{c,l}$	0.471***	0.336 * *	0.288*	0.297 * *
	(0.161)	(0.147)	(0.149)	(0.150)
$pubexp_{j,l}$	−0.007	−0.002	−0.002	−0.002
	(0.007)	(0.007)	(0.007)	(0.007)
$competition_{c,t}$	−0.873***	−0.689***	−0.728***	−0.743***
	(0.171)	(0.159)	(0.159)	(0.161)
$screenshots_{i,l}$	0.210***	0.224***	0.161***	0.146***
	(0.048)	(0.048)	(0.045)	(0.044)
$ageres_{i,l}$	−0.148*	−0.217***	−0.081	−0.105
	(0.088)	(0.082)	(0.079)	(0.083)
$multicategory_{i,l}$	−0.123	−0.115	−0.199	−0.204*
	(0.136)	(0.123)	(0.122)	(0.120)
$multiplatform_{i,l}$	0.940***	0.876***	0.840***	0.839***
	(0.169)	(0.150)	(0.157)	(0.153)
$appsize_{i,l}$	0.002***	0.002***	0.002***	0.002***
	(0.000)	(0.000)	(0.000)	(0.000)
$updates_{i,t}$	0.974***	0.923***	0.964***	0.931***
	(0.126)	(0.111)	(0.121)	(0.123)
$sincelaunch_{i,t}$	0.015***	0.016***	0.014***	0.015***
	(0.002)	(0.002)	(0.002)	(0.002)
$diversity_{i,l} \times richness_{i,l}$		0.025***		
		(0.003)		
$lnmkact_{c,l} \times diversity_{i,l}$			−0.031***	
			(0.010)	
$lnmkact_{c,l} \times richness_{i,l}$				−0.087***
				(0.031)
Constant	4.228***	6.094***	6.428***	6.417***
	(1.373)	(1.251)	(1.285)	(1.252)
Subcategory FE	YES	YES	YES	YES
Observations	1,724	1,724	1,724	1,724
R-squared	0.247	0.259	0.243	0.244
Number of Publishers	687	687	687	687

Robust standard errors clustered by publisher in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Table A.5: IV Regression Results on Non-Holiday Apps Sample

	(1)	(2)	(3)	(4)
$diversity_{i,l}$	−0.244***	−0.400***	−0.361***	−0.359***
	(0.082)	(0.069)	(0.071)	(0.061)
$richness_{i,l}$	0.072***	−0.077 * *	0.126***	0.122***
	(0.025)	(0.033)	(0.022)	(0.021)
$lnmkttrend_{c,l}$	0.427***	0.366***	0.589***	0.700***
	(0.119)	(0.111)	(0.139)	(0.202)
$lnmktact_{c,l}$	0.517***	0.441***	0.364 * *	0.367 * *
	(0.160)	(0.147)	(0.149)	(0.149)
$pubexp_{j,l}$	−0.010	−0.003	−0.006	−0.006
	(0.008)	(0.007)	(0.007)	(0.007)
$competition_{c,t}$	−0.875***	−0.741***	−0.773***	−0.765***
	(0.174)	(0.162)	(0.161)	(0.163)
$screenshots_{i,l}$	0.237***	0.246***	0.180***	0.168***
	(0.048)	(0.048)	(0.045)	(0.045)
$ageres_{i,l}$	−0.189 * *	−0.234***	−0.161 * *	−0.183 * *
	(0.088)	(0.082)	(0.080)	(0.083)
$multicategory_{i,l}$	−0.153	−0.142	−0.239 * *	−0.241 * *
	(0.135)	(0.123)	(0.120)	(0.120)
$multiplatform_{i,l}$	1.039***	1.004***	0.971***	0.949***
	(0.173)	(0.156)	(0.163)	(0.161)
$appsize_{i,l}$	0.002***	0.002***	0.002***	0.002***
	(0.000)	(0.000)	(0.000)	(0.000)
$updates_{i,t}$	1.069***	1.031***	1.082***	1.035***
	(0.124)	(0.112)	(0.124)	(0.123)
$sincelaunch_{i,t}$	0.012***	0.013***	0.012***	0.012***
	(0.002)	(0.002)	(0.002)	(0.002)
$diversity_{i,l} \times richness_{i,l}$		0.022***		
		(0.003)		
$lnmkact_{c,l} \times diversity_{i,l}$			−0.035***	
			(0.010)	
$lnmkact_{c,l} \times richness_{i,l}$				−0.082 * *
				(0.033)
Constant	4.409***	5.583***	6.323***	6.344***
	(1.389)	(1.261)	(1.288)	(1.265)
Subcategory FE	YES	YES	YES	YES
Observations	1,762	1,762	1,762	1,762
R-squared	0.278	0.288	0.277	0.277
Number of Publishers	710	710	710	710

Robust standard errors clustered by publisher in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Appendix B

Essay 2 Robustness Check Results

We present additional results using alternative variable operationalizations and model specifications.

Table B.1: Difference-in-Differences Estimation Results

DV: $lndown_{ijt}$	(1)	(2)	(3)	(4)
$after_t$	$-0.023**$	$-0.034**$	$-0.024*$	-0.230
	(0.009)	(0.016)	(0.014)	(0.249)
$treat_{ij}$	$1.372***$	$1.192***$	$1.910***$	-0.353
	(0.199)	(0.183)	(0.259)	(1.226)
$after_t \times treat_{ij}$	$0.051**$	$0.096***$	$0.058**$	-0.054
	(0.020)	(0.034)	(0.029)	(0.533)
$upsd_{ijt}$	0.001	$-0.007**$	0.001	0.001
	(0.001)	(0.003)	(0.001)	(0.001)
hhi_{ct}	0.548	0.546	0.548	0.545
	(0.447)	(0.447)	(0.447)	(0.449)
$rating_{ijt}$	0.144	0.145	0.144	0.144
	(0.164)	(0.164)	(0.164)	(0.164)
$mkttr_{ct}$	$0.424***$	$0.424***$	$0.424***$	$0.334***$
	(0.031)	(0.031)	(0.031)	(0.068)
$sincelaunch_{ijt}$	$-0.004***$	$-0.004***$	$-0.004***$	$-0.004***$
	(0.000)	(0.000)	(0.000)	(0.000)
$lcstage2_{ijt}$	$0.955***$	$0.955***$	$1.954***$	$0.955***$
	(0.183)	(0.183)	(0.241)	(0.183)
$lcstage3_{ijt}$	0.006	0.005	$0.349*$	0.006
	(0.114)	(0.114)	(0.199)	(0.114)
$after_t \times upsd_{ijt}$		$0.002*$		
		(0.001)		
$treat_{ij} \times upsd_{ijt}$		$0.016***$		
		(0.006)		
$after_t \times treat_{ij} \times upsd_{ijt}$		$-0.006***$		
		(0.002)		
$after_t \times lcstage2_{ijt}$			0.013	
			(0.026)	
$after_t \times lcstage3_{ijt}$			-0.005	
			(0.019)	
$treat_{ij} \times lcstage2_{ijt}$			$-1.960***$	
			(0.265)	
$treat_{ij} \times lcstage3_{ijt}$			$-0.685***$	
			(0.265)	
$after_t \times treat_{ij} \times lcstage2_{ijt}$			$-0.088**$	
			(0.044)	
$after_t \times treat_{ij} \times lcstage3_{ijt}$			0.007	
			(0.045)	
$after_t \times mkttr_{ct}$				0.019
				(0.022)
$treat_{ij} \times mkttr_{ct}$				0.155
				(0.108)
$after_t \times treat_{ij} \times mkttr_{ct}$				0.009
				(0.047)
Constant	-1.114	-1.032	$-1.383*$	-0.108
	(0.815)	(0.813)	(0.835)	(1.044)
App FE	YES	YES	YES	YES
Month FE	YES	YES	YES	YES
Weekday FE	YES	YES	YES	YES
Observations	20,254	20,254	20,254	20,254
R-squared	0.666	0.668	0.674	0.667
Number of appid	433	433	433	433

Robust standard errors clustered by app in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Table B.2: Survival Analysis Results

DV: Duration	(1) cox PH	(2) exponential RE	(3) weibull RE
Stage1 Update	0.134 * **	0.142 * **	0.118 * **
	(0.062)	(0.061)	(0.055)
Stage2 Update	0.174 * **	0.187 * **	0.152 * **
	(0.010)	(0.089)	(0.079)
Stage3 Update	0.633	0.601	0.650
	(19.070)	(0.326)	(0.326)
upsd	1.011 * **	1.014 * **	1.010 * *
	(0.003)	(0.004)	(0.004)
Stage1 Patches	0.085 * **	0.088 * **	0.075 * **
	(0.000)	(0.053)	(0.047)
Stage2 Patches	0.111 * **	0.117 * **	0.099 * **
	(0.024)	(0.077)	(0.069)
Stage3 Patches	0.000 * **	0.000 * **	0.000 * **
	(0.000)	(0.000)	(0.000)
Log Length	0.998	0.998*	0.998*
	(0.001)	(0.001)	(0.001)
Avg. Inter time	1.000	1.001	0.999
	(0.000)	(0.001)	(0.001)
First day hype	0.867 * **	0.861 * *	0.862 * *
	(0.044)	(0.059)	(0.061)
App size	0.924	0.914	0.926
	(0.083)	(0.182)	(0.187)
HHI	0.697	0.294	0.609
	(1.069)	(0.440)	(0.851)
Age Restriction	1.122	1.128	1.114
	(0.083)	(0.152)	(0.152)
Avg. DAU	1.000	1.000	1.000
	(0.000)	(0.000)	(0.000)
mkttr	0.865 * **	0.875	0.861
	(0.028)	(0.101)	(0.098)
Observations	821,841	821,841	821,841
Number of appid	2,157	2,157	2,157

Robust standard errors clustered by app in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)

Table B.3: 5 Stage Life Cycle Stage Estimation Result

DV: $lndown_{ijt}$	Coefficient	Standard Error
$after_t$	0.029	(0.055)
$after_t \times treat_{ij}$	0.074 ∗ ∗	(0.025)
$lcstage2_{ijt}$	−0.291 ∗ ∗	(0.114)
$lcstage3_{ijt}$	−0.230	(0.145)
$lcstage4_{ijt}$	0.098	(0.078)
$lcstage5_{ijt}$	0.301 ∗ ∗	(0.097)
$after_t \times lcstage2_{ijt}$	0.000	(0.014)
$after_t \times lcstage3_{ijt}$	−0.044 ∗ ∗	(0.016)
$after_t \times lcstage4_{ijt}$	−0.052 ∗ ∗	(0.019)
$after_t \times lcstage5_{ijt}$	−0.039 ∗ ∗	(0.012)
$treat_{ij} \times lcstage2_{ijt}$	1.185 ∗ ∗	(0.284)
$treat_{ij} \times lcstage3_{ijt}$	0.346	(0.336)
$treat_{ij} \times lcstage4_{ijt}$	−0.276	(0.237)
$treat_{ij} \times lcstage5_{ijt}$	−1.209 ∗ ∗	(0.280)
$after_t \times treat_{ij} \times lcstage2_{ijt}$	−0.189 ∗ ∗	(0.048)
$after_t \times treat_{ij} \times lcstage3_{ijt}$	−0.041	(0.044)
$after_t \times treat_{ij} \times lcstage4_{ijt}$	−0.027	(0.046)
$after_t \times treat_{ij} \times lcstage5_{ijt}$	0.008	(0.039)
$upsd_{ijt}$	−0.000	(0.000)
hhi_{ct}	−0.146∗	(0.088)
$rating_{ijt}$	−0.158∗	(0.084)
$mkttr_{ct}$	0.420 ∗ ∗	(0.026)
$sincelaunch_{ijt}$	0.003	(0.004)
Constant	0.556	(0.909)
Weekday FE	YES	
Month FE	YES	
Observations	20,254	
R-squared	0.221	
Number of updateid	4,052	

Robust standard errors clustered by app in parentheses
(*** $p < 0.01$, ** $p < 0.05$, * $p < 0.10$)