

Strategic Signals in the App Economy

An Empirical Study of *Google Play Store*



The University of
Nottingham

UNITED KINGDOM • CHINA • MALAYSIA

Jianhua Shao

Horizon Digital Economy Research

University of Nottingham

This dissertation is submitted for the degree of

Doctor of Philosophy

April 2016

I would like to dedicate this thesis to my my family and loving parents ...

Abstract

The dominance of *Android* and *iOS* have created a duopoly in the smartphone market. In this context, both *Android* and *iOS* need to select high quality apps from its massive number of app developers to recommend to app consumers. Generally speaking, when the market is small, the platform like *Android* and *iOS* can allocate resources to manual test the app quality for selection; but as the market size increased, the required resource for selection is also increased and would challenge the platform's capability, thus the platform faces the *selection difficulty*. Unlike *iOS*, *Android* adopts an open strategy to compete against *iOS* for innovative developers. This strategy has led to inadvertent consequences including: "fragmentation" issues, weakened governance power, entry of non-competent developers and, notably, vetting of the quality of apps and developers.

An app market is a two-sided platform based market. Research in platforms ([Gawer and Cusumano, 2013](#)) have primarily focused on what the technology platform owners should do. They assume that developers will equally respond to their platform strategy, for example, leaving the market when the platform owner sets higher barriers or enters when the platforms are more friendly. A few recent platform literatures ([Boudreau, 2012](#); [Eisenmann et al., 2011](#)) start to look at developers' *unequal behaviour*, such as developers with different competence would have different *switching behaviours* between different markets. However, in a duopoly setting, developers have limited alternatives but continue to work with both markets. The duopoly setting is interesting as the communication of app quality is critical for competition, and this thesis aims to examine the competent developers' strategic behaviours from the lens of signal theory.

I argue that competent developers will engage in costly actions to compete against non-competent counterparts in such a duopoly context. For example, competent developers will spend time to make their apps to be unique and innovative in the *Android* app market, or enhance the technology used in their apps to protect app consumers' privacy, and other more behaviours discussed in the thesis. It is inspired to understand these unique behaviour by competent developers as strategic signals to communicate their innovative behaviour in the app economy. Information asymmetry ([Stiglitz, 2002](#)) is believed to cause the difficulty for platform selection based on the developer's innovativeness in such a crowded space. Based

on the signalling theory (Spence, 1973), the thesis develops a signalling selection model to understand the motivation and implication of these strategic signals.

iOS is not open and is, therefore, difficult to collect research data. To study strategic signals, I collected a panel data set composed of 93% of all apps and their developers in the *Google Play Store* which is the official *Android* app market. The thesis firstly filtered out all developers who are featured as "Top Developer" by *Google* in the *Google Play Store*. They are treated as innovative developers. The thesis then filtered out all non-featured developers who are relevant to these innovative developers and treated them as non-innovative developers. The thesis qualitatively and quantitatively analyses innovative developers' strategic behaviour in the *Google Play Store*.

It is found that these strategic signals are unlikely to be generated by non-innovative developers because they have a higher cost. The cost could be due to, for example, the R&D on technology advancement, the challenge of creative business model design, etc. It provides the platform owner with an opportunity to observe these signals only uniquely by innovative developers, and feedback with featuring awards. These awards would bring huge customer access and generate large long-term benefits on business performance. The benefits motivate the innovative developers to continue on these strategic signals generation, which increases the availability of innovative apps. The signals can also be observed by other developers and motivate non-featured developers to learn from featured innovative developers to increase their featuring opportunity, which is studied as the peer effect in the thesis. This peer effect in learning enables the opportunity for the platform owner to develop and maintain the app market into an ecosystem. Rather than managing all developers, as emphasised by existing platform strategies, this thesis argues the platform owner can focus on selecting a small number of innovative developers to influence the large developers community with the signalling selection model.

The contribution of this thesis is to shift the focus of platform strategy from platform centric to developer centric. I argue that innovative developers would behave differently from others. Their strategic behaviour serves as a signal for the platform owner to vet the quality of innovativeness. The thesis studied a duopoly context of app economy where switching is limited to two dominant platforms. It develops a signalling selection model to solve the selection difficulty of innovativeness in such a unique context. According to my knowledge, it is the first time the signalling theory has been applied to the app economy. The selection of innovativeness is friendly to newly entranced developers which is less focused by existing platform centric strategies. The research on app economy is still rare but the importance of the app economy is significantly increased in daily life. The large-scale data set, methodology and results from this research should be valuable for future research in the app economy.

Acknowledgements

There are too many people I need to acknowledge. Without their contribution, this research could not be finished.

Foremost, I would like to express my sincere gratitude to my supervisors team: Dr. George Kuk (innovation strategy), Dr. Richard Mortier (computer science), Dr. Rodion Skovoroda (cooperate governance). They provide me continuous support of my PhD study and research, and I want to appreciate for their patience, motivation, enthusiasm, and immense knowledge. Their guidance help me in all the time of research and writing of this thesis. I could not have imagined having a better advisor team for my PhD study.

Secondly, I would like to thank my funding body (EPSRC, Digital Economy Research Program), hosting university (University of Nottingham), and all colleagues in my institution (Horizon Digital Economy Research Institute and Doctoral Training Centre). They provide kindly support, facility and opportunity, and enable me to concentrate on the research.

Thirdly, I would like to thank to the rest of my thesis committee: Dr. Jonathan Tan (Final viva, Business School, University of Nottingham) and Prof. Stefan Haeffliger (Final viva, Cass Business School, City University London), Prof. Thorsten Chmura (Third Year viva, Business School, University of Nottingham), Dr Michael Brown (Second Year viva, Horizon Digital Economy Research Institute, University of Nottingham). I appreciate for their encouragement, insightful comments, and hard questions.

Fourthly, I have a big list of friends to thank for their friendship during my PhD. Particularly, I like to thank Chaoyu Ye and Huijing Yu as lifetime friends and also for their encouragement during the 2015 hard time; Hongyan Yan and Dr. Su Chen during my two visits in China Mobile Research Institute; Dr. Marina Moreira for kindly thesis review and paper collaboration; Matthew Terrell for internship and conference; Clare Simcock for final version proof-reading help; Andrea Tomlinson for administration support.

In the end and particularly important, I want to thank to my lovely family: Wenwen Bao, Mingye Shao, Sunlian He, Shugen Shao, Lidan Shao, Lingcan Bao, Xianmei Xu and Xuyi Bao. You always bring me happiness and spiritually help me to recovery quickly from the depress during my PhD. It is all because of you so that I can earn the PhD title while building a small family and have a little girl. I am proud to have the kinship.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 10,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Jianhua Shao

April 2016

Table of contents

List of figures	xii
List of tables	xiv
1 Introduction	1
1.1 The Duopoly	1
1.1.1 The App Markets and Massive Adoption	2
1.1.2 Competing For Innovative Developers	2
1.1.3 Existing Attempts Of Innovation Management	3
1.1.4 Limitations of Platform Centric Management Strategies	4
1.1.5 The Shift to Developer Centric Management Strategies	4
1.1.6 Summary	6
1.2 Thesis Overview	7
1.2.1 Research Questions	7
1.2.2 Background	8
1.2.3 Literature Review	8
1.2.4 Theory Model	9
1.2.5 Research Method	9
1.2.6 Qualitative Study	10
1.2.7 Quantitative Study	10
1.2.8 Summary, Limitation and Contribution	11
2 Research Questions	13
2.1 Motivation	14
2.1.1 Two-Sided Market	14
2.1.2 App Market	15
2.1.3 App Economy	16
2.2 Problem	17

2.2.1	Research Interests	17
2.2.2	Existing Platform Strategies	20
2.2.3	Strategic Behaviour	22
2.3	Solution	24
2.3.1	Signalling Selection Strategy	24
2.3.2	Research Questions	27
2.3.3	Research Method	27
2.3.4	Solution Summary	28
3	Background	30
3.1	App Economy Researches	30
3.2	App Economy Characteristics	33
3.2.1	Installed User Base	33
3.2.2	Ecosystem	34
3.2.3	Competition	34
3.2.4	Cheap	35
3.2.5	Speed	36
3.3	<i>The Google Play Store</i>	37
3.3.1	History of <i>Android</i> and the <i>Google Play Store</i>	38
3.3.2	Development	39
3.3.3	Fragmentation	40
3.3.4	Competition	43
3.3.5	Innovation in the App Economy	44
3.3.6	Problem of Competition in the App Market	45
3.4	Information Asymmetry in App Market	47
3.5	Summary of Problem	49
4	Literature Review	51
4.1	Platform Strategy	51
4.1.1	Platform Thinking	52
4.1.2	Attraction For Complements	52
4.1.3	Differentiation Of Complements	54
4.1.4	Open Platform Strategies	56
4.2	Strategic Behaviours	57
4.2.1	Competition	58
4.2.2	Innovation Strategies	59
4.2.3	Open Source Software Innovation	63

4.2.4	Open Innovation	66
4.2.5	User Innovation	69
4.2.6	User Entrepreneurs	71
4.2.7	App Innovation	73
4.3	Signalling Theory	82
4.3.1	Information Asymmetry	83
4.3.2	Signalling Theory	87
4.4	Summary of Literatures	90
5	Model Development	92
5.1	Model Observation	92
5.1.1	Openness Develops Crowdedness	92
5.1.2	Developers Compete For Featuring Selection Opportunity	93
5.1.3	Information Asymmetry Increases The Selection Difficulties	93
5.2	Signalling Selection Model	94
5.2.1	What Constitutes A Signal?	94
5.2.2	How Are Signals Observed?	97
5.2.3	Summary of Model	98
6	Research Method	100
6.1	Data Source & Collection	100
6.1.1	Fundamental Data in App Level	101
6.1.2	Fundamental Data In The Developer Level	103
6.1.3	Fundamental Data In Permission Level	104
6.1.4	Additional Data Variables From <i>AppBrain.com</i>	105
6.1.5	Additional Data Variables From <i>WayBack Machine</i>	105
6.1.6	Collection Method	106
6.1.7	Data Overview	107
6.2	Qualitative Research Methods	108
6.3	Quantitative Research Methods	109
6.3.1	Panel Data Analysis	109
7	Qualitative Analysis	111
7.1	App Permission	111
7.2	User Data	113
7.3	Privacy Policy	115
7.4	How Do Developers Innovate With User Data?	117

7.5	Business Model On User Data	121
7.5.1	Monetisation on User Data	122
7.5.2	Value Creation On User Data	123
7.5.3	User Data Business Model Components	124
7.5.4	Business Model Analysis On 100 Featured Developers	125
7.6	Why Does The Developer Demand The Same User Data In Diversification?	127
7.6.1	Developers Do Not Use User Data	128
7.6.2	A Story Of An Independent Developer	128
7.6.3	Developers Who Only Use <i>system</i> User Data	130
7.6.4	Developers Using All Types Of User Data	131
7.7	Why Do Developers Not Diversify?	132
7.8	Summary Of Qualitative Study	134
8	Quantitative Analysis	136
8.1	Comparison Groups	136
8.2	Platform Selection	138
8.2.1	Dependant Variables	139
8.2.2	Uniqueness	139
8.2.3	Diversification	142
8.2.4	Following	144
8.2.5	Control Variables	147
8.2.6	Unit For Analysis	148
8.2.7	Statistical Method	149
8.2.8	Statistical Results	149
8.3	Peer Recognition	151
8.3.1	Dependant Variable	153
8.3.2	Independent Variables	153
8.3.3	Control Variables	154
8.3.4	Statistical Method	155
8.3.5	Statistical Results	155
8.4	Discussion	156
9	Conclusion	160
9.1	Discussion	160
9.1.1	The Platform Owner's Reciprocity Function	160
9.1.2	The Chaos Of Crowdedness	161
9.1.3	The Communication In Platform Thinking	163

9.1.4	<i>Google's Strategy</i>	164
9.1.5	Developers' Strategy	166
9.1.6	Who To Follow?	169
9.2	The Answer To Research Question	170
9.2.1	what costly actions generate signals in app economy?	171
9.2.2	who can interpret signals in app economy?	171
9.2.3	what feedback would encourage signaling in the app economy?	171
9.3	Contributions	172
9.3.1	To App Economy	172
9.3.2	To Platform Strategy	172
9.3.3	To Strategic Behaviour	173
9.3.4	To Signaling Theory	174
9.3.5	To Others	174
9.4	Summary	174
9.5	Limitations	176
9.6	Future Works	177
References		181
Appendix A		202
A.1	Panel Data Setting	202
A.2	Python Code to Determine Developer's Featuring Time	203
A.3	Statistics Output	205
A.4	Other Tables	212
A.5	Other Figures	234
A.6	Terms Used In The Thesis	262

List of figures

A.1	The <i>Android</i> app market	234
A.2	Three stages of app economy	235
A.3	Differentiation of complements	236
A.4	The signaling selection strategy in app economy	237
A.5	Signaling Model in App Market	238
A.6	Who to follow and signal?	238
A.7	<i>Android</i> key events	239
A.8	Example of an app profile	240
A.9	User review and other meta data in an app profile	241
A.10	List of relevant apps in an app profile	242
A.11	Examples of user review comment in an app profile	243
A.12	Top apps in an example category	244
A.13	A screenshot of review meta data for an app	244
A.14	A developer who received two selection rewards	245
A.15	History of available <i>Android</i> apps in <i>Google Play Store</i>	245
A.16	<i>Google Play Store</i> 's front page	246
A.17	Example of lead user who tried to challenge the developer	247
A.18	Example of a developer profile	248
A.19	Permission requested by a particular app during installation	248
A.20	Web archive in Wayback Machine for a particular app	249
A.21	Distribution of counting of unique categories by featured developers	249
A.22	Network of featured developers and app categories	250
A.23	Distribution of featured developers and their apps across app categories	251
A.24	Uniqueness change of apps by featured developers in non-game categories	252
A.25	Uniqueness change of apps by featured developers in game categories	253
A.26	Distribution of Permission Requested by Apps	254
A.27	Monetization of user data in app economy	255

A.28 Value creation with user data	255
A.29 Comparison groups	256
A.30 Unit diagram of multilevel analysis in the study	257
A.31 Market Performance Measurement	257
A.32 Uniqueness	258
A.33 Diversification	258
A.34 Following	259
A.35 Benchmark on following featured vs. non-featured developers	259
A.36 Curvilinear relationship to following featured developers	260
A.37 Benchmark on different selection status after following non-featured developers	260
A.38 Future works around app economy from this thesis	261

List of tables

8.1	An illustration of how <i>unique</i> is calculated for two apps	141
A.1	Variable descriptions	205
A.2	Descriptive Statistics	207
A.3	Correlation matrix for Signalling Model on platform	208
A.4	Signalling Model on platform selection	209
A.5	Correlation matrix for Signalling Model on peer recognition	210
A.6	Signalling Model on peer recognition	211
A.7	Key literatures: Platform Strategy	212
A.8	Key literatures: App Economy	213
A.9	Key literatures: Signaling Theory	213
A.10	Key literatures: Strategic Behaviour	214
A.11	The history of Android and Google Play Store	215
A.11	The history of Android and Google Play Store - continued	216
A.11	The history of Android and Google Play Store - continued	217
A.11	The history of Android and Google Play Store - continued	218
A.11	The history of Android and Google Play Store - continued	219
A.12	All non-game categories in Google Play Store	219
A.13	All game categories in Google Play Store	220
A.14	Official app download information	220
A.15	List of permission groups defined by Android platform	221
A.16	Distribution of category counting of featured developers	222
A.17	Study on featured developers' business model on user data	223
A.17	Study on featured developers' business model on user data - continue	224
A.17	Study on featured developers' business model on user data - continue	225
A.18	Developers who adopt diversification but demand the same user data	226
A.18	Developers who adopt diversification but demand the same user data - continue	227
A.18	Developers who adopt diversification but demand the same user data - continue	228

A.18 Developers who adopt diversification but demand the same user data - continue	229
A.19 Developers who only have one app	230
A.19 Developers who only have one app - continue	231
A.19 Developers who only have one app - continue	232
A.19 Developers who only have one app - continue	233

Chapter 1

Introduction

This chapter firstly introduces the context of study in this thesis, which is the current duopoly formed by *Android* and *iOS*. It briefly introduces the evolution in app markets since the appearance of the "app store" idea by *Android* and *iOS*. The "app store" generates massive adoptions by developers which significantly develops the duopoly, but also increases the difficulty of management on innovation. Both *Android* and *iOS* are competing for innovative developers to maintain its high innovativeness. Unlike other types of markets structure, developers cannot switch between different markets for an early move advantage, so they have demands to communicate their innovativeness in order to win the competition in the duopoly. However, information asymmetry creates the difficulty for this communication. The thesis intends to study the strategic behaviour which is unique to innovative developers as strategic signals to understand developers' innovativeness. With the empirical examination of a particular *Android* app market which is the *Google Play Store*, the thesis shifts the focus of platform strategies from platform centric to developer centric by looking at developers' unequal behaviour. The rest of this introduction provides an overview of the whole thesis.

1.1 The Duopoly

Nowadays, when people talk about a smartphone, they generally refer to either *Android* or *iOS*. Since the *iOS* and *Android* opened their app stores or app markets in 2008, there has been a revolutionary change in the smartphone market. Previous dominant players in the PDAs market, like *Nokia*, *Blackberry* and *Palm*, have lost their leading positions in the platform wars with *Android* and *iOS*. Industry researchers like *comScore*¹ reported that *Android* and *iOS* took over 93% of the total market share in total in 2014.

¹<http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-January-2015-US-Smartphone-Subscriber-Market-Share>

1.1.1 The App Markets and Massive Adoption

The unique features of the app store idea have significantly developed the massive user adoption. Unlikely in the PDAs age, apps in *iOS* and *Android* are retailed in their central online stores. Ordinary users can visit the store easily and only need to click a few buttons in their mobile devices to complete the app installation. This easy-to-use feature removes the technology boundary which existed for PDAs users and liberates the broader adoption of users (West and Mace, 2010). The second unique feature is the low costs of apps to users. Most apps in *iOS* and *Android* are financially zero priced (Liu et al., 2014), which means users can have them for free. For the other paid-for apps, *Wall Street Journal*² reported the average price of current apps is only a few dollars. This low-price feature removes the financial barrier for user adoption. The third unique feature is the availability of apps. Public information disclosed that *Google Play Store* had 675,000 apps³ in September 2012, 1,300,000 apps in July 2014 and 1,430,000⁴ apps in December 2014. This figure is still increasing, and users would generally have lots of choices to meet their needs on apps.

The massive adoption has not only happened on the users side but also on the developers side. The massive user adoption attracts the app developers to create and sell their apps in the app markets of *iOS* and *Android*. The virtual nature of an app market removes the geographical restriction and enables developers to easily trade their apps with global users in its market. Some developers are motivated to work in the middleware or infrastructure with their specialists, which expand the availability of supporting tools and infrastructure services and lower down the technology barrier in app development. The low technology barrier enables many new or part-time developers to enter into the market. Nowadays, companies see the need in having a mobile app on *Android* or *iOS*, as much as they have a website for the companies before (Wilcox and Voskoglou, 2015).

1.1.2 Competing For Innovative Developers

The massive adoption by developers leads to the difficulty of platform management. Different kinds of developers are attracted into the market to enjoy the benefits of massive user adoption. Some developers are profit-maximum driven (Carlin et al., 2001). Their apps may bring low user experience or even be malicious to damage users' interests for profits (Chia et al., 2012). Some developers may also make copies of other developers' popular apps and then repack to sell under their own names (Zhou et al., 2012a). This piracy activity would damage to the

²<http://www.wsj.com/articles/SB10001424127887323511804578300183961009070>

³<http://officialandroid.blogspot.co.uk/2012/09/google-play-hits-25-billion-downloads.html>

⁴<http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/>

interests of innovative developers, and discourage the original work and culture of innovation. Therefore, the platform owner, such as *Google* in the *Google Play Store*, has demands to differentiate the quality of apps and their developers in the market.

The differentiation of quality is not only demanded by the platform owner, but also the final users who are the app consumers. The massive number of apps in the market expand product availability, but also increases the cost of searching for desired apps. A search may return a long list of apps are very similar to one another, and users have to try all of them in order to find out which is the best if there is no mechanism to understand the quality prior to usage. In an app market with such a huge size, information asymmetry (Stiglitz, 2002) would significantly increase the difficulty on selection. Final users want a mechanism to easily understand the quality of the app and have a desire for a higher than average innovation in the market.

In order to compete with rivals, the platform needs innovative developers to increase its competitive advantage (Lin et al., 2011). Innovative developers are more likely to invest in innovation and create novel and popular apps for the platform in order to attract users. This is also partially explains (Ghose and Han, 2014) why both *Android* and *iOS* spend much of their resources each year hosting contests, organising conferences, developing tutorials and supporting innovative activities in app development. *Android* and *iOS* are competing for good developers, especially innovative developers.

1.1.3 Existing Attempts Of Innovation Management

However, the selection of innovativeness is not easy in such a crowded context. *Google* developed an artificial intelligent service called “Bouncer”⁵ to detect malware apps by regularly scanning each app to gain more insight in its quality. This technical approach is limited as many malicious behaviour are difficult to detect in source code especially if they pretend to “cheat” (Felt et al., 2011b). *Google* also adopted an approach of using audience participation by encouraging users to review and report upon the app quality. This non-technical approach is also limited, as most users are lack a technological background and, more importantly, they do not value highly about their micro behaviour or they are just too “lazy” to report (Chia et al., 2012).

Meanwhile, the open strategy adopted by *Android* to compete with *iOS* increases the difficulty of innovative selection and promotion. In order to promote the collaborative efforts, *Android* has been open sourced since its formation. This approach has significantly pushed the development of *Android* in the early stages. However, it generated a “fragmentation” issue where multiple versions of *Android* software and hardware systems existed in the markets at the

⁵<http://googlemobile.blogspot.co.uk/2012/02/android-and-security.html>

same time. *OpenSignal*⁶ reported there were 3997 distinct models of *Android* devices in 2012. It sets up a difficult for developers to target a specific device for innovation. It also increases the difficulty for the platform owner to filter the innovativeness with such large variety models available at one time.

1.1.4 Limitations of Platform Centric Management Strategies

The purpose of selection on innovativeness is to vet the quality. There is some important literature to argue the vetting of quality in two-sided markets (Cusumano, 2010; Rysman, 2009). The strategies, they argued, is to focus on how the platform owner should "control" the market which is platform centric by setting up barriers; for example, pricing, exclusive contracting, and other strategies as discussed in the later chapters. They assume that developers will respond the same to the platform strategy, which leads to some limitations; for example, a percentage of developers being filtered out can reduce the market size which will affect on the network effect, and some experienced non-innovative developers are difficult to filtered out even when setting up high barriers.

More importantly, platform centric is not friendly to new entry but potentially innovative developers. *VisionMobile* estimated in its 2014 report that at least 8,000 UK companies are engaged in the mobile app business (Pappas, 2014). However, one-man operations are common in a significant number of these companies (approximately 15%). Some users cannot find one app for their needs and feel other users may also have the same needs, so they create the apps and start the entrepreneurship (Shah and Tripsas, 2007). These new entrants, start-ups, or innovative users become important sources of innovation in the app economy, but they can create difficulties should there be a rising of the barrier. Therefore, developer centric studies are required to supplement a platform centric strategy.

1.1.5 The Shift to Developer Centric Management Strategies

Developer centric studies are being picked up by scholars, but most of them are focused on the developers' switching behaviour. Boudreau (2012) studied the developers' switching behaviour during the PDAs age. He examined the scenario that, as the market become crowded, the completion rate becomes higher while the profit margins become lower. So good developers will leave the current crowded market to join in other, less crowded, markets to maintain the leadership and high returns. Eisenmann et al. (2011) observed that developers' switching behaviour brings the opportunity for less popular platforms to learn and implement their unique features that popular platforms in order to attract good developers. These prior works inspired

⁶<http://opensignal.com/reports/fragmentation.php>

me to study an interesting but missing context: what if the platforms can learn from each other to become similar and developers have no option to switch?

The current duopoly of app markets is in the following context. *Android* and *iOS* form the duopoly of the app economy, and other platforms have a very tiny market share. Developers have to learn the platform technology in order to develop apps in a specific platform, and they have costs to update the apps as the platform technology is developing. For this reason, other smaller platforms are of very little interest to developers. It also means, even without lock-in, that developers have to target the duopoly. *Android* and *iOS* have become more and more similar to each other. They have similar financial barriers to developers; for example, having similar entry registration fee and commission rate. *Android* and *iOS* have also learnt from each other by having similar technology and market features. Some arguments said that the stage of *Android* has a lower percentage of users who are willing to pay for apps, but this situation has been significantly improved in the current duopoly context ⁷. *Android* and *iOS* are treated equally by developers. Meantime, cross-platform tools like *AppMethod* ⁸, services like *MBaaS*, and technologies like *HTML5*, have simply reduced the boundaries between the platforms. The gap between *Android* and *iOS* in this duopoly is blurring. All of this has confirmed that the developers' switching strategy is not efficient in this duopoly of app economy. Developers have to compete with many other developers on both *Android* and *iOS*, and it is necessary for them to communicate their innovative skills in the app market.

Information asymmetry has created the difficulty in selection on innovation, especially in such a crowded environment. Only developers themselves know how good they are, but all developers make efforts to persuade the market that their apps have the best qualities. Some innovative developers may not be very good at "advertising" their innovativeness thinking due to the limitation of their current resources. Some extreme developers may attempt to "cheat" the market with "unreal" information to claim their "fake" high quality. The platform owner and final users need to process the information given by developers to understand how good they really are. To solve this information asymmetry, innovative developers could make some costly actions to communicate their innovativeness. The opportunity cost associating to these costly actions is generally higher to non-innovative developers. Therefore, only innovative developers are likely to make these costly actions, which can be understood as strategic signals. The platform owner could observe these signals to understand whether a developer is innovative. Innovative developers could observe these signals to understand how they want to target in order to compete and learn for future development. This behaviour is what drives me to study the

⁷<http://uk.businessinsider.com/android-apple-revenue-digi-capital-china-ios-google-app-store-2015-4>

⁸<http://www.appmethod.com/>

motivation of strategic signals generation and the implication of strategic signal interpretation in order to understand developers' unequal behaviour in the duopoly of the app economy.

This thesis wants to study on the platform strategy for vetting the quality in such a unique duopoly context of the app economy. It would shift the focus of existing strategies from a platform centric view into a developer centric view by looking at the developer's unequal behaviour on innovation. Since switching strategies are inefficient in this context, innovative developers could behave differently from the rest to disclose their innovative behaviour to overcome the effects of information asymmetry. This enables the platform to observe these unique behaviours and filter out the innovative developers. I have chosen *Android* as the study base for three reasons. The first reason is that *Android* has a larger market share than *iOS*. *Gartner*⁹ reported that 1.9 million mobile phones has shipped in 2014, and more than one billion users are using *Android* devices in 2014. *Android* has become a "must-have" platform that app developers target. The second reason is due to the availability of data for research. *iOS* is not open which makes the effective data collection for valid research purposes difficult. In contrast to *iOS*, *Android* has richer public data and is, therefore, more friendly to data collection for academic research. The third reason is that *Google* as the platform owner, selects a number of developers to feature as "Top Developers" awards in the *Google Play Store*. I can treat them as recognized innovative developers and select their rivals in the *Google Play Store* to treat as non-innovative developers. This enables me to have two groups of developers to compare and contrast their strategic behaviour with different level of innovativeness.

1.1.6 Summary

The platforms, such as *Android* and *iOS*, are competing for "good" developers in order to build and maintain the sustainability of their platform-based markets. However, as the market becomes crowded, the possibility to identity each app's and its developer's quality became challenge. This challenge becomes extremely difficult, especially when the duopoly is formed. So the app developers are motivated to signal their quality to the platform to share the responsibility of market sustainability, and also compete for the selection opportunity in the platform. In order to understand the signaling between developers and the platform, we studied a two-levels signaling model. The first level is that developers signal to the platform to compete for the selection opportunity, and the second level is that developers signal to other developers to compete for the legitimacy to claim the market. This two levels model shifts the focus from platform-centric into developer-centric, and make a construct with following components: players (developers, platform, market), actions (signaling, feedback, follow), payoffs (featuring

⁹<http://www.gartner.com/newsroom/id/2645115>

selection, market performance, legitimacy and claiming the market), information (app permission, market lead time, platform featuring status transition) and timing of moves (app releasing, market entry).

In summary, this thesis attempts to examine the different strategic behaviours by developers having different innovative abilities. Innovative developers could make some unique actions to communicate their innovativeness to the public. These unique actions are associated with higher opportunity cost to the non-innovative developers. So the platform owner and other interesting individuals can treat these actions as strategic signals by innovative developers. This forms the generation and interpretation of strategic signals to communicate developers' innovativeness. This study on developers' unequal behaviour is particularly valuable to understand in the current duopoly of app economy. This is because app developers have no switching options and the markets in the duopoly are similar, as a result developers have to compete with their innovativeness and the platform owner needs to differentiate the innovativeness in order to protect the competitive advantage. To study strategic signals, I collected a panel data set composed of 93% of all apps and their developers in the *Google Play Store*. The rest of the chapters will discuss the details of research questions, methodologies, empirical findings and conclusions of this research.

1.2 Thesis Overview

To explain the research work, the thesis is organised using the following structure.

1.2.1 Research Questions

The next first chapter (Chapter 2) is to frame the research questions. It starts with a brief introduction of the app economy, and then introduces the problems generated by extreme competition in the app economy. To solve the problems, it firstly discusses the present situation and limitations of existing platform strategies summarised from the literature. The thesis then promotes the shift of focuses from platform centric to developer centric to solve the limitations of existing platform strategies. It then introduces the idea of strategic behaviour and is inspired to develop a signalling selection model based on signalling theory. The signalling selection model is observed from the *Google Play Store*'s featuring mechanism, and is developed with the aim of efficiently managing the ecological development in the app economy, especially when the duopoly of app markets is formed. Based on this theory building process, it aims to study three research questions around developers' strategic behaviour to understand the work

of the signaling selection model. The end of this chapter is to introduce how research methods are guided to answer the research questions.

1.2.2 Background

Chapter 3 starts with the background of the app economy. The app economy has started to appear in industry technology and academic technology research, but not often in academic business research. According to existing materials, the thesis summarises a few characteristics of app economy including having a large installed user base, having an established ecosystem, having high competition within and between app markets, having cheap price products and low cost production, and having a fast development speed. The chapter then concentrates on the introduction of the *Google Play Store* which is currently one of the most important app markets and also the primary app market to analyse in this thesis. The chapter introduces a complete history of the *Google Play Store*, how the *Google Play Store* adopted open strategy to grow and compete with *iOS* and also the consequence of open strategy which is *Android Fragmentation*. *Android Fragmentation* has affected the innovation speed and ecological development of the *Android* platform. Large scale and high competition enhances the information asymmetry and brings the problems to the innovation management in the app market. To solve this problem, an efficient selection mechanism is demanded.

1.2.3 Literature Review

Chapter 4 aims to find solutions by reviewing the existing literature. Existing platform strategies literature focuses on the attraction for complements to enhance the network effects to increase the market size, and also differentiation of complements to control the quality of complements in order to increase the platform competitive advantage. Meantime, open strategy helps *Android* to encourage the collaboration and aggregate the contributions, but also massively increases the competition and loses the control of entries in the app market.

Existing literature confirms the importance of innovation to platform ecological development. The chapter then summarises the commonly adopted innovation strategies, including product differentiation, product diversification, business model and learning in order to increase the competitive advantage. The thesis then attempts to find solutions by looking at the similar problems in software innovation, and found certain components in open source software innovation, open innovation and user innovation, user entrepreneurs. Compared to traditional software innovation, app innovation has its unique characteristics, for example, app innovation has strong network effects, infinite scale and scope, culture of recombination and reuse to solve

technology complexity, huge uncertainty and has skewed outcomes, low-cost and facilitated development, and low barriers to entry.

The literature review process finds the limitations of existing strategies to differentiate between the complements in innovation, as existing strategies concentrate on setting up the barriers to control the quality, a selection based strategy is demanded to differentiate the complements based on intrinsic innovativeness. Meanwhile, the available literatures provides theoretical foundations to explain the innovation process in the app economy and also the cause of general problems in ecological development, but had not solved the problem for selection which is enhanced by information asymmetry. Signalling theory has been suggested to solve the information asymmetry in other market domains, so the end of this chapter reviews the existing applications of signalling theory.

1.2.4 Theory Model

The background research helps to form a theory of a signalling selection model which is introduced in Chapter 5. It combined the signalling theory to solve the information asymmetry problem and selection mechanism to improve ecological development in the app economy. This chapter theoretically explains what constitutes a signal in the app economy and identified three major strategic behaviours by developers from the literature, including differentiation, diversification, and learning. The chapters also discusses how signals are observed in theory. The thesis finds that not only is *Google* interested in observing the signals, but also developers. In order to make efficient strategic behaviour, developers have to observe other developers' actions, especially to learn from the developers who are already selected to enjoy the great benefits. Different developers have different abilities to observe and interpret other developer's signal. Therefore, developers are intended to stay closer to other developers having similar or higher ability in the market. Users who purchase apps are in the other side of the app market. Their satisfaction influences the decision to evaluate the performance of apps. Some researchers work on "crowd wisdom", who also have an interest in observing the signal and influence of the *Android* development, but the majority of users lack of technical ability or motivation. Existing research has already confirmed that users would be more likely to listen to the selection decision made by the market operator, so the focus of this thesis is to understand the strategic behaviour by developers and market operators.

1.2.5 Research Method

In order to empirically examine the signaling selection model, a large scale of data collection on the *Google Play Store* has been made. Chapter 6 will discuss the complete research method. In

order to study the developers' strategic behaviour, the data was scraped to collect behaviour data for 1,340,505 apps and 232,397 developers which counted as 93% of total apps in 2014 in the *Google Play Store*. It also collected each selection decision made by *Google* towards apps and their developers as a feedback from market operators to signallers. *Featured* developers are the primary focus in this study. Based on the data collected, there are only 365 developers selected to be featured as "Top Developer" out of 232,397 developers, which shows the extremely difficulty for *featuring* opportunity and the competition for platform featuring is extremely high. These featured developers have, in total, produced 3542 apps. I ranked these apps in chronological order with their created time, and also ranked their developers in chronological order with the information about when they are *featured* or selected by the market operator. Data are prepared in panel data format and be ready for qualitative and quantitative analysis.

1.2.6 Qualitative Study

Chapter 7 discusses the qualitative study on *featured* developers' companies websites and privacy policies, in order to understand the trajectory of how they behave to be selected by *Google* as *featured* developers. The thesis finds out the importance of app permissions in developers' behaviour. App permissions are the key for a developer to implement certain functions and he needs to clearly declare the request in apps' source code which can be reviewed by anyone. It becomes an important resource for developers to technically learn from each another. More importantly, app permissions become the foundation for developers to acquire user data and innovate apps features. The developers' privacy policy confirmed that the positive association between user data and innovation. The priority for developers is to increase user base in the app economy. Monetisation in the app economy is heavily reliant upon user data and user base. The unique characteristics of the app economy makes the difference between a business model in the app economy and traditional economy. The patterns of strategic behaviour on user data have influenced the adoption of the business model by developers in app economy. This study provides evidence that developers are competing against each other, but they are also aware of each other. However, it is still not clear how developers signal their competitive advantage for selection and what is the impact of such selection. To answer these details, a quantitatively analysis is used.

1.2.7 Quantitative Study

Chapter 8 adopts a panel statistical analysis to explore the details of how a developer constitute the signal to *Google* for platform selection (about how the developer manages to go from being a developer of an app to being seen by *Google* as being worthy for platform selection) and how

market performance responds to developers' following behaviour. The chapter focuses on three strategic behaviours by developers including uniqueness (how an app is unique from other apps), diversification (how a developer diversifies on the type of apps) and following (how a developer follows to learn from other developers). The statistical results confirmed that *Google* welcomes the uniqueness or innovation, so apps that are unique will increase their chance for platform selection. However, by being too unique will cost too much in terms of resources and will increase the uncertainty of user acceptance or bugs in source code. Diversification behaviour is discouraged by *Google*. It is because that only excellent apps can stand out from a large supply of products. *Google* encourages developers to be specialists in one area until they have accumulated enough resources and power to expand into other areas in the app market. *Google* also encourages the learning culture in the *Google Play Store*. Existing research has claimed that producers prefer sub-domain with fewer other producers and more users. This thesis expands on this knowledge to argue that following selected developers will increase the chance of being selected. Developers who have similar or higher technical ability can understand the future potential of other developers' behaviour. This is a reason that good developers are likely to be clustered together even though they are competing against each other. Even more, the thesis find that selected developers will also follow non-selected developers to collaboratively explore the sub-domain in the app market. This following behaviour enhance the open culture to collaboratively expand the market size.

In summary, the statistical results suggest that developers should make costly actions to constitute a signal and this signal will be observed by *Google* to interpret for underlying innovation quality to make its selection decision. The costly actions include the unique use of app permission, less diversified apps to enhance higher levels of specialty, and proper following to learn how to innovate. Developers should follow other good developers to maximise their market performance. The finding confirms that *Google* can efficiently manage the innovation in such a highly competitive market via the selection of developers. With its selection mechanism, *Google* signals the innovation selection criteria to the developer community, and encourages them to learn from each other for competition. Without scanning all of its developers, *Google* can concentrate more on influencing good developers to innovate, and good developers will spread *Google's* desire to all other developers to push the *Google Play Store* into a healthy ecosystem.

1.2.8 Summary, Limitation and Contribution

The last chapter provides a discussion and summary to the thesis. The signalling selection model was built upon a signaling theory to solve the information asymmetry problem and aims for efficiency of the platform ecological development in the app economy. The thesis

makes an empirical study in the *Google Play Store* to answer the research questions including 1) **what** (constitutes a signal), 2) **who** (can observe the signal), and 3) **how** (to feedback for ecological development). This theory is designed to efficiently manage the two-sided market that is already massively expanded in duopoly. The contribution of this thesis is to expand the platform management knowledge into the duopoly situation which has not been thoroughly studied before. Even though the app economy is emerged and discussed in industry, it is not widely studied in the academic literature, and this thesis provides a foundation for future app economy study. More importantly, it also applied the signalling theory in the app economy for the first time and expands the knowledge of feedback in existing signalling theory. The thesis also has its own limitations. The major limitation is due to the dataset. The thesis only studies a very limited number of strategic behaviours in the *Google Play Store*. Its main competitor *iOS* adopts a closed strategy, which also should be studied to benchmark the foundation. The author believes that the app economy will become very important in the future economy, and this thesis is only a beginning.

Chapter 2

Research Questions

This chapter theoretically frames the research question on strategic signals. It firstly introduces the motivation to study the app economy as a supplement to existing two-sided market researches by updating to the current duopoly context. Unlike other literatures which focuses on how to increase the growth of the markets, I am interested in the negative impact of a massive growth in a two-sided market, as massive growth makes the competition hard and the innovation management difficult. The negative impact is particularly serious in the current duopoly context, and it is a priority to improve the quality of developers in addition to the quantity of apps.

To solve this, most existing studies of platform strategies are platform centric by assuming that developers respond the same to platform strategies. I identified that this assumption generates a limitation in practice, particularly missing the protection to newly but potentially innovative developers. Some recently literature has begun to look at developers' unequal behaviour, but their focus is on how developers switch between different markets to avoid the competition. This switching strategy is inefficient in the duopoly, as developers have no alternative options to switch to. So they have to directly compete with a massive number of peers, and innovation becomes important for them to win.

I observed developers' strategic behaviours to communicate their innovativeness. I found the innovative developers would make some unique but costly actions which are less likely to be attempted by less-innovative developers, as less-innovative developers have a higher opportunity cost than innovative developers to make these innovative actions. It inspires me to treat these costly actions as signals for innovative developers to strategically communicate their innovativeness. Based on the signalling theory and the examination of the featuring selection mechanism in the *Google Play Store*, I name a signalling selection strategy, and propose three research questions on 1) **what** (constitutes a signal), 2) **who** (can observe the signal), and 3) **how** (to feedback for ecological development).

2.1 Motivation

2.1.1 Two-Sided Market

A two-sided (or more generally, multi-sided) market (Rochet and Tirole, 2006) often involves two or more parties to trade in the market. Frequently studied two-sided markets include: payment card markets (Armstrong, 2006; Rysman, 2009), advertisement & media markets (Anderson and Gabszewicz, 2006; Kaiser and Wright, 2006; Von Ehrlich and Greiner, 2013), matching markets (Caillaud and Jullien, 2003; Damiano et al., 2008), B2B markets (Yoo and Choudhary, 2003), video game console markets (Rochet and Tirole, 2003, 2006; Srinivasan and Venkatraman, 2010), operating system markets (Eisenmann et al., 2006; Parker and Van Alstyne, 2000), computer hardware (Gawer and Cusumano, 2008; Gawer and Henderson, 2007), etc. This kind of markets has attracted a great amount of scholarly attentions with two perspectives (Rochet and Tirole, 2006; Rysman, 2009): platform dependency and network effect.

The first perspective is the platform dependency. More and more goods or services (for simplicity, referring to them as products) are technology or platform dependent (Brousseau and Penard, 2007; Gawer and Henderson, 2007; Ghazawneh and Henfridsson, 2012), for example, Intel or PowerPC platform based computers (Gawer and Cusumano, 2008; Gawer and Henderson, 2007), Microsoft IE or Netscape browser based websites (Yoffie and Cusumano, 1998), Visa or MasterCard associated credit cards (Rochet and Tirole, 2003). These products are stacked on certain complex and advanced technologies (Boudreau, 2012), and the platform provides a technology infrastructure to enable the reuse of these commonly demanded technologies (Ghazawneh and Henfridsson, 2012). By being dependent on a platform, producers then can dedicate their attention to developing their unique product features (Evans, 2009), and the platform owner can concentrate its efforts on developing "boundary resources" (Ghazawneh and Henfridsson, 2012) to support the activities of both the producers and consumers in the market. This encourages innovation in each production section (Evans et al., 2006; Gawer and Cusumano, 2013) and expands the products availability (Gawer and Henderson, 2007; Hagiu, 2009b). Platform dependency has influenced the business model to be transformed into open style (Boudreau, 2010; Economides and Katsamakas, 2006; Eisenmann et al., 2008) and scholars have paid particular attentions to open innovation (Chesbrough, 2003a), user innovation (von Hippel, 2005), open business model (Chesbrough, 2007b), user entrepreneur (Shah and Tripsas, 2007), etc.

The second perspective is the network effect. Existing researches tell us that firms with substantial network effects can be able to grow rapidly from a small customer base, because customers attract more customers (Rysman, 2009). A two-sided market must establish the

intermediaries to attract two or more parties to come to the market (Rysman, 2009). The market operators accumulate the revenue from the trade interaction between parties (Rochet and Tirole, 2003, 2006). With the network effect in a two-sided market, the size of one party influences the size of other party, and the revenue of one party depends on the aggregation of other side (Parker and Van Alstyne, 2005). So the market operators often subsidise one party to attract the other party in order to enhance the network effect (Lee and Wu, 2009). The positive result of network effect is to increase the economy scale (Evans, 2003) by enjoying the massive growth in market size and ecosystem (Evans and Schmalensee, 2007). The two-sided markets connect both producers and consumers to reach one another with ease (Pehrsson, 2004; Srinivasan and Venkatraman, 2010) and trade products in significant high cost efficiency (Gassmann, 2006). This motivates producers to share and collaborate in order to reduce the opportunity cost on product innovation (Farrell and Klemperer, 2007; Perrons, 2009). It has influenced the strategic thinking to develop from traditional value chain into value network (Normann and Ramírez, 1993; Peppard and Rylander, 2006) and the desire of critical mass (Evans and Schmalensee, 2010). Nowadays, many industry based companies are aiming towards the development of an ecosystem or already enjoying on the benefit of an ecosystem establishment (Gawer and Cusumano, 2013).

2.1.2 App Market

The app markets are recently emerged two-sided markets that typically have three parties involved. An app is a software application that runs on a specific platform technology and trade in the market (Ghose and Han, 2014). Taking *Android*¹ app markets as an example (see Figure A.1), the *Android* platform provides many reusable technologies for an *Android* app development and also an operating system for *Android* apps to run (Campbell-Kelly et al., 2014). In simple words, *Android* apps are the products to trade in the market and they are dependent on the *Android* platform (Basole and Karla, 2011).

The primary *Android* app market is the *Google Play Store*² which is owned and managed by *Google* as the market operator. The first party focuses on the development of the platform technology which is the *Android* platform. *Google* and other industry players are the major contributors to the *Android* technologies. The second party is dedicated to the development of *Android* apps which can sell on the *Google Play Store*. The third party purchases *Android* apps from the *Google Play Store* to install and run on their *Android* platform based devices. A direct competitor to *Android* is *iOS*. Nowadays, *Android* and *iOS* are the two mobile app platforms

¹<http://www.android.com/>

²<https://play.google.com/store/apps>

having the largest market share and strongest ecosystem (Ghose and Han, 2014). The rest of the mobile app platforms only takes less than 10% of total market share³.

2.1.3 App Economy

The economy behind the app market is named as the app economy in this thesis. I categorised the researches on app economy into three stages, based on the emergence and maturity of app markets (see Figure A.2). The first stage counts from the very early stage when people start to trade software until the emergence of mobile devices. In this stage, consumers purchased applications to install on a PC primarily. The computing devices were expensive and not commonly available. More importantly, software products were not centrally distributed at that stage and primary used traditional retail methods (Yoffie and Cusumano, 1998), so the app economy wasn't deeply studied in first stage with the focus being more on how to gain a platform competitive advantage (Gawer and Henderson, 2007). The research focus of platform strategy in the first stage was on the emergence (Evans and Schmalensee, 2010; Gawer and Henderson, 2007), leadership (Gawer and Cusumano, 2002, 2008) and dominance (Srinivasan and Venkatraman, 2010; Suarez, 2004) in platform growth.

The appearance of online apps retail channels has led to the second stage. Boudreau (2012) studied the early stage of mobile technologies between the late 1990s and early 2000s, which is dominated by PDAs. In this stage, mobile app platforms started to build their own platform-specific retail channels, these platforms included *Palm*, *Microsoft*, *Symbian*, *RIM*, *Linux*, *Motorola*, *Casio*, *Java*, etc. App consumers still require certain technical skills to purchase, install and run the apps in order to finish the trading transaction (Jansen and Bloemendal, 2013). This was the emergence of app markets but mobile devices were expensive at this stage, so the total market size of app economy was small compared to now. These retails channels were independent from each other and had a similar market size (Basole and Karla, 2011). Therefore, scholars were motivated to study app producers' switching behaviour (Farrell and Klemperer, 2007) among different retail channels (Boudreau, 2012) with price strategy (Hagiu, 2006, 2009b; Lin et al., 2011) and exclusive contracting (Armstrong and Wright, 2007; Hagiu, 2009a; Hagiu and Lee, 2011; Mantena et al., 2010). The aim was to find out how to compete to attract app producers to come into a platform-specific app market (Cennamo and Santalo, 2013; Chen and Miller, 2012; Eisenmann et al., 2011).

The widespread app markets is the third stage which is the case of the current app economy. The price of smartphone devices (Basole and Karla, 2011) and its app (Liu et al., 2014) have been significantly dropped. The technical barrier of app trading and installation has been

³<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

removed (Ghazawneh and Henfridsson, 2012; Ghose and Han, 2014). The market size has never been so big (Ghose and Han, 2014). More importantly, two giant ecosystem platforms have been established to form a duopoly (Basole and Karla, 2012). IDC⁴ reported in 2014 that *Android* and *iOS* have occupied 81.2% and 15.2% of total smartphone market share respectively, which is a signal of duopoly. Besides the smartphone, both platforms have been developed into many other different business domains, like TV, automotive, wearable, home, etc. *Android* and *iOS* have become two platforms that not only mobile app producers but also other types of app producers want to target, and they have no choice but to switch due to the duopoly of the establishment. The great success of the platform approach has stimulated the appearance of other types of app market around the ecosystems of *Android* and *iOS*, for example, app markets for TV, automotive, web browser, desktop software, camera, etc. Therefore, new management strategy researches for such a competitive duopoly has become necessary in this stage. According to my knowledge, research into this stage of the app economy is still rare.

2.2 Problem

2.2.1 Research Interests

How to grow and maintain the sustainable ecosystem is an important question to platform strategy (Gawer and Cusumano, 2013), and is also a key interest to all parties in the app economy (Genakos and Valletti, 2012). An app platform has limited resources to develop all kinds of apps to meet all kinds of user demands (Mantena et al., 2010). Rather than creating all products by itself, the platform chose to establish its competitive advantage by enhancing its platform technology (Mantena et al., 2010) to attract external producers and by providing efficient supports to producers to encourage them to develop apps demanded by consumers (Ghazawneh and Henfridsson, 2012). By being dependent on a platform, app producers can improve production by reusing the platform's infrastructure, but they need to devote certain resources to learn and then develop the app products to sell in the market. This inter-dependency enhances the lock-in effect (Farrell and Klemperer, 2007) and increases the opportunity cost for app producers to switch to other platforms (Armstrong and Wright, 2007). It also motivates both platform and producers to share the same goal to expand the products range to attract more consumers (Hagiu, 2009b; Sawhney, 1998) with consideration of the value network (Normann and Ramírez, 1993). By sharing the same goal to expand the market size, the platform and the producers collaborate (Lavie and Rosenkopf, 2006) and share (Katila and Chen, 2008) to

⁴<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

exploit and explore innovation (March, 1991) and stack up the technologies (Jiang et al., 2004; von Krogh and von Hippel, 2006). All parties enjoy the benefits of an established ecosystem.

Management becomes radically difficult, especially when app markets developed into a duopoly. The majority of past researches explores the knowledge of platform growth (Gawer and Cusumano, 2008) and platform competition (Cennamo and Santalo, 2013) in order to generate a large market size for critical mass (Evans and Schmalensee, 2010) in an ecosystem (Gawer and Cusumano, 2013). As far as I am aware, very little attention was given to the negative effects of market growth in the platform ecosystem. I knew that the increase on market size would attract more producers enter into the market due to the network effect (Cusumano, 2010), but would increase the competition among app producers (Boudreau, 2012) and may narrow the leading time for app innovation (Katila and Chen, 2008). The competition in current duopoly is extremely high⁵, since all developers want to compete for a market share in the *Google Play Store* and *iOS*. App developers are enjoying the great benefits from the other side of massive app consumers in an app market, but also undertake the pressure to compete with a massive number of peers on the same side. Very competitive and crowded market environment can narrow producers' margin profit margin (Zhu and Iansiti, 2012). This would motivate disorderly behaviour in the markets, for example, copycat (Yadong Luo et al., 2011), imitation (Enkel and Gassmann, 2010), free-riding (Evans, 2003; Han et al., 2012b), malicious (Zhou et al., 2012b) and legitimate products (Zhou et al., 2012a). Platform growth is fundamentally driven by continuous innovation that supplies products to the platform market (Lin et al., 2011). These behaviour can damage innovators' profits and distract the motivation for innovative and original active players, especially for people who are want to innovate (Rasch and Wenzel, 2013). Furthermore, it would have a negative impact on the market's reputation, user experience and market size.

The second negative impact of high competition is to increase the difficulties and costs for new entrants to grow. Innovative new entrants play an important role as boundary spanning (Gruber et al., 2008) or weak tie (Hoffmann, 2007) or directly introduce fresh ideas or products into the markets (Eisenmann et al., 2008; Gawer and Henderson, 2007; Zhu and Iansiti, 2012). Boudreau (2012) confirmed that not only established producers are discouraged to stay, new entrants may also be easily crowded out in competitive app markets. New entrants have to innovate on niche markets to avoid the direct competition from other established giants (Anderson et al., 2010; Parrish et al., 2006; Santos and Eisenhardt, 2009), or distract their resources from innovation to deal with non-innovative competitors (MacCormack and Verganti, 2003). In another aspect, open strategy has been broadly adopted by current popular platforms (Economides and Katsamakas, 2006; Schiff, 2003). Many entrants or existing innovators are

⁵http://www.stardust-testing.com/ressources_1/notice-m-reputation-en.pdf

"accidental" entrepreneurs (Shah and Tripsas, 2007) in these open platforms (Boudreau, 2010). They generally have less resources to compete with experienced competitors. However, their innovative activities should be protected or encouraged, as their innovation normally comes from the direct experience or observation of their own community's demand (Dahlander and Magnusson, 2008; Von Krogh et al., 2003; West and Lakhani, 2008). Meanwhile, established players in two-sided markets would be more likely to have the "winner-takes-all" advantage to attract a skewed distributed market share and user attention (Cusumano, 2010; Sun and Tse, 2009), which makes new entrants more difficult to survive.

I am interested in exploring the platform strategy to balance the protection and encouragement to both well established producers and new but potentially good entrants, in order to develop the ecology of the platform ecosystem. This question is particularly important to explore in the third stage of the app economy, as the market is already a duopoly. In the first two stages of app economy, a higher priority in platform strategy is to focus on a platform's emergence (Gawer and Cusumano, 2002) and leadership (Gawer and Henderson, 2007). So a key mission for an app market in the first two stages is to attract enough number of app producers (Rysman, 2009) which is the quantity of producers (Hagiu, 2009a). After two stages of development of the app economy, an app market of duopoly has already accumulated a large number of app producers and consumers. In the third stage, app producers compete aggressively for a chance to serve a sizeable market and innovate intensively to sustain or recapture a market share (Andrew et al., 2013). Their innovation intensity then determines the rate of market turnover (Lin et al., 2011). So the key mission for an app market should shift to attract innovative app producers and encourage them to keep innovative in order to establish and maintain the critical mass in the ecosystem (Evans and Schmalensee, 2010), which is to enhance the quality of app producers in addition to quantity. The competition environment is also changed from the app producers' point of view, which pushes app producers to adapt their strategies. For example, before the duopoly, app producers can switch between markets (Boudreau, 2012) to enjoy the first or early move advantage (Srinivasan and Venkatraman, 2010) or fast exploration and exploitation to capture the innovation opportunity (Katila and Chen, 2008). App producers can also survive with the subsidizing from platforms who are aiming to use price strategy to grow one side of the market (Lee and Wu, 2009; Lin et al., 2011). In a duopoly, these welfare seems to be very narrow, as app producers aim to crowd and directly compete in each app market of duopoly, and producers have no alternative markets to switch to. Even more, app markets use platform envelopment strategy (Eisenmann et al., 2011) to learn and observe new features from other successful platforms and even innovate into other business domains (Gawer and Cusumano, 2013), which generate a perception of a long life of duopoly in the app economy. Therefore, it is necessary to study app producers' strategic

behaviour and an efficient platform strategy to protect and encourage the innovative activities in the duopoly of the app economy, which forms the research interests in this thesis.

2.2.2 Existing Platform Strategies

Scholars have attempted several platform strategies to improve the ecology of two-sided markets. The first strategy is to increase the barriers for entry into the market (see Figure A.3a). Platforms always set up some barriers for producers to enter into the market (Eisenmann et al., 2011; Gawer and Henderson, 2007; Pehrsson, 2009). For example, producers have to learn the platform technologies in order to create products based on the platform (Evans et al., 2006), or producers have to pay entry fees in order to list products to sell on the market (Rysman, 2009), or producers have to share profits with platform operators for selling their products (Lin et al., 2011), etc. Previous scholars have particularly studied the barrier strategies via pricing (Hagiu, 2009b; Lin et al., 2011), customer accessing (Pehrsson, 2009), exclusive contracting (Hagiu and Lee, 2011; Mantena et al., 2010), tying (Amelio and Jullien, 2012; Choi, 2010; Lee and Wu, 2009), boundary resource (Ghazawneh and Henfridsson, 2012), regulation (Bauer, 2014; Genakos and Valletti, 2012; Perrons, 2009), switching costs (Farrell and Klemperer, 2007), etc. Higher entry barriers would increase the opportunity costs for producers to enter and also deduct the profits of each single transaction for existing players (Pehrsson, 2009). Good producers with having higher abilities are more likely to survive in the market with high barriers, as their opportunity costs are relatively lower. This strategy aims to filter out good producers via barriers and lets less competitive producers exit the markets.

The limitation of this strategy is that all producers have an increased opportunity cost by facing the same barriers. This especially prevents the chance for new entrants or young players in the market to grow (see Figure A.3a). In the long run, this may keep the existing good producers, but would also deduct the total number of producers which may damage the future market size. Historically, we know that Intel-Microsoft adopted a low entry barrier strategy to win the market size, and Mac has adopted high entry barrier strategy to keep the high quality of software (Gawer and Cusumano, 2002). The market size is the foundation of network effects in two-sided market, and the barriers impact on the market size. Meanwhile, the two-sided market scholars found that market operator should subsidize the more price sensitive side, and charge the side whose demand increases stronger in response to growth on the other side (Lee and Wu, 2009). The platform can shape the market structure in a way that encourages investment (Mantena et al., 2010). For example, a platform can control market structure through fees, barriers to access, or the direct exclusion of some firms and thus affect incentives for innovation. Therefore, either increasing or decreasing barriers to producers should be carefully evaluated in a two-sided market.

The second strategy is to develop an open culture in the market (Boudreau, 2010; Economides and Katsamakas, 2006; Ghazawneh and Henfridsson, 2012). Close culture means each party pretends to keep its intelligence and resources in private, opposite to the open culture which encourages sharing (Schiff, 2003). Close culture would keep more control in the process and quality (Boudreau, 2010), for example *Apple* keeps *iOS* private and manufactures all hardware devices by itself (Basole and Karla, 2011). An open culture would attract more external contributions (Boudreau, 2010), as each contributor would gain benefit from the sharing and collaboration. *Android* has adopted an ethos of open strategy since its inception. Unlike *iOS*, it establishes an *Open Handset Alliance*⁶ with initially 34 industry members to drive the development of *Android* (Remneland-Wikhamn et al., 2011). It also makes *Android* technologies completely free via open source (Basole and Karla, 2011), so that anyone can modify the source code and manufacture the hardware devices. This strategy significantly attracts the collaboration for innovation and develops the *Android* market size.

The disadvantage of open strategy is the loss of control of the ecosystem. *Android Fragmentation* is a typical issue (I will discuss these details in Section 3.3.3) and enhances the difficulty on management (Ham and Park, 2011; Han et al., 2012a), which in turn damage the development of a healthy ecosystem. Besides, open sourced *Android* enables people to establish third-party *Android* app markets (Zhou et al., 2012b) which would dilute the central power of the official *Android* app market which is the *Google Play Store*. Producers confirmed⁷ that they feel more ownership in *Android* contributions and that they enjoy the benefits of a large market share, but feel less pain on *iOS* app development.

Barrier (especially pricing) and openness are the two most important platform strategies that scholars emphasised in the past. In the current duopoly context, *Android* and *iOS* have adopted a lot of similar barriers to their markets. For example, both platforms have entrance fees and all share 30% of the selling revenue with app producers (Basole and Karla, 2011). Even though *iOS* is less open on its platform source code in comparison to *Android*, *Apple* shares a lot of free tools and other resources with its developers (Remneland-Wikhamn et al., 2011). Open strategies have been widely encouraged by scholars (Boudreau, 2010; Chesbrough, 2007b; Economides and Katsamakas, 2006; Eisenmann et al., 2008; von Hippel, 2005; West, 2003), and is an non-stopping trend in the computing business. These existing strategies attempt to govern all producers with the same standard and to adjust the opportunity costs and other costs to all producers in the same criteria, for example, setting up the same barriers to all producers or open source code to all producers in an app market. I believe the developers should be treated differentially (see Figure A.3b), and a supplement non-price strategy based on existing

⁶<http://www.openhandsetalliance.com/>

⁷<http://blog.gqueues.com/2013/07/android-vs-ios-comparing-development.html>

strategies is worth investigating to fit into the current duopoly context (Evans and Schmalensee, 2010).

2.2.3 Strategic Behaviour

Last section has summarised the limitation of current platform strategies which is to apply same barriers or benefits to all producers. I already discussed that in the current duopoly, platforms should shift more attentions towards innovative producers, including well established players or new but potentially good entrants. Therefore, a key problem is how to understand the app producer's ability to innovate. The study is not only demanded by the platform (or the market operator), but also by app producers and consumers, as high ability producers are the targets for other producers to learn from, and also the target for app consumers to buy apps from. From the consumers' aspect, their search cost would be significantly increased to find a desirable product from such a large variety of choices (Boudreau, 2012; Lee and Raghu, 2011). They have demands on an efficient mechanism to help them quickly filter out the products based on quality and find the best app that they are desired (Hagiu, 2009b).

The crowded environment would make the identification of the quality and level of app producers' abilities very difficult. It is impossible for market operators to manually carry out detailed reviews of all the source code of each app in the app market. Taking *Android* as an example, a 2012 survey⁸ reported that in a single day it had 2,371 new apps published and 2,341 *Android* apps updated in the *Google Play Store*. Public information disclosed that *Google Play Store* should have 675,000 apps⁹ in September 2012, and 1,430,000¹⁰ apps in December 2014. These figures demonstrate that a huge amount of human effort is required to manually check the quality of apps, which is not practical in the reality. Crowded wisdom is also not efficient as most consumers lack technical capabilities and motivations to identify quality of apps (Lin et al., 2012). More importantly, information asymmetry (Wittenberg-Moerman, 2008) significantly increases the difficulty of quality identification (Stiglitz, 2002). Only app producers themselves know how good they are, and how much value their apps would be worth of (Akerlof, 1970). However, a app producer will try his best to claim that his apps have the best quality (Spence, 1973). Some of them even "cheat", for example, malicious app can steal data but it is difficult to observe by people without technical skills (Chia et al., 2012). This inequality of information between producers and others causes the difficulties for ensuring for quality identification. Therefore, it is believed that the negative effects of information asymmetry should be counted into the solutions proposed in this thesis.

⁸http://www.stardust-testing.com/ressources_1/notice-m-reputation-en.pdf

⁹<http://officialandroid.blogspot.co.uk/2012/09/google-play-hits-25-billion-downloads.html>

¹⁰<http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/>

Earlier researchers have explored some strategic behaviour by producers in innovation. An often mentioned behaviour is to differentiate future coming products by making them different from existing available products. Producers can compete in niche (Parrish et al., 2006) or nascent (Santos and Eisenhardt, 2009) markets for new types of products (Parrish et al., 2006; Santos and Eisenhardt, 2009) or adapt projects into specific contexts (like differences between countries (Pehrsson, 2004)). Diversification is the second often mentioned behaviour, which is to make significant differences to upcoming products (Ulaga and Eggert, 2006), and some of which are even "disruptive" (Christensen et al., 2006). Diversification comes with more uncertainty (Goranova et al., 2007), but the successful deployment of diversification would open great opportunities for business growth (Wiersema and Bowen, 2008). Exploration and exploitation are the common approaches for diversification (Lavie and Rosenkopf, 2006). Learning helps producers to grow and maintain their current leading status. Some common types of learning include: copycat (Yadong Luo et al., 2011), imitation (Enkel and Gassmann, 2010), fast second (Markides and Geroski, 2004), and search (Katila and Chen, 2008). The purpose of these behaviours is to establish the competitive advantage (Barney, 1995; Porter, 2008; Sheffi, 2005).

Some strategic behaviours have received particular attentions in the open source software and open strategy communities. Reuse (Frakes and Kang, 2005; Mojica et al., 2014) is an important one, which is very common in the software industry (von Krogh and von Hippel, 2006). Reuse saves the cost of duplication and also accelerates the speed of innovation (Lerner and Tirole, 2002). Software developers may reuse some open source, and this has pushed the research on open source behaviour (Baldwin and Clark, 2006; Dahlander and Magnusson, 2008; Hertel et al., 2003; Lakhani and Von Hippel, 2003; Raymond, 2001; Roberts et al., 2006; von Hippel and von Krogh, 2003; Von Krogh et al., 2003). Some open sources projects have achieved great commercial successes (Bessen, 2006; Fitzgerald, 2006; Hauge et al., 2010; Roberts et al., 2006; Rossi, 2006), and some developers have created entrepreneurial ventures from their contributions to open source projects. This has encouraged the researches of strategic behaviour in open business models (Chesbrough, 2007b; Zott et al., 2011), open innovation (Aitamurto and Lewis, 2013; Chesbrough and Crowther, 2006; Chesbrough, 2003a,b; Enkel et al., 2009; Harison and Koski, 2010; Huizingh, 2011; van de Vrande et al., 2009), user innovation (Baldwin et al., 2006) and user entrepreneurs (Boudreau and Lakhani, 2013; Shah and Tripsas, 2007; Shane and Venkataraman, 2000a).

Certain strategic behaviours have become particular interest within the app economy. The first one is the emphasis on some unique resources (Lim and Bentley, 2012b), like user data. Mobile apps generate and capture a great amount of user data, and this data becomes an important asset in the digital economy (Kalapesi, 2013; Schwab et al., 2011). Users can

stay with the businesses that have the richest form of user data at their disposal (Koch and Kerschbaum, 2014). Many new business models, like *freemium* (Liu et al., 2014), have become popular in the app economy. This also pushes the development of the API strategy (McDonnell et al., 2013), which performs as an efficient tools for internal or external communication and collaboration between apps and their developers (Aitamurto and Lewis, 2013; Chin et al., 2011; Li et al., 2014), and attracts other externalities (Boudreau and Lakhani, 2009), like mashup (Yu et al., 2008). In order to attract user attention in the app competition, app producers can frequently update (Möller et al., 2012) their apps to introduce new features or just to compete for the chance to display as a high priority product. To save the cost on R&D, some producers repack (Zhou et al., 2012a) existing apps by giving them new titles or a new look. Some of them confuse the consumers and also damage other producers' benefits, which is considered piracy (Rasch and Wenzel, 2013).

I have collected and summarised all strategic behaviour to the my best of my knowledge from the literature (see Table A.10). There are other strategic behaviours which are not entirely relevant to this thesis which have not been mentioned, for example marketing and finance, as a result they have not been referred to in my research. The purpose is to identify these strategic behaviours in order to assist with the examination of patterns on producers' abilities under information asymmetry, so that platform owners can select to work with producers who are really innovative. Based on above argument, I name a signalling selection strategy to be introduced in next section.

2.3 Solution

2.3.1 Signalling Selection Strategy

Information asymmetry has caused the difficulty on ability or quality identification (Connelly et al., 2011). Only producers know their real ability and the quality of their products (Spence, 1973). Each member in the market discloses different amounts of information to the market (Akerlof, 1970), and parties who receive the same amount of information may also have different interpretations (Spence, 2002). This makes the information become unequal or asymmetric between parties (Stiglitz, 2002). High ability producers want to disclose enough information to tell the world about their good abilities (Kirmani and Rao, 2000), and less able producers also want to disclose enough information to confuse the market with "fake" quality so that they can sell their products at the same or a similar price in comparison to high ability producers (Akerlof, 1970). This brings the challenge of market selection. Therefore, the key challenge is to filter out 'fake' information and observe useful information (Spence,

2002). An efficient solution to information asymmetry is the signalling theory (Connelly et al., 2011) which has been applied into many market domains, for example the job market (Spence, 1973), "lemon" market (Akerlof, 1970), education market (Bedard, 2001; Lang and Kropp, 1986), crowdfunding market (Ahlers et al., 2015), IPO market (Certo et al., 2001; Cohen and Dean, 2005; Deb, 2013; Filatotchev and Bishop, 2002), etc. This thesis attempts to apply the signalling theory into app markets to overcome the information asymmetry in identifying a producer's ability and to develop a strategy to efficiently manage the ecological development in the app economy.

The signaling theory explains a behaviour that a signaller sends costly signals to a receiver and the receiver then feeds back his interpretation back to the signaller (Connelly et al., 2011). A signaller firstly makes certain costly actions to generate signals. These actions require less opportunity costs for producers having real abilities to engage, so "faked" producers would be less likely to act to generate the same signals by paying higher opportunity costs with less amount of return (Stiglitz, 2002). These signals are observed by a receiver who is interested in understanding a signaller's true quality (Kirmani and Rao, 2000). Quality is invisible (Kirmani and Rao, 2000), but only high quality producers make these costly actions, so receivers interpret that the signaller should have the relevant real quality and will then feedback his interpretation back to the signaller (Connelly et al., 2011). This frames the basic components of a strategy solution for platform ecosystem management proposed in this thesis: parties who are interested in ecological development observe producers' behaviour in the market, and then select the producers who make the desired behaviour to award benefits and these benefits are much more than the cost that the maker paid and these actions should be encouraged. This strategy is named as **signalling selection strategy** in this thesis (see Figure A.4).

Signalling and feedback are the two most important components in order to complete a signalling selection strategy. Signalling is to observe the costly actions that specifically relate to the ability to innovate. No matter whether they are established players or new entrants, innovative producers have the similar patterns of strategic behaviour towards innovation. Non-innovative activities can be filtered out in the selection criteria. This differentiation of the producers' ability solves the limitation of prior strategies that consider all producers to be the same.

Feedback generates the motivation for signalling (Connelly et al., 2011). Feedback discloses selection results to the public to reduce the effects of information asymmetry, and the pay back benefits the makers' actions. Signalling is costly and, if the benefits from feedback are greater than the cost of generating signals, producers will be encouraged to produce signals on innovation (Akerlof, 1970). This differentiates the benefits to be received by producers having

different actions and abilities but ignores the producers' current market status and entry time, which overcomes the limitation of prior platform centric strategies.

Previous strategies have emphasised the need to subsidise producers (Choi, 2010; Hagi, 2006, 2009b; Lee and Wu, 2009; Lin et al., 2011), which is a kind of feedback equivalent to the signalling selection strategy. It is agreed that direct subsidy is useful to attract producers by giving them direct financial support to increase the growth the platform (Lee and Wu, 2009). This kind of subsidy method is criteria based, and some producers can temporarily "attempt" to meet the criteria to gain subsidy (Boudreau, 2012; Hyrynsalmi et al., 2012a), which partially leads to a multi-homing issue (Choi, 2010). Signalling selection strategy recommends selection based subsidies, so that producers have to be selected in order to enjoy the benefits from feedback. It also means that the platform holds the authority to observe producers' behaviour in the long term to make or modify their selection decision. Another issue of prior subsidy strategies is directly linked to price incentive (Amelio and Jullien, 2012). However, consumers are the final targets of producers and only satisfied consumers can provide sustainable financial supports to producers. Therefore, the signalling selection strategy emphasises more on the indirect subsidy by opening the opportunity for consumer access (Belleflamme and Peitz, 2010; Pehrsson, 2009) and increasing the chances for customer selection in the markets (Lee and Wu, 2009).

Signalling selection strategy is designed to build the network effect on selection. Selected producers will receive great benefits of feedback after signalling. Selected producers watch other selected producers' innovative behaviour to keep their selected status (Katila and Chen, 2008). Other non-selected producers will be motivated to learn from these selected producers in order to increase their chances of selection (Calantone et al., 2002). This enables the platform to use a small number of selected producers to influence the entire producers side, similar to the function of leader users in open and user innovation (von Hippel, 1986). Rather than the governance of a massive number of producers, the platform can focus their dedication onto managing a small number of selected producers, which saves a lot in terms of costs. App consumers also dedicate their attention to selected producers to save their search costs, and have a reference on the app quality that should be recognised in the platform. The network effect is that the platform is developed into a desired ecosystem with critical mass (Evans and Schmalensee, 2010).

In summary, even though information can be asymmetric, the signalling theory believes that people having different abilities would behave differently (Connelly et al., 2011). The purpose is to identify the producer's real ability in such a competitive market environment. By applying signal theory to the app economy, producers who are or have potential to produce high quality apps behave differently from producers having less ability in order to generate

observable signals. If the market can properly interpret these signals and feedback to signaller by giving them protection and encouragement, producers who are or have the potential to produce high quality apps are more interested to engage in costly actions, which improves the general incentive of innovation in the ecosystem.

2.3.2 Research Questions

This forms three research questions on the signalling selection strategy to study in this thesis:

- **1) what costly actions can generate efficient signals in the app economy?**
- **2) who can interpret those signals in the app economy?**
- **3) what kinds of feedback would encourage signallers to continue to send signals in the app economy?**

To find out what costly actions can be signals in the app economy, two groups of producers with different levels of ability are required to benchmark for their behaviour patterns. By comparing and contrasting the different behaviours between high and low ability producers, it will inform what costly actions are more likely to be made by high ability producers. To find out who can interpret the signals, the study needs to examine who has the interests to observe the producer's quality (who wants to observe), who has the capability to understand the meaning behind signals (who can observe), and who has the power to feedback benefits to the signaller's behaviour (signaller's motivation). To find out what feedback should be given, the study needs to examine what benefits signallers can obtain to cover the extra opportunity costs of making costly actions. It also means that the signaller can be selected from other massive producers for awards for benefits.

2.3.3 Research Method

The *Google Play Store* provides a foundation to study the signalling selection strategy in the app economy. There are two distinguished groups of developers in the *Google Play Store*; one group is the *featured* developers who received a special badge ("Top Developer"), the other group is the rest developers who are not featured in the platform. *Google*, which is the market operator in the *Google Play Store*, wants to identify the developer's quality. So *Google* periodically reviews developers and their apps, and then makes a decision to award this badge to the developers that *Google* wishes to encourage. This badge brings users more confidence as users can treat it as a guarantee after reading the *Google* review. More importantly, *Google* gives a higher priority to display *featured* developers and their apps to users. This significantly increases the exposure opportunity and selling performance by having a significantly larger customer access.

The rest developers who have not received the special badge is belonging to the second group called the *non-featured* developers. *Featuring* is a feedback from *Google* to developers who sending desired signals, and this feedback generates great benefits to developers to pay for the opportunity costs when they make desired behaviour. The great benefits of *featuring* motivate other *non-featured* developers to accumulate the ability to lower down their opportunity costs in order to generate these signals. More importantly, *non-featured* developers learn from *featured* developers to increase their chance of selection, as *Google* never discloses the details of its selection criteria to the public. The learning brings two benefits to the signalling selection strategy. The first benefit is that *Google* can influence the whole market by only working with the selection of *featured* developers, which eases *Google*'s workload in comparison to manually checking. The second benefit is to encourage the sharing and collaboration between developers to continually expand the market size. Learning encourages developers to identify other developers' abilities, and these evolving developers are closer to other developers having same or higher abilities via searching. This searching process will speed up the innovation inspiration which generates a variety of products and improve the average quality (Katila and Chen, 2008). This thesis makes an empirical study on the *Google Play Store* to answer the research questions.

2.3.4 Solution Summary

In summary, this thesis names a signalling selection strategy with the aim of efficiently managing the ecological development in the app economy, especially when a duopoly of app markets forms. When an app market grows into a duopoly, all developers aim to complete in this app market. This makes the app market to be very crowded. This crowding effect makes new developers difficult to grow and discourages established developers to innovate. So the market operator has the motivation to select good or potentially good developers to give them protection and encouragement. The crowded nature of the app market and information asymmetry makes the traditional selection process difficult. In order to efficiently manage the ecological development, a signalling selection strategy was developed by applying the signalling theory into the app economy. In the strategy, the market operator observes developers' behaviour and then selects desired developers to feedback with great benefits. The benefits are much more than the opportunity costs for selected developers to make desired costly behaviour in order to generate signals. However, these costly behaviours can only be done by developers who have a very high ability as their opportunity costs would be much less than other developers. The great benefits of signalling encourages other developers to learn from the selected developers. Learning can ease the market operator's management work by focusing on influencing the small number of selected developers, and the selected developers will spread the market operator's

desire into the whole developer community with peer effects in learning. This strategy will then encourage the sharing and collaboration between developers to develop the market into a healthier ecosystem.

Chapter 3

Background

This chapter aims to provide a background discussion into the app economy. The app economy has begun to appear in industry technology and academic technology research, but not often in academic business research. According to existing materials, this chapter summarises the app economy's characteristics that are pertinent to the research questions. It then concentrates on the introduction of the *Google Play Store* which is the primary app market to analyse in this thesis. The chapter introduces a complete history of the *Google Play Store*, how the *Google Play Store* adopted open strategy to grow and compete with *iOS* and also the consequence of this open strategy which is *Android Fragmentation*. *Android Fragmentation* has affected the innovation speed and ecological development of the *Android* platform. Large scale and high competition enhances the information asymmetry and brings the problems of innovation management in the app market. To solve this problem, an efficient selection mechanism is required.

3.1 App Economy Researches

The app economy has been mentioned in the news many times, for example *Fortune* ¹, *Bloomberg* ², *The Guardian* ³, *The Wall Street Journal* ⁴, *Financial Times* ⁵, and etc. These news have introduced the amazing development speed, size of market and broadly participation in the app economy. This chapter briefly reviews the concept of the app economy, and then specifically dives into one of the largest mobile app markets called the *Google Play Store*. The

¹<http://fortune.com/2015/01/22/apples-apps-economy-bigger-than-hollywood/>

²http://www.bloomberg.com/bw/magazine/content/09_44/b4153044881892.htm

³<http://www.theguardian.com/technology/2009/jul/29/iphone-apps>

⁴<http://www.wsj.com/articles/SB10001424127887323511804578300183961009070>

⁵<http://www.ft.com/cms/s/0/d3aacf38-1cb8-11e4-88c3-00144feabdc0.html>

concept of the app market has emerged over many years, but has never been competitive as it is as this moment (Ghose and Han, 2014). Innovation is the key to survive in competition (Lin et al., 2011), and drives a market forward in a healthy manner. The purpose of this thesis is to understand how developers can signal their ability for tech competence in the app economy. This chapter introduces the research background.

An app, an abbreviation for “software application”, generally refers to a mobile app that specifically runs on a smartphone, tablet computers and other mobile devices (Basole and Karla, 2011). The app economy has appeared since mobile technology was introduced (Basole and Karla, 2011), in the age of *Nokia*, *Blackberry*, *Palm* and *Windows Mobile Phone* (estimated to be between the years 2000 to 2007). During this period, apps have a different distribution method and developers’ participation was different from today (Jansen and Bloemendal, 2013). The mobile phone was expensive, using apps (such as installing or uninstalling) required a lot of technological ability for users. Hence, the market size was not significantly big enough to attract broader attention at that time. However, since the popularisation of *iPhone*, *Android*, *Windows Phone* and other recent brands (from 2008 to the present day), the app economy has become a hot topic and is already embedded into people’s daily life (Campbell-Kelly et al., 2014). It has compelled both enterprises and individuals to adopt the mobile technology. For this reason, when people discuss the app economy, it generally refers to the later stage. *Android* and *iOS* together occupy most of the market share (comScore⁶ reported that *Android* and *iOS* took over 93% of the total market share in 2014). Companies attempt to have a mobile app on *Android* or *iOS*, similar to having a company website prior to the smartphone era. *iOS* is a closed platform; hence a limitation applies to collect enough data for a research study. Therefore, in this thesis, the app economy is targeted on the study of the *Android* platforms for app development.

Even though the app economy has been mentioned, there is no common accepted definition of what the app economy is (Basole and Karla, 2011). In the definition of this thesis, an app is the product to trade and run on a particular mobile technology platform in the app economy. For example, an *Android* app is running on an *Android* device and the *iOS* app is running on an *iPhone*. The app producers (or developers) can create apps in any physical locations, but virtually sell their apps in an online virtual central market place. For example, *Android* apps are traded in the *Google Play Store*⁷, and *iOS* apps are traded in the *Apple App Store*⁸. App consumers (or users), purchase apps from the online market place and install them on their device. The financial capital transfers from users to platform and then distributes from platform

⁶<http://www.comscore.com/Insights/Market-Rankings/comScore-Reports-January-2015-US-Smartphone-Subscriber-Market-Share>

⁷<https://play.google.com/store/apps>

⁸<https://itunes.apple.com/gb/genre/ios/id36?mt=8>

to developers. Many apps currently use the freemium business model (Anderson, 2009), so that revenue is generated from a third party after purchased rather than directly from users. It is a market based economy, as the price for apps are freely set by consent between developers and users (West and Mace, 2010). The market operator manages the discipline to govern the fair price which is driven by quality. Once an app starts to sell on a platform, it is always available unless removed by the market operator or the app developer. The virtual exchange mechanism removes the geographical limitation, and reduces the trading transaction costs (Clemons, 2009). Meanwhile, both producers and consumers are heavily locked into a specific mobile technology platform, as developers have to learn the technology in order to develop a product based on the platform and consumers have to purchase the platform device in order to use the apps. The switch cost is also low. For example, consumers can switch mobile platforms easily or have both *Android* and *iOS* devices (Hyrnsalmi et al., 2012b), and it is common to see that producers develop apps for multiple mobile technology platforms (Hyrnsalmi et al., 2012a). Even though each transaction between consumers and developers is cheap, the large scale of apps and users participation leads to the significant increase in the economies of scale in the app economy (Campbell-Kelly et al., 2014).

The above description is not aimed at providing a comprehensive definition of the app economy as the app markets are still emerging and developing. However, this is an attempt to tell the difference from the other economies (Campbell-Kelly et al., 2014). Compared to traditional industry economy, the app economy is less restricted by economic processes (MacCormack and Verganti, 2003), like production, trading and consumption. Unlike traditional economies that depended on depleting natural resources, the app economy relies on other kinds of resources including talented human resources (Schultz et al., 2011), advanced computing technology (Charland et al., 2011) and a large user community (Fu et al., 2013). Produced apps are digitalised and the market operator maintains the app storage and distribution (Jansen and Bloemendal, 2013). This does not mean that the app creation is a one-off process; developers have to update apps to embrace the advances of hardware and mobile operating systems (Möller et al., 2012; Xing et al., 2014). The app economy is developed from the traditional economy on software and desktop Internet (Clemons, 2009). There are some common features but also there are difference in details. Compared to the traditional software economy, the app economy has a much larger number of participants involved in the economic activities (Koch and Kerschbaum, 2014), and developers have an easy market entry and require less commercial ability (Yamakami, 2011). Internet is designed to be stateless (Clemons, 2009). In comparison to the desktop Internet economy, the app market is relatively concentrated as the app economy always has a central market place (Ghose and Han, 2014). The app economy significantly depends on the number and networked nature of consumers (or user base) (Boudreau, 2012). A

large user base will stimulate the number of producers. Business can still survive with a large user base, even though consumers acquire free products (Clemons, 2009). A large user base is also the foundation to many app features (Liu et al., 2014), like social networking or collective intelligence. As a result, a top priority for all members in the app economy is to establish a large number of installed user bases.

Some independent industry research organisations have started publishing reports on the app economy. These organisation includes *Statista*⁹, *Portior Research*¹⁰, *comScore*¹¹, *Flurry*¹², *VisionMobile*¹³, *Forrester*¹⁴, *Gartner*¹⁵, *GIGAOM*¹⁶ and etc. These reports explore various paths towards understanding the app economy by surveying or tracking parameters, such as market size, market prediction, developer segmentation, and impact on the job market. Meanwhile, both *Google*¹⁷ and *Apple*¹⁸ displayed economic insights in their annual development conference. In addition, the app economy also attracted the interest of non-profit organisations, including government departments to understand both its infrastructure and impact on society and on the national economy. For example, the *EU* has already published several reports¹⁹ on the *EU* app ecosystem. Nearly all these research results confirm optimistic expectations regarding the app economy development. This thesis reviewed existing research and summarised the characteristics that are common to current popular app economy markets.

3.2 App Economy Characteristics

3.2.1 Installed User Base

The first evidence for optimistic expectation is from the users' significant shift to the mobile technology platform. *Localytics* pointed out in its 2014 report²⁰ that the amount of time people spend on apps has increased by a fifth over the past year. *comScore* showed in its 2014 report (Lella and Lipsman, 2014) that mobile app usage now exceeds time spent with desktop Internet access. Actually, it is a trend that the whole world is shifting from desktop to mobile, and

⁹<http://www.statista.com/>

¹⁰<http://www.portioresearch.com/>

¹¹<http://www.comscore.com/>

¹²<http://www.flurry.com/>

¹³<http://www.visionmobile.com/>

¹⁴<https://www.forrester.com>

¹⁵<http://www.gartner.com/>

¹⁶<http://research.gigaom.com/>

¹⁷<https://events.google.com/io2015/>

¹⁸<https://developer.apple.com/wwdc/>

¹⁹<https://ec.europa.eu/digital-agenda/en/news/sizing-eu-app-economy>

²⁰<http://info.localytics.com/blog/time-in-app-increases-by-21-across-all-apps>

many apps and its user experiences are already core to human life, like entertainment, banking, healthcare, etc. *Gartner* ²¹ reported that there are 1.9 million mobile phones shipped in 2014, and more than one billion users are using *Android* devices in 2014. This shift is not only in individual users, but also in business users. *VisionMobile*'s report ²² estimated that more than \$28 billions was spend on apps by business and professional users in 2013, and this figure will increase to \$58 billion in 2016.

3.2.2 Ecosystem

The second characteristic is the establishment of an ecosystem in the app economy. More and more tools and services have appeared to support the app's production and consumption (Petsas et al., 2013). Continuously apps generation greatly expands the user's selection on product, and increases the market size to attract more developers (Campbell-Kelly et al., 2014). *VisionMobile* survey reports that tool awareness is increasing in the app economy, and the percentage of developers using no third party tools has significantly dropped (Wilcox and Voskoglou, 2015). Over 6 months across their serial surveys, they reported that the percentage of developers adopting cross-platform tools, has increased from 23% to 30%.

The greatness of the ecosystem is the ability to 'cuddle' to other economy (Ghose and Han, 2014). Apps are not only for mobile devices, but also now connect to many other kinds of devices (Basole and Karla, 2012). For example, both *Android* and *iOS* are now able to connect to wearable hardware, smart home, automotive, healthcare, etc. Developers in the app economy are pushing the development of other economies. *VisionMobile* claimed that although *Internet of Thing* (IoT) platforms are still immature, mobile developers' interest is high (Wilcox and Voskoglou, 2015). They found that 53% of mobile developers were already working on some kind of IoT project in their survey. In addition, 37% of app developers are working on the smart home market, and 35% are in the wearable technology.

3.2.3 Competition

A large user base attracts a large number of producers according to (Basole and Karla, 2012). The virtual exchange nature in the app economy removes the geographic limitations, and enhances the competition into an extremely high level. In 2015, *VisionMobile* reported that there are 5.7 million app developers globally and 1.3 million are based in *EU* (Wilcox and Voskoglou, 2015). 795,000 of them are employed for *iOS* app development. In the UK, app related employment accounts for 1.1% of total employment. Regarding the employment structure,

²¹<http://www.gartner.com/newsroom/id/2645115>

²²<https://www.developereconomics.com/reports/business-productivity-apps/>

40% of full time professional developers are primarily working on *Android*, 37% on *iOS*, while *Windows Phone* accounts for 8% and the mobile browser for 7%. Another report (Pappas, 2014) estimated that in 2014, £4 billion revenues was generated by app companies in the UK app economy. Meanwhile, 48% of UK developers, 37.8% of Germany developers and 11.3% of French developers generated most of their revenues from sales outside Europe.

The digitising trend pushes competition across domains. Many traditional industries have moved their products and services to mobile: from retail²³, entertainment²⁴, ticketing²⁵, and etc. This leads to a large number of physical products, from toothbrushes²⁶ to shoes²⁷ which are being offered with an accompanying app. Beyond the demand for developer resources that this will create, apps will translate into new revenue streams for app enabled products. “*In the next 5 years, users expect that every company will have an app, much like every company today has a website*” (Wilcox and Voskoglou, 2015).

The low technical requirements push the competition to become fairer (Voskoglou et al., 2014). Historical wealth and technology accumulation plays less weight in the competitive advantage. Start-ups in the app business have less restrictions by age²⁸, gender^{29 30 31} and geography (Lim et al., 2015). *VisionMobile* confirmed that Asia is expected to have a greater weight on the overall trends of the app economy in the near future (Voskoglou et al., 2014). Hobbyists, students and other types of less informal business forces continually contribute to the overall performance of the app economy (Voskoglou et al., 2014). For example, *VisionMobile* reported³² that “*explorers and hobbyists (those developers seeking to learn, have fun and self-improve) make up 33% of the mobile developer population, even though they only generate 13% of the app economy revenues*”.

3.2.4 Cheap

Unlike traditional software products, apps are much cheaper to users. This is also a reason to stimulate the rapid increase of the user base (Ghose and Han, 2014). *Wall Street Journal*³³ reported that “*as of the end of 2012, the average price for a paid app in the Apple app*

²³<http://www.tesco.com/apps/>

²⁴<https://play.google.com/store/apps/details?id=bbc.mobile.news.uk&hl=en>

²⁵<https://play.google.com/store/apps/details?id=com.cineworld&hl=en>

²⁶<http://www.oralb.co.uk/en-GB/oralb-app>

²⁷<https://play.google.com/store/apps/details?id=com.nike.plusgps&hl=en>

²⁸<http://business.time.com/2013/03/14/ask-the-expert-the-best-age-for-a-start-up-founder/>

²⁹<http://www.wsj.com/articles/how-to-close-the-startup-gender-gap-1408912047>

³⁰<http://www.theguardian.com/small-business-network/2015/apr/13/gender-imbalance-tech-sector-martha-lane-fox>

³¹http://www.huffingtonpost.com/erica-swallow/y-combinator-gathers-800-_b_6730880.html

³²<https://www.developereconomics.com/reports/developer-segmentation-2013/>

³³<http://www.wsj.com/articles/SB10001424127887323511804578300183961009070>

store was \$3.18 on an iPhone and \$4.44 on an iPad, according to research firm Distimo. That compares with an average \$3.06 in the Google Play store and \$2.84 on Amazon.com Inc.'s app store. App stores generally take a 30% cut of the sale." Meanwhile, the freemium business model is popular in the app economy. Users can try the product first, and then pay only for the required function in the app, or even get a lot of apps that are complete free to users.

Even though an app is cheap, the total economy is huge. *VisionMobile* estimated in its 2014 report that at least 8,000 UK companies are engaged in the mobile app business (Pappas, 2014). However, one-man operations are commonly seen in a significant number of these companies (approximately 15%). It means that the income can be small, but the operation and cost of the app business can be tiny as well. Thus, app developers are still profitable. The large scale of user base, and the expanding speed (discussed in next paragraph), would aggregate a total economy into a significantly large number. *VisionMobile* (Pappas, 2014) estimated that the UK app economy exceeded £4 billion in 2014.

3.2.5 Speed

The app economy is high speed in both technology development and user consumption. The large user base and healthy ecosystem drive the high speed of market development. New tools and technology are continually invented or improved. Developers have to continually learn to keep their leading position. For example, *VisionMobile*'s development language rankings³⁴ showed 20% of mobile app developers were using *Swift* (*Apple*'s latest development language), and *Apple* only introduced it to the world just 4 months before the survey was carried out. However, 23% of *Swift* adopters were not using *Objective C* (*Apple*'s primary development language before *Swift*), which is a sign that a wider range of developers rather than a niche of dedicated developers are watching the development of *Swift*. Annual conferences of *Google* and *Apple* always attract a large number of developers who attend physically or watch online³⁵.

Another reflection of speed is on consumers' switching speed among apps. Apps' cheap costs lowers down users' switching costs between apps (Liu et al., 2014). *comScore*'s study showed that the top 7% of smartphone owners "accounting for developers adopt cross-platform tool" (Lella and Lipsman, 2014). The low switch cost weakens the lock-in restrictions to encourage developers to focus more on bringing value to users (Basole and Karla, 2012; Kenney and Pon, 2011; Schultz et al., 2011). Developers have more chance of survival by inventing or updating apps to attract new users or to maintain existing users (Möller et al.,

³⁴<http://www.visionmobile.com/product/developer-economics-q1-2015-state-developer-nation/>

³⁵<http://techcrunch.com/2013/03/13/google-io-2013-registration-sells-out-in-49-minutes-as-users-report-problems-early-on-making-payments/>

2012). Unlike traditional software, an app is easy to install, update and delete, by requiring less on technical skills. This ease of use characteristic also removes the difficulty for users to switch between apps (Campbell-Kelly et al., 2014).

3.3 The Google Play Store

In this section, I focus on a specific app economy market to elicit the problem to study in this thesis. *Android* and *iOS* are currently the two most popular app markets. *IDC*³⁶ has an annual report of the mobile market share since 2012. In 2014, *Android* is estimated to have 81.2% of market share, *iOS* has 15.2%, and then followed by *Windows Phone* having 2.5% of market share. Since *iOS* is closed to the public, it generated a difficulty for research data collection. *Android* is open sourced and I am able to collect significant historical data for study. *Windows Phone* has tiny market share and is mainly driven by hobbyists rather than commercial identities³⁷. Therefore, I chose to study *Android* as a research site to understand the app economy. The *Google Play Store* is the official online central market for *Android* apps. As I am focused on business strategy research rather than technology research, I am, therefore, basing my research study on the *Google Play Store* for this thesis.

The rest of this section briefly introduces the history of *Android* and the *Google Play Store*, and then reviews its problems and strategies. To recap the background, I reviewed all posts from six official online blogs that are relating to *Android*^{38 39} and the *Google Play Store*^{40 41 42 43}. These blogs act as a window for *Android* and *Google* to demonstrate and communicate to the public. As expected, these blogs received great attention and subscriptions by developers and users. After reviewing the blogs, I have collected all major events (including major releases, policy influence, strategy changes and rival competition), and listed them chronologically in Table A.11. Meanwhile, to understand *Google*'s strategy on the openness and control of *Android*, I collected the major events of the *Android* SDK major release, the *Google Play Store Service API* major release, the *Android Developers Challenge* contests, and aligned with disclosed information on the number of apps, as seen in Figure A.7. Different to other

³⁶<http://www.idc.com/prodserv/smartphone-os-market-share.jsp>

³⁷<http://www.theguardian.com/technology/2014/apr/03/can-cortana-and-other-new-features-turn-windows-phone-around>

³⁸<http://android-developers.blogspot.co.uk/>

³⁹<http://officialandroid.blogspot.co.uk/>

⁴⁰<http://googledevelopers.blogspot.co.uk/>

⁴¹<http://googleblog.blogspot.co.uk/>

⁴²<http://googleresearch.blogspot.co.uk/>

⁴³<http://googlemobile.blogspot.co.uk/>

well-known economies, there are many new terms specifically generated in the app economy. Therefore, I have made a table of terms for explanation in Appendix A.6.

3.3.1 History of *Android* and the *Google Play Store*

Bloomberg told the story of early *Android* in a 2015 news press release ⁴⁴. In October 2003, *Andy Rubin* and other developers founded *Android Inc.* in California to develop “smarter mobile devices that are more aware of its owner’s location and preferences”. *Andy Rubin* had previously started *Danger Inc.*, which designed and produced the first client-server or cloud based smartphone and created its app marketplace. *Android Inc.*’s early intention was to create a new operating system specifically for digital cameras, but they soon realised that the digital camera market was not big enough. The company then turned its attention to producing a smartphone operating system that would “rival *Symbina (Nokia)* and *Microsoft Windows Mobile*”. *Android Inc.* was acquired by *Google* in July 2005, and all major developers of *Android Inc.* were kept by *Google* to continue developing the *Android* system, including *Andy Rubin*.

The development of the *Android* system was secret until late 2007. On 5th November 2007, *Google* organised a consortium of technology companies to found the *Open Handset Alliance* (OHA) ⁴⁵, including device manufactures such as *Samsung*, *Sony* and *HTC*, wireless carriers such as *T-Mobile* and *Sprint Nextel*, and chipset makers such as *Texas Instruments* and *Qualcomm*. OHA announced its objective to collectively develop open mobile standards, and the first milestone achievement was *Android* (beta version) on that day. The *Android* system was built above a *Linux* kernel, and soon became open source with available source code for public download. Even after releasing the beta version, many bugs needed to be fixed and they finally released *Android* SDK 1.0 version on 23rd September 2008. The first commercial *Android* device was released on 22nd October 2008 by *HTC* (manufacturer) and *T-Mobile* (carrier).

After version 1.0, *Android* was frequently updating on its SDK with major released versions in 2009 (1.1, 1.5, 1.6, 2.0), 2010 (2.1, 2.2, 2.3), 2011 (3.0, 3.1), 2012 (4.0, 4.1, 4.2), 2013 (4.3, 4.4) and 2014 (5.0). By looking at Figure A.7, it is easy to see that many updates happened in the early stages with the purpose of improving *Android* into a stable mobile platform by having less bugs. The later stage was mainly to extend *Android*, by supporting more applications scenario and hardware that is broader than the mobile phone, such as tablets, TV, wearable devices, and etc. Meanwhile, *Google* also embedded its proprietary services into *Android* SDK. In the early versions, those services were strongly coupled with *Android* SDK, but with the rise

⁴⁴<http://www.bloomberg.com/bw/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal>

⁴⁵<http://www.openhandsetalliance.com/>

of the *Android* fragmentation problem (discussed more in Section 3.3.3), *Google* decided to lose the coupling by making *Google Play Store Services API* independent from *Android* SDK from 26th September 2012. *Android Markets* were set up by *Google* on 22nd October 2008 to allow developers to sell apps to users, and then later renamed it to the *Google Play Store* on 6th March 2012 by integrating other virtual products, like music, books, movies and etc.

Android apps can be distributed to other third party marketplace, and there are several well-known third-party online stores, like *Amazon Appstore* ⁴⁶, *AppBrain* ⁴⁷, *Samsung Galaxy Apps* ⁴⁸, *Opera Mobile Store* ⁴⁹, and etc. Many of those third-party markets are duplicated to *Google Play Store*, which means developers who released apps to third-party markets also publish on *Google Play Store* (Petsas et al., 2013). Most developers and users choose *Google Play Store* to release and buy apps. Therefore, I only focus on *Google Play Store* as the primary market for *Android* apps in this thesis. I then discuss the problems and relevant strategies observed from *Android* and the *Google Play Store* history.

3.3.2 Development

Android required huge efforts at the beginning of its development stage. The first *iOS* SDK was released on 17th October 2007 to developers and the first *Android* SDK was later released on 5th November 2007. They released SDK in very close dates, but there is a huge date gap from the release of the commercial device for users to buy. *Apple* released the first *iPhone* device on 29th June 2007, and the second generation of *iPhone* was released on 11th July 2008. After the success of *iPhone 1*, *iPhone 2* had already attracted a great deal of attention from both developers and users. However, the first *Android* device was released already three months after *iPhone 2*, and this *Android* device was not released by *Google*. Meanwhile, there were still huge efforts required to improve the *Android* SDK. Figure A.7 and Table A.11 confirmed that there are many *Android* SDK updates from 2007 to 2009. The official *Android* website ⁵⁰ only recognises *Android* SDK 1.6 (released on 15th September 2009) as the earliest officially recognised stable version. All this indicates the weakness of *Android* in its early stage.

Open strategy was significantly beneficial in solving the weaknesses that accompanied *Android*. In contrast to *Apple*, who was putting a team of 1000 employees in *iOS* development ⁵¹, *Google* adopted various methods to overcome the weaknesses of *Android* in its early stages.

⁴⁶<http://www.amazon.co.uk/mobile-apps/b?node=1661657031>

⁴⁷<http://www.appbrain.com/>

⁴⁸<http://seller.samsungapps.com/>

⁴⁹<http://developers.apps.opera.com/>

⁵⁰<http://www.android.com/history/>

⁵¹<http://allthingsd.com/20120803/apples-scott-forstall-on-how-project-purple-turned-into-the-iphone/>

- The **first** method was to establish an alliance (OHA), so as to promote collaborative production. In contrast to *Apple*, the first *Android* device (released on 22nd October 2008) was not produced by *Google*. In fact, *Google* only released its own brand of *Android* device in January 2010. This collaboration enabled *Google* to focus on its specialists and to collaborate with other alliance members who were contributing on other specialized core activities.
- The **second** method was to make *Android* into an open source. By giving up on the absolute control of *Android*, it returned with a large number of developers from various backgrounds to contribute to the improvement of *Android*. The open source style attracted a lot of geeks, hobbyists and people from educational sectors who wanted to demonstrate their ability and to improve *Android*.
- The **third** step was to reward developers with contests to motivate them as well as the users to encourage more innovation on *Android*. *Android Developer Challenge I* (2008)⁵² and *Android Developer Challenge II* (2009)⁵³ were two major contests hosted by *Google* to attract developers in the early stage. The first contest attracted 1788 entries from 70 countries. The second contest successfully pushed *Android Market* to both developers and users. *Google* also continually sponsors other workshop and conference events to promote *Android* development.
- The **fourth** method is to provide easy-to-follow documentation and showcase developers' work in official *Android* channels. Developers have a clear channel to follow and communicate.

Those methods enabled *Google* to focus on infrastructure development, and encourage open and user innovation on peripheral products. *Android* soon developed into a strong ecosystem.

3.3.3 Fragmentation

The negative impact of open strategy is the *Android* fragmentation issue (Ham and Park, 2011; Han et al., 2012a; Pon et al., 2014). Fragmentation in computing happens when a combination of software and hardware do not provide a consistent, top-level experience for the vast majority of its user-base. Fragmentation is firstly caused by hardware inconsistency (Han et al., 2012a). Due to the open policy, different manufacturers customised devices with different screen size, sensors and computing capability to fit into the specific market target. This diversifies the competition between manufacturers, and attracts broader users to join in the *Android* market. *OpenSignal*⁵⁴ reported there were 3997 distinct models of *Android* devices in 2012. However,

⁵²<http://android-developers.blogspot.co.uk/2008/08/presenting-winners-of-android-developer.html>

⁵³<http://android-developers.blogspot.co.uk/2009/11/announcing-winners-of-adc-2.html>

⁵⁴<http://opensignal.com/reports/fragmentation.php>

app developers were challenged by app development (Ham and Park, 2011). For example, the same app may not look the same on different screen sizes; the same app may not work on some devices due to the absence of certain hardware sensors; the same app can also run slower in certain low specific mobile devices. It brings uncertainty for app developers and inconsistent app experience to users. Even more, apps are more likely to crash with those uncertainties, which gave an impression of average or low quality for the *Android* platform⁵⁵. Opposite to *Android*, *iOS* has experienced fewer problems with this type of fragmentation, as *iOS* apps are running on, and specifically designed for, the same specific model of *iPhone* (Yamakami, 2011).

The second reason for fragmentation is due to the different versions of operating system (McDonnell et al., 2013), and this is happening for both *Android* and *iOS*. The early *Android* SDK still has bugs and often needs to be updated to support advanced function, however, updating can cause users to run different versions of *Android* at one time. Developers then face the problem to decide which SDK they should adopt in their apps. Supporting a newer SDK allows developers to incorporate new features in the app, however, many users lose the opportunities of experiencing new features if they are still using an old version of *Android* (Gawer and Cusumano, 2008). Supporting an older SDK will limit new features, and older SDK generally have more bugs. In the meantime, updating on SDK needs to be done by manufacturers, however they all have various levels of ability when it comes to updating SDK and many might choose not to update the devices that are no longer being manufactured, although they are still being used in the market.

Both hardware and software fragmentations generated difficulties for *Google* and the communities to push *Android* forward in the market (Ham and Park, 2011; Han et al., 2012a). *Google* and the communities have had to spend a large amount of resources to work on backward compatibility, instead of innovation (Gawer and Cusumano, 2013); and this has significantly impacted on the speed of innovation (Yoon, 2014). Without enough efforts on growth and innovation, the newer SDK would have very little market size and therefore the developers would be less likely to create new app features with new SDK.

Google adopted multiple steps to solve this problem.

- The **first** step was to modify the *Android* policy⁵⁶ to prevent fragmentation, saying that: “You agree that you will not take any actions that may cause or result in the fragmentation of *Android*, including but not limited to distributing, participating in the creation of, or promoting in any way a software development kit derived from the SDK”.

⁵⁵<http://www.cnet.com/uk/news/is-android-fragmentation-over-with-jelly-beans-rise-nope/>

⁵⁶<http://developer.android.com/sdk/terms.html>

- The **second** step was to demonstrate updates on their own brand *Nexus* device as a "how-to" for other manufacturers, which is to stimulate others manufacturers to update on their devices and also to demonstrate to them how to update as well.
- The **third** step was to improve the SDK update technology and make it automatic on users' devices ⁵⁷. This enables users to receive system updating notifications from cloud automatically allowing users to do their updates themselves, rather than requiring manufacturers to spend extra resource on upgrading the devices. Devices that are too customised may experience issues updating directly from the cloud causing manufacturers to upgrade their customising system in order to maintain market attraction. It may also influence the community to grow a cloud-based development culture by synchronising ⁵⁸ online and private services across multiple devices for the same app.
- The **fourth** step was to separate its proprietary services from *Android* SDK. Each manufacturer has a different level in technology development and integration, which is the main reason to cause hardware fragmentation. By separating from *Android* SDK, *Google* updates its services independently from manufacturers and other developers via the *Google Play Store Service API* ⁵⁹. This ultimately opens a separate channel for users to directly benefit from *Google*'s powerful innovation ability. By doing this, *Google* can focus more on core innovation and provide value to users directly from this channel (Basole and Karla, 2012). *Android* history (Figure A.7 and Table A.11) tells that *Android* SDK has been less frequently updated after 2012, but *Google Play Service API* has been frequently updated since then.

The separation of *Android* SDK and the *Google Play Store Service API* also enables *Google* to innovate on developer supports, by developing infrastructure to speed up open and user innovation. Even through the *Google Play Service API* is proprietary, it actually works as an infrastructure to support the common works by developers, allowing them to focus on their own app innovation. The official *Android* blog clearly explained the motivation ⁶⁰ by saying: "*Most Android apps have some sort of server-side back end, to persist and share data. Even the most basic game needs to remember its players' high scores. When you're building your back end, one problem you have to solve is how the back-end code knows what app it's talking to and who the person using it is. You probably have HTTP endpoints for communicating with your client apps, but how can the server-side code be sure who's sending messages to it? After all, anyone can send HTTP POST requests from anywhere; could they impersonate your users if they could guess their identities? It's really user-unfriendly to ask people to type in*

⁵⁷<https://support.google.com/googleplay/answer/113412?hl=en-GB>

⁵⁸<http://developer.android.com/training/cloudsync/index.html>

⁵⁹<https://developers.google.com/android/guides/overview>

⁶⁰<http://android-developers.blogspot.co.uk/2013/01/verifying-back-end-calls-from-android.html>

usernames and passwords on mobile devices. In particular, if someone has installed your app and given it permission to use the Internet and know your identity, they shouldn't be pestered any more. It turns out that Google Play services, now available on every compatible device running Android release 2.2 or higher, offers a good solution to this problem, based on the use of Google Accounts. With Google Play services, your app can take advantage of the latest, Google-powered features such as Maps, Google+, and more, with automatic platform updates distributed as an APK through the Google Play store. This makes it faster for your users to receive updates and easier for you to integrate the newest that Google has to offer. Google Play services gives you the freedom to use the newest APIs for popular Google services without worrying about device support. Updates to Google Play services are distributed automatically by the Google Play Store and new versions of the client library are delivered through the Android SDK Manager. This makes it easy for you to focus on what's important: your users' experience."

3.3.4 Competition

The great success of the *Google Play Store* has led to the significantly extreme competition. This high competition is also existing in *iOS*, and seems to be appearing in other future economies with similar market structures. *StarDust* reported in 2012 ⁶¹ that 41% of 484,271 apps in the *Google Play Store* had never been updated or have fewer than 10 customer reviews, the same as 65% of the 617,436 apps in *Apple App Store*. It was a clear signal that those developers have not managed to reach a significant level of app usage, or a significant level of app investment return. Even more, most apps only have two weeks to get the majority of reviews and downloads ⁶². In this case, after two weeks, users will switch to other apps.

Other, more detailed, figures explaining the issue of app reviews are as follows. After releasing to public, the average *Android* apps get 80% of user reviews in the first 16 days. In fact, 68% of all *Google Play Store* apps are downloaded fewer than 5,000 times. Meanwhile, each and every day in average, there are 2,371 new apps published, and 2,341 *Android* apps are updated. *StarDust* attempted to explain that: "This can be explained by a common practice of *Android* developers who publish an update in order to position their applications into the top ranking. It also can be due to the fragmentation of the platform that requires more maintenance and regular patches."

Over competition increases the cost for users to search for desired apps and the opportunity cost of good developers. Users have no time to try all new apps. Good developers can fail

⁶¹http://www.stardust-testing.com/ressources_1/notice-m-reputation-en.pdf

⁶²<http://venturebeat.com/2013/08/26/700k-of-the-1-2m-apps-available-for-iphone-android-and-windows-are-zombies/>

in their businesses, not due to the low quality of app, but just because the developer did not choose the right time ⁶³ to release and, therefore, they miss the opportunity to expose their app to users properly (Möller et al., 2012). A solution adopted by *Google* is to display apps in selection order, showing high quality apps at the top for the user therefore promoting high quality developers' apps first. This thesis will discuss more on this selection mechanism later.

3.3.5 Innovation in the App Economy

Although an app economy has been mentioned in the news and studied in industry, it is not yet deeply explored in the academic world. During the time of writing this thesis, searching for “app economy” keywords in *Google Scholar* returns with many technical academic articles (such as privacy control (Chia et al., 2012; Felt et al., 2012; Sarma et al., 2012), user experience (Ickin et al., 2012; Lin et al., 2012) and technology development (Chin et al., 2011)), but with very little choice for business related academic literatures (Basole and Karla, 2012; Boudreau, 2012). Technology is key to the platform development, whilst management and business knowledge make sure the development is in a healthy condition. As seen previously, there is awareness that open strategy has helped *Android* to consolidate multiple resources to develop the technology, improve functional features, and expand the market into an ecosystem. However, this also leads to the fragmentation issue and high competition. Therefore, the understanding of managerial strategy is valuable for the sustainability of the app economy.

Nowadays, the app market has developed into a very competitive market only in just a few years. Competition contributes to innovation, but such high competition also leads to negative issues (Gilbert, 2006). I have already discussed fragmentation issue, and continue to discuss other negative issues in this section. Innovation is the key to maintain the high speed development of the app economy, and developers are the key to produce innovation (Lin et al., 2011). Therefore, how to attract and motivate developers to innovate is key to keeping the sustainability in the app economy. An app economy is a highly networked market, so to answer this question, we need to consider multiple parties as a unit, which increases the complexity of this study. Meanwhile, an app economy is developed from mobile technology and cloud computing, and it would greatly enhance the forthcoming economy such as *Internet of Things* and *Big Data*. So the value of this study is also to provide a guidance of strategic innovation for future markets.

Meanwhile, the negative issues including fragmentation are generated due to the inconsistent behaviours and benefit consideration between different parties in the ecosystem (Yoon, 2014). The inconsistent behaviours enhance the information asymmetry between different parties

⁶³<http://techcrunch.com/2011/12/19/sunday-is-the-best-day-to-launch-your-mobile-app/>

and damage to innovation ability. The rest of this chapter discusses this particular problem to understand how it is generated. The problem identified in this section guides the literature review in next chapter, and provides the foundation to the theory proposed in Chapter 5.

3.3.6 Problem of Competition in the App Market

Android is a two-sided market (Kouris and Kleer, 2012). To be more specific, the *Android* app market is constituted of three parties based on the *Android* platform. The first party is the market operator who owns, manages and governs the market. The second party is the producer who sells the product in the market. The third party is the consumer who buys the product. The main interest for the market operator is on market growth, as its revenue generates from the trading commission between producers and consumers. Therefore, the market operator takes the responsibility to develop a better infrastructure for trading between producers and consumers, and conducts governance among all individuals. By trading in a central place, producers are competing with each other to attract consumers, and consumers are selecting desired producers for their demands.

Sometimes, the individuals in three parties can play different roles. For example, the market operator may also sell own-brand products in the market. The producer may buy goods from or rely on other producers. Consumers may develop tools if their demand can not be fully satisfied by a product. In reality, there would be many complex ambiguous roles that exist like those. However, to abstract the research context, the thesis assumes that three parties are clearly distinguished by ignoring temporary and minor ambiguous roles. Therefore, to be specific on *Google Play Store*, the three parties are including: 1) *Google* as the market operator who is running the online marketplace for *Android* apps, 2) developers as producers who are creating apps to release on the *Google Play Store*, and 3) users as consumers who are buying apps from the *Google Play Store*.

In the previous section, I have discussed that high competition exists in the *Google Play Store*. High competition represents to developers with an increasing pressure and a higher desire to differentiate their products (Anderson et al., 2010). In order to survive from the competition, they have to either improve the existing quality (Evans, 2009; Katila et al., 2008), or experiment with new apps to target user demand which has little or no competition (Parrish et al., 2006; Santos and Eisenhardt, 2009). Both actions would expand users' choices on product, and could increase the user satisfaction of platform (Sawhney, 1998). Users would love to see more products of high quality and maybe cheaper price (Basole and Karla, 2012). They are also happy to see a new product for the functionality that they cannot find from existing products (Lakhani and Von Hippel, 2003). Users would share and spread positive user experience to other individuals, which would create the network effect of bringing more new users into the

market (Lim and Bentley, 2012a). A larger user base and higher spending speed would boost the market size (Rysman, 2009). This would create a signal to attract more developers to join the market (Zhu and Iansiti, 2012). The influx of developers would increase competition. This would form an iteration on growth (Jansen and Bloemendal, 2013).

However, the iteration may not always run in a positive direction. Massive choices on products would increase the selection time and cost for users to find the most desired product (Hyrnsalmi et al., 2012b). Meanwhile, market growth would not always attract good developers, but also attract defective developers (Boudreau, 2012). Their products disturb the market operation by causing damage to the user experience and good developers' benefits. This undesired attraction is extremely true in the *Google Play Store*, because the entry boundary in the *Google Play Store* is very low. Developers only have to pay a one-off \$25 registration fee⁶⁴, which gives unlimited access to submit apps to the *Google Play Store* for reviews and then release to all users in the *Google Play Store*. *Google* makes the first check on the app before releasing it to users, but in a massive competitive market, it would be difficult for *Google* to guarantee the quality for every app. Also, many apps are actually elusive to purposely confuse the reviewer⁶⁵. Defective products would also significantly damage the users' experience and increase more difficulty for the user to select the right app. Therefore, a challenge for all parties in the market is to reduce the entry of defective (or noising) apps and developers.

The challenge is not easy to overcome. Simply increasing the boundary for entry would increase the opportunity costs for defective developers as they are generally less profitable, but it would also increase the difficulty for good developers to profit (Pehrsson, 2009). Enhancing the initial review process would filter out defective products in the early stages, but this would increase the cost for the market operator and extend the review time which would also negatively effect on good products (Felt et al., 2011c, 2012). In reality, new apps that are massively added everyday in huge numbers which actually makes it impossible for *Google* to carefully examine each source code in each app to adequately guarantee the quality⁶⁶. So this kind of equal treatment strategy seems either be very costly or damages the benefits of good developers. Meanwhile, the *Google Play Store* also competes with other apps in the market (Pon et al., 2014), like the *Apple App Store*. The market operator needs to make sure the market has a large choices of products, and the above solutions would reduce the attractiveness to both developers and users.

All parties in the markets are proposed to collaborate to solve this challenge. The market operator encourages competition to improve the average product quality and to increase the

⁶⁴<http://developer.android.com/distribute/googleplay/start.html>

⁶⁵<https://play.google.com/about/developer-content-policy.html>

⁶⁶<http://venturebeat.com/2013/08/26/700k-of-the-1-2m-apps-available-for-iphone-android-and-windows-are-zombies/>

market size (Karhu et al., 2014). Users can give feedbacks with reviews and comments to disclose the real quality to public (Fu et al., 2013). This allows defective developers to be filtered out by market selection and lets good developers to be standout. The market operator has a long term planning strategy for the market development, therefore the market operator would select good developers to protect and promote their high quality (Basole and Karla, 2011). Rather than trying every product, consumers would rely more on platform selection as both users and the market operator share the same goal (Chia et al., 2012). This cooperation between consumers and the market operator would enhance the effect of selection, pushing developers to learn more from competitors selected by the market operator (Remneland-Wikhamn et al., 2011). The population of peer learning eases the market operator's governance job, as the market operator can focus more on sending signals to developers to spread the desire for a healthy and sustainable market environment.

3.4 Information Asymmetry in App Market

Information asymmetry (Stiglitz, 2002) causes the problem that developers are not be easily distinguished from one another. To drive a positive iterating ecosystem, the key is to improve developers and their apps' average quality. Both good and defective products will "promise" to the public that they are of high quality (Spence, 2002). Users can not always understand and distinguish the quality of the apps before purchasing (Connelly et al., 2011). Even if purchased, users may still not be able to examine the hidden behaviours of apps (Connelly et al., 2011) (for example, a malicious app can steal data but it is difficult to observe by users without technical skills, etc.), if the producer has tried to cheat. There is a limited amount of information disclosed to indicate the quality of a product and producer in the market, so it becomes difficult for the market operator to inspect the quality of developers with a fine granularity.

Google, as the market operator, is working hard to reduce the information asymmetry on the *Google Play Store* by trying multiple mechanisms.

- Firstly, *Google* developed an artificial intelligence service called "Bouncer"⁶⁷ to detect malware app by regularly scanning each app to get more insight into its quality.
- Secondly, *Google* is trying to help users to narrow down the selection range by only showing them the most relevant information. The *Google Play Store* has a category system and each app needs to choose for a category under which to publish. In each

⁶⁷<http://googlemobile.blogspot.co.uk/2012/02/android-and-security.html>

category and price type, *Google* would make certain “window exhibition” by listing the top 200 apps (see Figure A.12).

- For each app profile, *Google* would also list all the developer’s all other apps and other similar apps by other developers for benchmark(see Figure A.10).
- In addition to this, users can post their opinions for each apps have used (see Figure A.11). This peer review provides an indication of the quality, even though, peers’ options sometimes can also be confused by fake actions ⁶⁸. *Google* tried to aggregate review information in multiple ways to minimise the ambiguity by unintended behaviour (see Figure A.13).
- The last mechanism is to set up the selection reward and disclose the selection result to the public. There are two types of rewards in the *Google Play Store*. One is “Editors’ Choice”, and the other is “Top Developers” (see Figure A.14 for a developer having both rewards). “Top Developer” is a lifetime reward, and all of their apps display this reward badge on the app profile.

These mechanisms basically provide users with a lot information for each product, for example, many historical records for an app and also whether it has been selected by *Google* for “window exhibition”. “Window exhibition” is a great opportunity for developers to increase their public exposure and attract users. Therefore, this is an opportunity that developers are aiming to compete for. As it is impossible for us to track the change of app rank all the time, I am using the “Top Developer” reward as an indication for selection by the market operator. So, for the rest of this thesis, the only focus is on the effect of “Top Developer”.

Due to the information asymmetry, indirect reciprocity happens between developers and consumers (Seinen and Schram, 2006). Users have difficulty to completely distinguish the quality of products, and it presents as high search cost for users to find desired products. Users may rely too heavily upon the market operator’s selection. Good developers want to separate from defective developers, so they wish to gain approval from the market operator to show their high quality and ability. In other words, consumers are more likely to buy products from developers who are selected by the market operator. Developers sell the products to consumers through the market and get a stamp of quality approved on the product by the market operator.

This indirect reciprocity provides the market operator with an opportunity to influence the market selection. By judging on selection, the market operator can signal their desire to all parties. Only products which meet the market operator’s desire will be selected by the market operator. Since users are more likely to buy a selected app, it makes a desired app profitable and the defective developer less or not profitable. This would generate a crowding out effect for defective developers and their apps (Boudreau, 2012). In reality, *Google* never disclosed

⁶⁸<https://support.google.com/googleplay/android-developer/answer/2985810>

the exact details of its criteria on how to select a featured app. This also generates a learning effect for new coming or defective developers to learn from existing selected developers. This learning effect can generate a network effect on the sustainability of market growth, as more and more developers are following the market operator's desire. In a long term perspective, the ecosystem will iteratively develop into a positive direction.

3.5 Summary of Problem

In summary, an app economy market, such as the *Google Play Store*, needs to have a large number of developers to innovate enough products in order to increase its user base. In order to attract developers, market operators like *Google* are attempting to lower down the boundaries of entry to attract developers to enter the market for growth of the *Google Play Store*. However, this attempt attracts not only good developers but also defective or even malicious developers, and makes the market very crowded with high competition. Defective developers have less interests on innovation and more interests on short-term benefits. They increase users' search costs and cause damage to good developers' benefit, and then impact on the healthy iteration of ecosystem.

Therefore, *Google*, as the market operator, selects high quality developers and their apps to gain exposure to users. After selection, developers gain great benefits in terms of business success, and push other developers to learn and compete for selection opportunities in such a free market. However, information asymmetry causes an imbalance in information between individuals, and leads to the difficulties in selection. *Google* is unable to review each app's source code to detect every features, as the number of apps and daily updates is so huge. Even so, many developers' behaviour and intentions are difficult to detect, especially as some of them try to hide, cheat or fake. Low quality developers try best to demonstrate their false qualities by interfering with the selection process. So it drives the motivation for good developers to signal their true high quality to secure their selection opportunity, which protects their innovation and let them continually focus on innovation in the market.

Google needs to gather multiple information sources to make their selection decisions. Therefore, developers' signalling is targeting on multiple parties in the market, including users, *Google* (the market operators) and other developers. Users purchase apps and then review the quality, but their purchase decisions are significantly influenced by platform selection. Good developers want to stay with other good developers to keep innovation ability under healthy competition for the aim of a healthy ecosystem. Other developers' entry helps good developers to claim the market, but this enhances the competition and crowding out effect on good developers. *Google* would rely on user feedback to aggregate the accumulated performance of

apps for selection, but it would also need to consider newcomers who are innovative but have no establishment to make sure of fair competition. Google should also be careful in their selection by not being subjective, as its selection decision is also a signal to the market which can heavily influence learning and innovation cultures in the market. The complexity of this problem is due to the network nature of two-sided markets, as in such highly competitive ecosystem.

In summary, this thesis investigates how developers signal to the market in such a crowded two-sided market. This study evaluates the efficiency of the selection mechanism by *Google* to manage the innovation in the *Google Play Store*. To understand this better, there is a full literature review in the next chapter.

Chapter 4

Literature Review

Chapter 2 briefly introduced the problem and solution to the app economy. This chapter aims to systematically review the literatures by specifically focusing on three topics: platform strategies, strategic behaviour and signaling theory. I have listed all the important literature by topics in Table A.7, Table A.8, Table A.9 and Table A.10.

4.1 Platform Strategy

The concept of platform has appeared in the researches in the last two decades. The concept emerged in the 1990s when companies started to build a series of related products on a foundation or base of common components (Cusumano, 2010). Scholars used it to explore the costs and benefits of modular product architectures and the reuse of components (Yoffie and Cusumano, 1998), by studying on the technologies like Microsoft Windows, PC and browsers, etc.

Gawer and Cusumano (2002) took this challenge to study industry platforms that provide reusable functions as part of technology "systems" whose components are likely to come from different companies (or maybe different departments of the same firm), which was called "complementors". They argued that an industry platform has relatively little value to users without these complementary products or services. Hence, in order for their technology to become an industry-wide platform, companies generally must have a strategy to open their technology to complementors and create economic incentives (such as free or low licensing fees, or financial subsidies) for other firms to join the same ecosystem and adopt the platform technology as their own (Gawer and Henderson, 2007).

4.1.1 Platform Thinking

This forms the idea of "platform thinking" (Cusumano, 2010), which is to establish dynamic network effects to encourage more external adopters in the ecosystem that create or use complementary innovations to enhance the value of the platform (and the complements). In other words, this kind of direct or indirect network effect attracts more users to adopt the platform, and more users' adoption attracts more complementors to enter the ecosystem, and then more complements attract even more users to adopt the platform, and so on to form an infinitive iteration of attraction.

This kind of network effect also received attentions from scholars in two-sided markets of field and network economics. Their collaborative researches (Gawer, 2011) has expanded the knowledge on the emergence, leadership and competition of platforms, for example how platform industries or products are affected by standards and technical compatibility (Caillaud and Jullien, 2003; Cusumano, 2010), the phenomenon of network or positive feedback effects (Bauer, 2014; Parker and Van Alstyne, 2005), and the role of switching costs and bundling (Farrell and Klemperer, 2007; Parker and Van Alstyne, 2000). This section identified some of these articles (see Table A.7), including pricing, tying, boundary resources, exclusive contracting, trust, open, barriers, switching cost and regulation, to discuss two kinds of purposes in platform strategies: attracting developers into the platform for complements (attraction for complements) and differentiating the quality of complements (differentiation of complements).

4.1.2 Attraction For Complements

A lot of research has been dedicated on how to attract for internal and external complements in order to make full use of network effect in the platform or two-sided markets. It fitted to the business trends that a lot of platform businesses established in the last two decades, for example, payment card markets (Armstrong, 2006; Rysman, 2009), advertisement & media markets (Anderson and Gabszewicz, 2006; Kaiser and Wright, 2006; Von Ehrlich and Greiner, 2013), matching markets (Caillaud and Jullien, 2003; Damiano et al., 2008), B2B markets (Yoo and Choudhary, 2003), video game console markets (Rochet and Tirole, 2003, 2006; Srinivasan and Venkatraman, 2010), operating system markets (Eisenmann et al., 2006; Parker and Van Alstyne, 2000), computer hardware (Gawer and Cusumano, 2008; Gawer and Henderson, 2007). Most of this research focuses on financial strategy (pricing, tying) and exclusivity (contracting), etc.

Economists focused on financial strategies (Armstrong, 2006; Baldwin and Woodard, 2009; Caillaud and Jullien, 2003; Hagiu, 2006; Rochet and Tirole, 2003, 2006), and have particular interests in studying the price structure (Rysman, 2009) and tying mechanisms (Amelio and

Jullien, 2012). The research methods they popularly used are mathematical modelling and theory simulation (Cusumano, 2010). They argued that platform providers have to choose a price for each side, based on the assumption of the other side's growth and willingness to pay (Eisenmann et al., 2006). They found that a platform or a two-sided market always has a "subsidy-side" and a "money-side". "Subsidy-side" is always attracted to volume based on the value of "money side". By applying the app economy, app developers are the "money-side", and app users are the "subsidy-side". App developers are always attracted to an app market having a large number of app users. Because the number of subsidy-side is crucial to developing strong network effects, the platform provider sets prices for that side below the level it would charge if it viewed the subsidy side as an independent market. The goal of pricing is to generate "cross-side" network effects: if the platform provider can attract enough subsidy-side users, money-side users will pay handsomely to reach them.

Two kinds of subsidy-side are of particular interested: when subsidy-side users are price-sensitive, or when subsidy-side users are quality-sensitive. Generally, it makes sense to subsidise the network's more price-sensitive side and to charge the side that increases its demand more strongly in response to the other side's growth. Adobe Reader is an example of widely studied case having a price-sensitive subsidy-side (Rochet and Tirole, 2006), so it would provide Adobe Reader for free to final customers to expand the market size and charge for professional PDF editing software. When the subsidy-side users are sensitive to quality, economists suggested that rather than charge the side that strongly demands quality, the platform should charge the side that must supply quality. Such a strategy is evident in the video games console markets (Rochet and Tirole, 2003). The platform often sells the video games console below under the production price to attract a large amount of users, and charge game developers for the development license and commission of sales.

Some literature theoretically demonstrates that a platform may even set "negative prices" on one side in order to enhance participation (Armstrong, 2006; Caillaud and Jullien, 2003). Tying is a popular mechanism to implement "negative prices" in platform strategies. Tying is to make implicit subsidies in the markets to raise participation and benefits on both sides (Choi, 2010). Financial tying can rapidly accelerate participation quickly, but may not be sustainable in the long term (Amelio and Jullien, 2012). Non-financial tying which improves one side's productivity or marketing efficiency or offers some free products as a bundle is starting to be recommended (Amelio and Jullien, 2012; Lee and Wu, 2009). For example, Ghazawneh and Henfridsson (2012) have studied the importance of providing efficient development environments and toolkits to app developers in order to help them innovate more efficiently.

Scholars also studied the attraction by competing across platforms. Platforms are competing against each other to attract users to join in the markets to increase the market size. Therefore,

a key job for the platform is to establish and maintain its competitive advantage (Chen and Miller, 2012). Platform envelopment (Eisenmann et al., 2011) is a widely adopted strategy to maintain the competitive advantage by learning and adopting new features from other platforms. Via observing developers' switching behaviours across platforms, a platform can identify the missing new features and adopt them into the technologies of its own platform. This reduces the differentiation between platforms and discourages the attraction from rival platforms by increasing the developer's opportunity cost (Farrell and Klemperer, 2007).

In the context of the app economy, app price is quite low, as most apps are free or less than one dollars (Liu et al., 2014) in the *Google Play Store* and *Apple App Store*. In the early stage of the *Google Play Store*, app consumers have a lower expectation of quality of the *Android* apps in comparison to *iOS*, but this exception has been significantly increased as *Android* took the largest market share. The platform should consider putting more efforts into improving the governance of app developers in duopoly based on existing pricing strategies. Meanwhile, the *Google Play Store* and *Apple App Store* are using similar pricing strategies at the moment, so financial tying may not very suitable at this stage. Amelio and Jullien (2012) even argued that in a duopoly, tying on one side makes a platform more or less competitive on the other side depending on externalities. Both the *Google Play Store* and *Apple App Store* are active and innovative in providing boundary resources to their developers to improve the platform envelopment. It basically means the platforms in a duopoly would have a similar market size and adopt similar pricing strategies, to complete in a similar way. Therefore, other attraction mechanisms should be identified to fit into the duopoly context of app markets.

4.1.3 Differentiation Of Complements

The attraction for complements plays an important role for a platform to emerge and gain leadership in the competition. However, not all complements are in the same high level of good quality (Evans and Schmalensee, 2010). Low quality complements may damage the user experience, and impact on satisfaction of the subsidy-side users who are sensitive to quality or brand value (Rasch and Wenzel, 2013). Therefore, another interest of platform strategy is to differentiate the complements to control the average quality level in the markets. The common methods of differentiation include setting up entry barriers and setting up exclusive contracting.

Every platform has entry barriers for complementors and complements (Rysman, 2009). A common barrier can be the technology dependency. Developers have to learn the technology in the platform in order to make products based on the platform technology. This technology dependency also generates the lock-in to increase the opportunity costs for all parties in the markets (Farrell and Klemperer, 2007). Other barriers can be price barriers such as setting up a registration fee, transaction fee or commission fee. In order to compete with other developers,

developers need to R&D on innovative products therefore, the requirement on innovation is another kind of barriers (Lin et al., 2011). The platform will often lower down the price barrier to attract more consumers for participation, but increase the barrier on innovativeness to control the quality (Lin et al., 2011). Some researchers have found that certain barriers like restricted access and control over the platform may lead to more investments in innovation (Boudreau, 2008).

Recently, a stream of scholars have demonstrated the interest in setting up the barriers in the app market by studying on the technology solutions to identify the app qualities. Many of them (Felt et al., 2011c; Sarma et al., 2012) attempted to examine the permission request in *Android* apps to identify malicious and piracy behaviours. Some of them (Chia et al., 2012) attempted to rely on crowd wisdom, which is to study the user reviews on apps. Those studies made a contribution to influence the design of the *Android* system and the malicious apps scanning service in the *Google Play Store*, but there is still a lack of the ability in identifying the apps' innovation. Other managerial barriers should be developed in addition to technology barriers in the app economy.

Exclusive contracting refers to subsidies and contracts to some particular developers in a platform that does not allow them to sell to other platforms. Exclusive contracting is a mechanism to keep the unique features of a platform in order to improve the competitive advantage (Hagiu and Lee, 2011). For example, the video game console platforms often use this mechanism to make a particular game title only available on its own platform, therefore, users who are interested in this game title can only choose to join in this platform not the rival platforms (Armstrong and Wright, 2007). Developers who are in exclusive contracts normally receive huge benefits from a platform by giving up their selling opportunities in other platforms. These benefits include huge financial subsidies, great marketing opportunities or large user attention, so that developers can concentrate their resources to working on and innovating a single platform (Hagiu, 2009a). Meanwhile, the participation of "marquee users" can be especially important for attracting participants to the other side of the network (Eisenmann et al., 2006). Marquee users may be exceptionally big buyers, like the U.S. government, or they may be high profile suppliers, like anchor stores in malls. A platform provider can accelerate its growth if it can secure the exclusive participation of marquee users in the form of a commitment from them to not to join rival platforms.

In the context of the app economy, the *Google Play Store* and *Apple App Store* have adopted similar barriers strategies at this current time. For example, developers have to pay similar registration prices and commission rates in both markets. The *Apple App Store* has a higher entry barrier to review the quality of apps before releasing them to the public. The *Google Play Store* does not have this barrier, but *Google* developed other strong technologies to detect the

malicious behaviours in apps which generate similar requirement in the app quality. Therefore, barriers are similar in a duopoly. Exclusive contracting is not often seen in the app economy. A possible explanation is because that app price is cheap and the variety of apps in each app market is also huge (Hagiu, 2009a). Therefore, there is no need for exclusive contracting for the app market to maintain a unique competitive advantage. Lin et al. (2011) argued that platform growth is fundamentally driven by continuous innovation that supplies products to the platform market. Therefore, other efficient differentiation method of innovative complements should be researched based on the duopoly context of app markets.

4.1.4 Open Platform Strategies

Another line of discussion in platform strategies is whether to be open or closed (Boudreau, 2008, 2010; Eisenmann et al., 2008), and this discussion is particularly hot in online and open source platforms (Economides and Katsamakos, 2006; Schiff, 2003). Open platform strategies share some components of platform with externalities by giving them more ownership in the participation of the market. In addition to open versus closed, contradictory terms in the strategies are also discussed such as, access vs. control (Boudreau, 2010), and proprietary vs. free (Economides and Katsamakos, 2006).

In contrast to a closed platform that would enjoy the monopoly profits and have a higher degree of control on the whole platform, an open platform shares the profits and ownership with other parties. However, there are still a lot of platforms that have adopted open strategies. I have discussed how the *Android* platform is open sourced in last chapter. *iOS* is not open sourced, but it adopts a lot of open source technologies and makes efforts to be friendly to third party developers, for example, organising the annual developers conference, providing efficient boundary resources and giving comprehensive development tutorials.

The major reasons of open platform strategies are to attract broader participation to expand the market size and also lower the technology barriers to encourage innovation. Those reasons attract platforms to adopt these strategies particularly for the purpose of platform growth and platform competition (Rysman, 2009). History has proved that the total market size can be greater with an open platform (Boudreau, 2010). During a battle for dominance in a two-sided network, some users will delay adoption, fearing that they will be stranded with obsolete investments. Open platform strategies make users feel more ownership of their involvement. Broad participation also increases the chances of innovation relying less on technology but more on freshness, which connects with the opinions from user innovation (von Hippel, 2005) and user entrepreneurs (Shah and Tripsas, 2007) literatures (which will be later discussed later in more detail in Chapter 4.2.5 and 4.2.6).

However, open platform strategies have their own disadvantages. The major disadvantage is the loss of control and dilution of the platform owner's profits (Eisenmann et al., 2008). Eisenmann et al. (2008) argue that with respect to end users and complementors, decisions to open or close a mature platform involve 1) backward compatibility with prior platform generations; 2) securing exclusive rights to certain complements; 3) absorbing complements into the core platform.

Over time, forces tend to push both closed and open platforms towards hybrid governance models characterised by centralised control over platform technology and shared responsibility for serving users (West, 2003). This hybrid model has been adopted by *Apple* for *iOS*, and *Google* adopted an opposite hybrid model for *Android*. *Google* keeps *Android* technology as open source to attract external contribution to improve the infrastructure for all parties, by giving up the opportunity to charge for a manufacturing license fee. *Google* then keeps the *Google Play Store* closed to gain all the profits of app trading.

Open sourced infrastructure attracts a large number of developers and users which stimulates the popularity of the *Google Play Store*. It attracts good developers, but bad developers have also been attracted into the markets. In order to maintain the competitive advantage by improving the average of innovation, the *Google Play Store* has adopted a selection mechanism based on the benefits of open strategies, and it has inspired the signalling selection strategy to be proposed in this thesis.

4.2 Strategic Behaviours

This section reviews the literature on how market players are responsive to the attraction and differentiation of platform strategies (see Table A.10 and A.8). For simplicity, I have focused on two kinds of market players, one is developers who provide products to sell in the market and the other is users who consume products in the markets. In previous chapters, I have also used other terms like, 'producers' or 'app producers' which are equivalent to developers here, and used 'consumers' and 'app consumers' which are equivalent to user here. From this chapters onwards, I am using the names developer and user which are accepted in the world of the app business (Evans et al., 2006; Kouris and Kleer, 2012).

Platform dependency saves developers' cost to reuse certain complex technologies, infrastructures or services to create products. However, this dependency also produces the lock-in effect and increases the opportunity costs for developers to compete and switch (Farrell and Klemperer, 2007). Being different to traditional business, joining a current platform always creates a high degree of competition for other developers. Therefore, the first step is to review the literature on competition.

4.2.1 Competition

So what is competition? Economists present competition as distributing productive resources that create the highest value for users (Durlauf and Blume, 2008), encouraging efficiency (Mankiw, 2006) and innovation (Utterback and Suárez, 1993). A lot of empirical research (Blundell et al., 1999; Carlin et al., 2001; Nickell, 1996) has been carried out to explain the positive relationship between competition and growth. For example, in comparing the firms having either no competitors or more than three, Carlin et al. (2001) found that firms with one to three competitors experienced a faster growth of between 10-12%. After controlling for the number of competitors, the research found that firms exhibiting inelastic demand also grew faster.

However, Carlin et al. (2001) also pointed out that increased competition would discourage innovation in an economy that is mainly composed of firms who are aiming to maximise profit. Higher frequency of innovation would generate more pressure on wages as innovation needs more advanced labour, as well as other costs. Firms having higher debt pressure would also intend to innovate less in order to avoid risk. In order to reduce production costs, firms normally do innovation “step-by-step”. It means that a laggard firm in any industry must first catch up with the technological leader before becoming an innovation leader in the future. Hence, Carlin et al. (2001) claimed that *“more product market competition stimulates innovation and growth, all the more when managers do not respond much to monetary incentives”*.

Competition tends to move resources to where they are most needed, and to where they can be used most efficiently for the economy as a whole. However, this is not always true even in the case that information is perfect (Rothschild and Stiglitz, 1976). For example, competition may lead to waste, duplication and increased costs and prices in some conditions (Aghion and Griffith, 2008). This is part of reasons why business and economic competition is often limited or restricted in most countries. The above arguments tends to agree that a dream market should maintain competition into a certain level, as too little competition can not drive innovation and too much competition will produce wasting of resources. In order to control the negative impact of competition, markets should control the influence of market players who are only interested in maximising profit.

Huyett and Viguerie (2005) predicted extreme competition would become common in their McKinseys report. They identified that three supply-side forces combine to impact on extreme competition: globalisation, technology and economic liberalisation. It requires decision-making to change from a slow, backward looking, and incremental process, into a speedy, flexible, and resilient process. Every corporation would require different degrees of labor, raw materials, a communications infrastructure, production facilities and capital. These factors of production and the way they combine in business systems have changed dramatically over time. Huyett

and Viguerie (2005) identified three broad sequences caused by the change or growth particular in supply chains: “*the aggregation of formerly distinct markets*”, “*enhanced market clearing and efficiency*” and “*greater specialisation*”.

The *Google Play Store* has been developed into such an extreme competition market. The nature of the virtual exchange nature makes the competition borderless in the *Google Play Store*. The ecological market development enriches the public knowledge and lowers down the technological requirements. Low entry boundaries motivates broader developers and lower prices attracts broader users. It has developed the *Google Play Store* into an extreme competition market with a large amount of developers and users. In the *Google Play Store*, a large number of developers and their apps are competing for users for generating profit. Developers have to be really specialists in order to maintain their market position. Competition has significantly developed the innovation, but has also introduced negative impacts. Unlike traditional competitive markets having governance to maintain order, the *Google Play Store* has less governance power due to the adoption of open strategies, but one power *Google* kept is to review apps before releasing them and selecting them for featuring. The *Google Play Store* tries its best to provide as much information as possible to the market to enhance the market selection power of the “invisible hand” (Evans et al., 2006). The next section will review innovation strategies commonly adopted to grow business in competition.

4.2.2 Innovation Strategies

The above discussion and previous section confirms the importance of innovation in order to sustain survival in competition (Lin et al., 2011). This section reviews some commonly used innovation strategies, including differentiation, diversification, the business model and learning, to establish and maintain the competitive advantage of innovativeness.

Differentiation

Differentiation in economics and marketing is to make a product or service more attractive to a particular target audience by distinguishing it from others (Kotler, 2000). It includes differentiating a firm’s own products as well as from a competitor’s products. This is to create value by demonstrating the unique product (Ulaga and Eggert, 2006). In order to improve differentiation, changing a product can be achieved by researching a niche market (Parrish et al., 2006). By describing the differences between products or services, differentiation has the objective to develop a position that potential customers would see as unique.

The freemium business model (Anderson, 2009), in which business makes both free and paid-for versions of a given product, is also a good example for differentiation. Given that they

are targeting on a same group of customers, free and paid-for versions are effectively differentiated on price and maybe also function. Primarily, differentiation can reduce the directness of competition to affect performance: the more difference between products, the more difficulty for categorisation and then the fewer comparisons with its competitors. Rather than primarily competing on price, a successful product differentiation strategy could move your product to competing on non-price factors (such as promotional variables, product characteristics or distribution strategy, etc) (Maurer and Tiwana, 2012).

Diversification

Ansoff (1957) defined four growth strategies:

1. market penetration: The organisation tries to grow using its existing offerings (products and services) in existing markets. In other words, it tries to increase its market share in current market scenario. This involves increasing its market share within existing market segments, by selling more products or services to established customers or by finding new customers within existing markets. Here, the company seeks to increase sales for its present products in its present markets through more aggressive promotion and distribution. It is not really about innovation.
2. market development: A firm tries to expand into new markets using its existing offering. The new market can be different geographies or countries or user segmentation.
3. product development: A company tries to create new products and services targeted to its existing markets to achieve growth.
4. diversification: An organisation tries to grow their market share by introducing new offerings in new markets. It is the riskiest strategy because both product and market development is required.

Diversification is a strategy for a firm to enter into a new market or industry where the business is not currently situated. This could be a strategy for creating a new product in a new market (Wiersema and Bowen, 2008). There is a substantial risk associated with the diversification, as firms have no experience in the new market and do not know if the product is going to be successful. Ansoff (1957) argues that diversification strategy is different from the other three strategies. The first three strategies normally work with the same merchandising, technical, and financial resources used for the original product line. Diversification, however, normally demands a firm to have new facilities, new skills and new techniques. Diversification strategy can be implemented through a variety of methods such as developing a new product internally, acquiring product lines from other firms building alliances with other complementary firms, procuring licenses for new technologies and importing or distributing new product lines

manufactured in other countries. Diversification can be defensive or offensive (Matsusaka, 2001). Defensive diversification happens when current products or the current market tends to lose further growth opportunities, or when firms try to spread the risk of market contraction. Offensive diversification happens when a firm acquires new market positions by using retained cash which may exceed total expansion needs, or it may take opportunities that promise greater profitability than expansion opportunities.

Business Model

Innovation not only lies in products, services and technologies, but also in the business model (Teece, 2010). Due to the popularity of the Internet, revenue is not only determined by purchase actions, but also by other elements such as the attentions it receives from customers. Many new business models have made great successes compared to the traditional economy. *Freemium* (Liu et al., 2014) is one of the most interesting new business models for the purpose of this thesis. Theoretically, differentiation may lead to a price premium, but *freemium* helps users to lower the cost of selection and experiment in the real value of differentiation (Sharp and Dawes, 2001).

Freemium is a combination of “free” and “premium”. In the *freemium* business model, consumers can receive products or services for free, but they have to pay a premium for advanced functionality, features, or related products and services. Consumers may purchase a paid app in the app market directly without trying out its free version. However, a 2013 study (Gordon, 2013) reveals that the majority of consumers in online mobile app stores take advantage of the free apps offering. In the study, many mobile app developers believe that offering a free version of their app is an effective way to attract consumers. Liu et al. (2014) found that the increased sales of the paid-for mobile apps are positively associated with the *freemium* strategy. Positive trial experience as represented by high review rating of the free version of mobile apps leads which higher sales of its paid version.

Learning

Innovation is closely associated with organisational learning, as innovation process involves the dissemination, acquisition and use of new knowledge (Calantone et al., 2002). Organisational interaction and observation of the environment would generally trigger learning. Learning enables organisation to acquire the knowledge and ability to understand and anticipate customer needs, which would increase the chance for organisation to capture the opportunities created by the emerging market demand. Thus, an organisation committed to learning is an indicator of having a greater innovation capability than competitors. Researchers have confirmed that

organisational learning is related to the new knowledge development, which is crucial for firm performance and innovation capability (Hurley and Hult, 1998).

Learning influences the firm to collect certain types of information in a specific context (Dixon, 1992) and then interpret (Argyris and Schön, 1978), evaluate (Sinkula et al., 1997) and share (Moorman and Miner, 1998) this information. Calantone et al. (2002) identifies four components of learning:

- commitment to learning: Commitment to learning indicates a long-term strategic orientation by a firm. They claim that the more an organisation values learning, the more it is likely that learning could occur at higher levels.
- shared vision: Which solves the problem that many creative ideas are never implemented owing to lack of common direction. A clear learning direction is likely to form an organisational strength or even a core competence.
- open-mindedness: Open-mindedness represents how a firm is willing to accept new ideas. Firms must cope with rapidly changing technology and turbulent markets.
- intra-organisational knowledge sharing: Intra-organisational knowledge sharing does not simply refer to obtaining information from various sources. It includes systematic reexamination and structuring of information.

Competitive Advantage

The purpose of innovation strategy is to establish a competitive advantage. Competitive advantage is an attribute that allows a firm to outperform its competitors. Porter (2008) defined the two types of competitive advantage for competition in an industry, one is lower cost, and the other is differentiation. This definition derives from competitive attributes such as superior market position, skills and resources. Porter (2008) suggested that strategic management should build and sustain a firm's competitive advantage (Porter, 1985). He identified four strategies for competitive advantage:

- The **first** strategy is cost leadership strategy, which is to offer products or services at the lowest cost in the market. An example of a company successfully using this strategy is *Walmart*; by targeting the price-aware customer with selected items at low prices while selling other merchandise with less aggressive discounts (Barney, 1995).
- The **second** strategy is differentiation strategy, which is to provide consumers with a variety of services, products or features, that is not provided by its competitors. This is a direct advantage to have a unique product or service that none of its competitors is able to offer. *Dell* is a typical example which launched mass-customisations on computers to fit consumers' needs (Sheffi, 2005).

- The **third** strategy is innovation strategy, which is to disrupt the market by creating completely new or notably better products or services (Christensen et al., 2006). It is common to see this strategy from technology start-up companies, because they normally offer a breakthrough product. A notable example is *Apple*, which disrupted the market with *iPod* personal music players, and *iPad* tablets (Sheffi, 2005).
- The **fourth** strategy is to improve operational effectiveness. The goal of this strategy is to perform internal business activities better than their competitors, making the company easier or more pleasurable to do business with than other market choices, and reducing the production time (Porter, 2008).

4.2.3 Open Source Software Innovation

An app is a software application. Therefore, reviewing software innovation provides a foundation for strategic behaviours in the app economy. Software development is a knowledge-intensive activity. It often needs very high levels of intensive learning, domain knowledge and experience (Fichman and Kemerer, 1997). Like other industries, complex technologies can set up great barriers for both software users and developers to understand and contribute towards software development, which leads to the difficulty of a newcomer's integration in software innovation (Fichman and Kemerer, 1997). Only developers who have been actively involved in the complex technology development can fully understand the software architecture and then effectively contribute to the source code in software development. It may make new contributors feel it is too costly to join the development (Fichman and Kemerer, 1997). In order to be successful in the software development, Voss (1985) outlines the following key factors:

- recognising user needs,
- good communication by the innovator,
- good management of technical aspects,
- presence of a product champion with status,
- availability of adequate resources.

Open Source Software Development

Von Krogh et al. (2003) have studied open source software development by focusing on a newcomer's entry into the community to make their initial contribution to development. Von Krogh et al. (2003) then promoted the theory of open source innovation process in combination with a "joining script", "specialisation", "contribution barriers" and "feature gifts". Von Krogh et al. (2003) proved that newcomers who follow joining scripts are more likely to join in the open source community to contribute. A joiner transits into a newcomer when he is granted

such access to commit on source code. Newcomers could benefit from specialising in their contributions, and their specialisation could also contribute to the feature gifts in a project. In addition, newcomers' prior direct experience could directly impact on featured gifts and contribution barriers, that newcomers create new entry points for other developers to follow.

Open Source Software Motivations

Several studies have explained the different motivations of contribution to open source projects (Rossi, 2006). The most studied motivation for open source software development participation is to scratch a "*personal itch*" on enjoyment, software functionality or willing for membership (Von Krogh et al., 2003). Some people like the open source community because they like the gift culture where the status of a participant depends on "*what he gives away*" (Raymond, 2001). Some existing labor economic theories (Lerner and Tirole, 2005) can also partially explain the motivation for open source software participation. The unpaid programmer working on an open source software development project faces a variety of benefits and costs (Lerner and Tirole, 2005). Meanwhile, many commercial firms increased their investment in open source projects. Thus, more and more developers get paid for their contribution. This adds pay as a incentive and motivation for open source software participation (Roberts et al., 2006). Rossi (2006) tried another way to classify the different motivations to participate, which can be intrinsic motivation or extrinsic motivation. Intrinsic motivation works when innovation activities satisfy basic human needs (like competence, control and autonomy). This makes people feel interested in the activity and likely to perform for its own benefit (Deci and Ryan, 2010). In contrast, extrinsic motivation relates to the task that is external to the environment. Open source studies often assume that motivations in open source communities are complementary or "*mutually reinforcing*" (Von Krogh et al., 2003).

Taking into account the complexity of motivation in open source software studies, Roberts et al. (2006) have studied how different motivations are interrelated. They have focused on how motivations influence participation and performance, and how past performance influences subsequent motivations. They observed that developers' motivations are not independent but rather that they are interrelated in complex ways. Above all, they observe the positive impact of open source software contributions on a developer's status motivations, especially for those who are motivated to enhance status or explore new career opportunities via open source software development. But the negative aspect is that open source software contribution is consuming user-value motivations for people who are motivated to develop for personal use. Secondly, they found that different motivations have different influences on open source software participation. They perceived that use-value motivations lead to below-average contribution levels, while paid participation and status motivations lead to above-average contribution levels. They also

observed that developers' contribution levels are positively influenced by their performance rankings. Based on the above observations, they suggested that developers' subsequent status motivations could be enhanced by past-performance rankings.

Innovations emerge in private-collective models (von Hippel and von Krogh, 2003), with one end consisting of purely private interests and the other end by purely collective interests. In the former case, private financial interests like patents, licenses and so forth operate as motivation to protect the innovators own investment. In the latter case, the innovator expects the collective benefits such as learning opportunities (Himanen, 2001), social prestige (Lee et al., 2003) and reputation (Lerner and Tirole, 2002). Different to the closed-source innovation model, open source innovators are not too concerned with free-riding problems, because their personal gains are higher than the free-riders (von Hippel and von Krogh, 2003). Non-contributors or free-riders may have a larger learning curve, and also be less likely to earn a social recognition status. Having a learning curve is key to the participant's future ability, as it helps firms to identify potential spin-off opportunities and develop other profitable products, like additional consulting or maintenance services. It is a trend that many open source software developers start to develop various modules and then take the possibility of spinning off from ad-hoc innovations. This then allows them to profit privately from their proprietary knowledge gained from open source software development (von Krogh and von Hippel, 2006).

Open Source Software Competition

Bessen (2006) attempted to study the competition between open source software and commercially developed software. Open source software is often labelled with the well-known under-supply and free-ridership problems. He argued that a commercial firm always selects to focus on the largest user groups. Users who cannot solve their needs using primary software which may be pushed into developing and debugging their own software, and this always forms the initial steps of open source software. Contrary to Bessen (2006), Baldwin et al. (2006) tried an alternative method. They proposed that the modularity and option values inherent in a software architecture can influence a developer's incentives to increase the supply effort and then reduce free-riding in a collective process. In other words, software that are more modular or have more option values, could increase developers' incentives to join and remain involved in an open source development effort, and then decrease the amount of free-riding in equilibrium. These outcomes emerge because modularity and option value create opportunities for the exchange of valuable work among developers.

Open Source Software Commercialisation

It is worth mentioning that “open source software” is not equivalent to “free software”. “Free software” generally means that individuals or organizations can not make money from the software. On the contrary, open source becomes more and more profitable. [Fitzgerald \(2006\)](#) even named the recent open source model as *Open Source Software (OSS) 2.0* which has a strong commercial orientation. He pointed out that more corporate firms are considering attaining the best competitive advantage from open source. For example, *Red Hat* has disclosed its plans to shift from the software stack to middleware and management tools. Some other proprietary companies have also seen open source’s strategic potential to enhance their competitive ability in their industry and to grow market share or undermine competition. For example, *IBM* spends the majority of its resources on supporting *Linux*, because it influences the profitability of the operating system market and adversely affects competitors like *Sun* and *Microsoft*. Many companies start to pay developers while they are assigned to work on open source products. These companies receive a high reputation while involved in open source innovation and attract other talent-demonstrating developers to join them even for their in-house development. After surveying 500 of Australia’s top firms, [Goode \(2005\)](#) found that managers rejected open source software because they could not grasp its relevance to their operation. They perceived a lack of reliable ongoing technical support as a result of implementing open source software and also appeared to see substantial learning costs. Therefore, they adopted other software that they believed to be incompatible with open source software. This condition has been significantly changed due to the broader participation and rise of commercial open source software ([Hauge et al., 2010](#)), and it will be verified when we will discuss the app market in next chapter.

4.2.4 Open Innovation

[Chesbrough and Crowther \(2006\)](#) identified a problem within companies where “*not all good ideas will come from inside the organisation and not all good ideas created within the organisation can be successfully marketed internally*”. Therefore, a more open model is demanded, particularly open innovation. [West and Gallagher \(2006\)](#) defined open innovation as “*systematically encouraging and exploring a wide range of internal and external sources for innovation opportunities, consciously integrating that exploration with firm capabilities and resources, and broadly exploiting those opportunities through multiple channels*”. This means that the open innovation paradigm is not just utilising external sources of innovation such as rivals, customers and universities, but is ever changing in the management, use and employment of intellectual property (IP) ([West and Gallagher, 2006](#)).

In the past research, three major processes have been identified for open innovation (Enkel et al., 2009).

- One process is the “*outside-in process*”, which integrates knowledge from customers, suppliers and external resources to expand a company’s own knowledge base, and thus improve a company’s innovation (Laursen and Salter, 2006; Lettl et al., 2006; Piller and Walcher, 2006).
- The second process is the “*inside-out process*” which refers to the profits earned by selling a company’s IP, bringing ideas to market and transferring ideas to the outside environment. By this process, a firm focuses more on externalising its knowledge and innovation in order to implement ideas faster than they could via internal development (Chesbrough, 2007a; Enkel and Gassmann, 2010; Vanhaverbeke and Van De Vrande, 2008).
- The third process is referred to as the “*coupled process*”, where a company creates in conjunction with complementary partners via cooperation, alliances and joint ventures (Enkel et al., 2009). A company may jointly develop and commercialise innovation by combining the “*outside-in*” and “*inside-out*” process (Enkel et al., 2009).

Open source software innovation is a clear example of the open innovation concept (Von Krogh et al., 2003). Open source software innovation allows developers to obtain private rewards from writing code for their own use, sharing their code and collectively contributing to the development and improvement of the software (von Hippel and von Krogh, 2003). The paradigm is that firms spend money on R&D efforts in developing open source software, but those efforts are also available to rival firms. This means that the investments made by the firms involved in an open-source software often shared in the return with real and potential rivals. As a result, open innovation has been adopted by many big players (Chesbrough, 2003b), either to maximize returns to internal innovation, or to incorporate external innovations, or to motivate spillovers.

Gassmann (2006) summarized a list of factors responsible in the drive for open innovation in open source software development, including:

- globalisation (Anderson and Tushman, 1990),
- technology intensity (Miotti and Sachwald, 2003),
- technology fusion (Kodama, 1993),
- new business models (Bonner and Walker, 2004),
- knowledge leveraging (Nonaka, 1994).

West and Gallagher (2006) identified four strategies often employed for open innovation in open source software:

- “*pooled R&D or product development*”,
- “*spin-outs*”,
- “*selling complements*”,
- “*attracting donated complements*”.

In the software industry, flexibility, agility, and concentration on core competencies are considered as sources of competitive advantage. The “*do-it-yourself*” mentality in technology and R&D management is outdated. With open innovation, open-source software would enhance a firm’s competitive advantage (West and Gallagher, 2006). Therefore, academic communities continually emphasise that firms should open boundaries to outside innovation (Boudreau, 2010; Chesbrough, 2010; Gassmann, 2006; West, 2003).

Huizingh (2011) summarises the context characteristics of open innovation into two groups, internal and external. Internal characteristics often refer to “*company characteristics related to demographics and strategies*”. Examples of demographic characteristics include employee age, location, sales, profits, market share and ownership type. Examples of strategy characteristics include incumbents versus new entrants, organisational culture, aspects or goals of the innovation strategy as well as other purposeful acts that could be related to open innovation performance. Harison and Koski (2010) found that employee characteristics also matters, where a company using open source software is likely to have highly educated employees. Huizingh (2011) discussed that company size is particularly important. Small companies that often have limited resources and market share can benefit greatly from open innovation. However, limited resources also prevent them from creating and enforcing intellectual property rights on the products they developed. In reality, smaller companies practice open innovation activities extensively, which creates a trend in business development (van de Vrande et al., 2009). Huizingh (2011) also claims that industry is the most obvious external characteristic. Nearly all studies in open innovation focus on specific industries. Although they observed a trend towards open innovation across industries, Poot et al. (2009) found that this trend is composed of shocks rather than continuous evolution, and that the timing of open innovation differs between industries. Applying open innovation seems to be more a matter of business strategy than a matter of industry trends. Keupp and Gassmann (2009) suggest that, for explaining open innovation adoption, the internal environment in firms is more important than the external environment.

In some cases, open innovation requires government to initiate the engine. Companies change their behaviour to make innovation publicly available. The demand for open data is generated through government collection and sharing of data. Private companies may also collect data and share as an “open” asset to the public. Data is a necessary input to innovation, and open data bridges the gap by allowing multiple parties to collaborate through a reduction in

the unequal distribution of benefit costs. A large portion (43%) of respondents in a large-scale survey ([Lakomaa and Kallberg, 2013](#)) claimed that access to public sector information open data would support and strengthen their business plans. Results further showed that open data in the public sector was essential in the innovation of web-based applications and information services. Some respondents reported that the absence of open data slows the innovation process and prevents the initiation of entrepreneurial innovation.

4.2.5 User Innovation

Industry firms (producers or manufacturers) obtain inputs from suppliers to develop and produce goods and or services, and then sell to buyers within the traditional value system. These buyers can be intermediate users, who can and will use the products as inputs for their own production processes, or end-consumer users, who would in turn use the products to satisfy their personal needs. The role that these users play during the innovation of the products that they ordinarily buy from producers has been the subject of research for an extended time ([Bogers et al., 2010](#)). An early research stream demonstrated that users play an important but still peripheral role by providing producers with critical inputs to more readily meet customers' needs ([Adams et al., 2006](#)). Subsequent research would go a step further by arguing that users can be the sources of innovation, and they can be innovators as well, and not just assist in the innovation process ([von Hippel, 1988](#)). Following von Hippel's 1986 groundbreaking work on lead users, the importance of them as a source of innovation has been widely recognised by both industry and academics ([Gassmann, 2006](#)). Most recently, however, the research on users as innovators has extended into areas as diverse as industry dynamics, entrepreneurship, firm boundaries, innovation communities, measurement, and even policy ([Bogers et al., 2010](#)). Users have begun to create their own desired products to complement the market's ecosystem with toolkits that were more user friendly ([von Hippel and Katz, 2002](#)). Integration of customers into the early phases of the innovation process has also been analysed regarding roles ([Brockhoff, 2003](#)), chances ([von Hippel, 2005](#)) and risks ([Enkel et al., 2005](#)).

Similar to open source and open innovation, motivation of users' free contribution is an interesting aspect. [Bogers et al. \(2010\)](#) recently reviewed "why do users innovate". The major reason is due to "*stick information*". Information stickiness is defined as "*the incremental expenditure required to transfer that unit of information to a specified locus in a form useably by a given information seeker*" ([von Hippel, 1994](#)). It is dependent on the amount of knowledge that must be transferred in order for innovation to occur. Innovation is the combination of technical information with a certain level of knowledge about what a user would specifically need, and translates that combination into a new product that users can see themselves actively using. In traditional innovation models, the required information is generated on the user side,

while the solution information lives with the producer. Information stickiness represents the ability and value for users to obtain and utilities that technical information. It emphasises more on the main focus of innovation rather than on how producers would search for information about others within the market itself; users are more likely to perform innovative activities if information about user needs are sticky compared to the technical solution information (Ogawa, 1998). More generally, based on unique knowledge and expertise, users effectively create low-cost innovative solutions tailored to solve their specific needs. Moreover, users' expertise and experience in using a product determines their ability to innovate given that expert users in a given product field have correspondingly lower knowledge costs and are more likely to innovate. von Hippel (2005) continued to argue that a user's decision to innovate is determined by the agency costs resulting from the misconstrued interests between the user (as a principal) and the producer (as an agent). Like the open source software community, innovative users have a first priority to satisfy their needs from their innovation process if it is not available from the producers. Especially for lead users, whose needs are demanded by the mainstream users months or even years later, who will benefit significantly from solving these needs early on, and may even draw profit from selling their innovation or develop a career path from these innovative processes (Bogers et al., 2010).

In some cases, fitting to various user needs is a paradox, mainly as producers are dreaming to develop a product that can supplement the broader users rather than many products for various user needs. Producers may also look to lower their development costs by incorporating solutions that have already been established or that can be used for other (potential) users (Bogers et al., 2010). User innovation helps other users to better adopt the producers' product to meet their needs, but leaves producers with freedom to focus on their main products. Research has also identified that user innovation can influence the producer to modify his innovation process (Spital, 1979), or how the products are being implemented (von Hippel and Tyre, 1995). By involving lead users, producers can ensure that entail the information collections about both user needs and solutions are not only from users at the leading edges of the target market but also from users within other markets that have faced similar problems in a more extreme form (Lilien et al., 2002). It helps the developers to create breakthrough products that have higher performance and marketplace potential than other innovation strategies (Lüthje, 2004). Also, encouraging user innovation would influence the relationship between producers and customers from purely the buy-selling relationship into co-creation collaboration. This would actually lead users to be even more active participants in product development and implementation (Buur and Matthews, 2008).

Compared to "Off-line" user innovation studies (Franke and Shah, 2003; Shah, 2000), online business is especially interested in user innovation studies (Lerner and Tirole, 2002;

O'Mahony, 2003). Online communication has dramatically lowered the costs of firm-to-user and user-to-user interaction, and compared an "off-line" and physically based community (Jeppesen and Frederiksen, 2006). The low-cost access enables users innovation with a more flexible production process, and also enables some firms to quickly respond to new information, such as user feedback, resulting in a better-performing product and developing project cycle (MacCormack et al., 2001). More firms have begun to employ communities of users to strengthen their innovative process, and so freely reveal innovations to a firm's product platform by putting the firm in a more favourable position (Jeppesen and Frederiksen, 2006). This is because 1) those new product features have become available to all users through sharing on a user-to-user basis, 2) has allowed the firm to pick up the innovations and integrate them in future products and then benefit by selling them to all potential clients. Such innovative users are likely to be "lead user" (von Hippel, 2005), and their characteristics can positively affect an innovator's willingness to share their innovations, or responsiveness to "firm recognition" as a motivating factor for undertaking innovation (Jeppesen and Frederiksen, 2006).

4.2.6 User Entrepreneurs

An interesting thing about user innovation is that users become entrepreneurs during innovation (Shah and Tripsas, 2007). Shah and Tripsas (2007) defined user entrepreneurship as *"the commercialisation of new product and/or service by an individual or group of individuals who are also users of that product and/or service"*. Shah and Tripsas (2007) distinguish user entrepreneurs between professional-users and end-users. Professional-users entrepreneurs are embedded in an organisation and employ a product in their professional life. They experience a need for improvement and leave their firm in order to develop and commercialise a solution. End users entrepreneurs, in contrast, are individuals who use a product in their day-to-day lives. A large amount of evidences supports the existence of user entrepreneurs. For example, professional user entrepreneurs in typesetters (Tripsas, 2008), then ice harvesting industry (Utterback, 2004), and more recently in probe microscopy (Mody, 2006), windsurfing, skateboarding, snowboarding (Shah, 2003), rodeo kayaking (Baldwin et al., 2006), automobile (Franz, 2006), and stereo components industries (Langlois and Robertson, 1992).

Classic entrepreneurship researchers argues that opportunities exist because of inefficiencies in the allocation of resources in the economy (Morris et al., 2005). By combining resources in a novel way, entrepreneurs can form new means-ends relationships (Kirzner and Discovery, 1997), thereby introducing a previously unseen or unknown good, service, production process, or raw materials to the commercial marketplace and creating the potential for economic profit (Shane and Venkataraman, 2000b). User entrepreneurship process is distinct from the classic entrepreneurship process in two primary ways (Shah and Tripsas, 2007). First,

a product innovated for personal use from user entrepreneurship may not have any formal acknowledgment or evaluation of a commercial opportunity. Second, user communities play a significant role in the development and diffusion of the innovation. Contrasting to classic entrepreneurs, user entrepreneurs would often obtain feedback and adapt prior to firm formation.

[Shah and Tripsas \(2007\)](#) also explained how user innovators transit into user entrepreneurs. Before the concept of an entrepreneurial venture even existed, a user would experience a problem or need first. After searching the market for available solutions, the user determined that the need could not be met by products or services available in the market. So the user developed a prototype product for personal use. Once the innovator begins to use the product/service in public, others observe it, often providing feedback and sometimes expressing an interest in adopting this product or service. In fact, some user innovators received unsolicited requests from others wishing to purchase their innovations, enabling them to assess willingness to pay. This expression of interest frequently sparked the idea of funding a firm. This is a common scenario of transition. [Shah and Tripsas \(2007\)](#) hypothesis that user entrepreneurship is more likely to dominate classic sources of entrepreneurship when use provides enjoyment, as opposed to pure economic benefit, or when users have relatively low opportunity costs, or when industry is characterised by small scale, peripheral, niche markets with high variety in demand, or when the product is new and when high levels of uncertainty and ambiguity about user needs exist, and when those needs are evolving.

As previously explained, community is important to open source software ([Von Krogh et al., 2003](#)), open innovation ([West and Lakhani, 2008](#)), and now also for user entrepreneurs ([Shah and Tripsas, 2007](#)). Loosely-affiliated members provide social structures and tools to facilitate interactions among users. Members of user communities engage in a variety of activities ranging from socialising with others who have common interests to teaching an activity to new members to share knowledge about how to use a product better to encourage the creation and diffusion of useful innovations. A community generally identifies with the group tends to motivate cooperation and to share ideas and resources ([Hertel et al., 2003](#)). To further reinforce and support cooperative behaviour, many user communities develop norms and rules, methods for attracting and socialising new members, and techniques for maintaining their structure and integrity over time. User innovators who participate in a user community can obtain first-hand information regarding the needs and preferences of potential adopters. Higher levels of novelty can emerge due to collective creativity because of the platform community. Members of a community often embraces a beneficial innovation introduced by another member, thereby validating its commercial potential - often before the user innovator had even thought of commercialising the product.

4.2.7 App Innovation

[Boudreau \(2012\)](#) reviewed several important properties of software development that he believed could influence software innovation. The dataset he studied went from 1999 to 2004, a period dominated by PDAs, which is most similar to the app market. Most purchasing transactions are via third party retailer website, so that it does not require a PDA platform owner to review and control the app quality prior to release. Nowadays, the app market has been significantly evolved. Before releasing to users, developers have to submit to a platform owner like *Google* or *Apple* to review their developed *Android* or *iOS* apps respectively. The platform owner is not only the technical architecture designer, but also the market operator. In the current model, users would buy, install and review apps directly via platform, rather than third party retailer. The market size is also significantly broader, and it has a much large number of developers and apps in the current market.

[Boudreau \(2012\)](#) emphasises the effect by adding more producers. However, my interesting is to also understand how developers can strategically survive in such a highly competitive two-sided market by signalling to receivers who are other top players in the market. Developers want to be recognised to being innovative. A platform would encourage innovative developers to join, and users would enjoy innovative apps. Therefore, innovation is an important quality each party in the market wants to see. The app innovation process is a good fit to understand how developers are signalling to receivers about innovation. Therefore, the author is attempting to review several important properties of the app innovation process by specifically examining the mobile app market.

Network Effects

The network effect on app innovation is indirect ([Lim and Bentley, 2012a](#)). Innovation would firstly drive the expansion of the installed user base on a platform. The Internet business has already proved that user base expansion would liberate the boundaries to business models at certain levels, for example, how the freemium business model rapidly emerged ([Anderson, 2009](#)). This would then increase the expectation of the business success rate and average revenue of apps on the platform, and finally encourage more developers and more innovation. We can see the network effect on innovation is indirect via the user base. The platform literature thus claims that accumulating a “critical mass” of users and developers is important for a platform strategy ([Evans, 2009](#); [Evans and Schmalensee, 2010](#); [Suarez, 2004](#)). A bandwagon effect would then follow with escalating network effects ([Swann, 2002](#)).

However, the network effects can also perform a negative impact. Innovation expands not only the variety, but also the user expectation ([Chia et al., 2012](#)). Continued innovation in one

domain would make the innovation space become narrower and the cost of innovation would be increased. The increase in cost would discourage continued innovation. Even more, some developers may simply copy innovative products. The copied app may generate less profit, but save the cost on R&D, so the revenue can still remain high. The cost and copy effects prevent positive development of network effect.

The network effect is especially important to small and medium enterprises (SME). Having limited R&D resources, SMEs have to cooperate along the value chain to have success with technological challenges and promising expensive and risky production innovation. In order to perform well in a networked environment, a recent study ([Rese and Baier, 2011](#)) listed out the factors that would influence new product performance. It demonstrated that new product performance in inter-firm networks is positively related to traditional success factors, like product advantage (in the eyes of the customer), proficiency of technological activities, proficiency of marketing activities, protocol (quality of product and project definition), market potential and (quality of) project team organisation. They also put emphasis on that the new product performance in inter-firm networks that is positively related to network related success factors like trust, commitment, dependency and compatibility. They argue that network stability and the dimensions related to social exchange are likely to bring sustainable advantages in terms of innovation and economic cost.

Scale and Scope

[Boudreau \(2012\)](#) claims that although a software compiler might develop within a finite set of instructions, the possibilities of innovative generation are virtually boundless. [Caves \(2003\)](#) refers to this as the “infinite variety property” of a network. This possibility of a virtually infinite variety is clearly consistent with the network effect characterisation.

Fast expansion of *Android* indicates the infinite possibilities. Opened in 2008, *Android* achieved 2.8% of the worldwide smartphone market share in Q2 of 2009 (second quarter) and then grown to 33% in Q4 of 2010 and then 52.5% in Q3 of 2011. By Q3 of 2012, *IDC* reported 75% of market belonged to *Android* ([O'Donnell et al., 2013](#)). Tracked data (see Figure [A.15](#)) showed that the *Google Play Store* has 2,300 *Android* apps available in March 2009, which increased to 16,000 in December 2009, and expanded to over 1,250,000 in July 2014. The figure is still expanding in scale.

The expansive novelty also happened in scope. The first commercial *Android* device called *T-Mobile G1* has a keyboard which was released in October 2008. Nearly all later *Android* phones are touch-screen based. Nowadays, *Android* operating system and apps are also running

on tablet¹, televisions (*Android TV*²), cars (*Android Auto*³), glass⁴ and wrist watches (*Android Wear*⁵). *Android*'s source code is released by *Google* under open source licenses. Nowadays, *Android* apps can even run on competitor's platform, such as *Blackberry* devices⁶ and possibly on *Windows* devices⁷.

The expansion also happened to cover more users. Due to the open nature of *Android*, a number of third-party app markets also exists for *Android*. They are invented to provide a substitute for devices that are not allowed to ship with the *Google Play Store*, provide applications that can not be offered on the *Google Play Store* due to policy violations, or for other reasons. Examples of these third-party stores have included the *Amazon Appstore*⁸, *GetJar*⁹ and *SlideMe*¹⁰. In 2014, *Google* launched *Android One*¹¹, which mainly targets customers in the developing world. Actually, manufactures are producing various *Android* device from high price range to low price ranges.

In an effort to lower the boundary for open and user innovation, the community has developed several convenient tools to achieve its target. *Google App Inventor*¹² is an open-source web application that was originally developed by *Google*, and is now maintained by *MIT*. It allows non-programmers to create *Android* apps by using a graphic interface, which is very similar to *Scratch* (Resnick et al., 2009) and is popularly used in primary schools to drag-and-drop visual objects when creating software. In 2014, *Google* unveiled an *Apache Cordova* based framework for porting *Chrome HTML5* applications directly wrapped in an *Android* native app shell¹³. Similar tools (like *Phone Gaps*¹⁴, *Appcelerator*¹⁵, and etc) are continually being invented, while other back-end services (like *MBaaS* (Gonçalves and Ballon, 2011)) are increasing in popularity, and other add-on tools (like tracking, analytics, etc) are now available. These peripheral players (Kuk, 2006) significantly expand the innovation ecosystem.

This expansion is not only a result of the dynamic innovation ecosystem of *Android*, but is also due to the blurring gap between them and their competitors, like *iOS* by *Apple*. In order to

¹<http://www.android.com/tablets/>

²<http://www.android.com/tv/>

³<http://www.android.com/auto/>

⁴<https://www.google.co.uk/intl/en/glass/start/>

⁵<http://www.android.com/wear/>

⁶https://developer.blackberry.com/playbook/android/files/webinars/BlackBerry_Runtime_for_Android_Apps.pdf

⁷<http://www.theverge.com/2015/4/29/8511439/microsoft-windows-10-android-ios-apps-bridges>

⁸<http://www.amazon.co.uk/appstore>

⁹<http://www.getjar.com/>

¹⁰<http://slideme.org/>

¹¹<http://www.android.com/one/>

¹²<http://appinventor.mit.edu/appinventor-sources>

¹³<http://blog.chromium.org/2014/01/run-chrome-apps-on-mobile-using-apache.html>

¹⁴<http://phonegap.com/>

¹⁵<http://www.appcelerator.com/>

win the platform war, they must win a substantial share of the market by offering revolutionary functionality (Eisenmann et al., 2011; Zhu and Iansiti, 2012). It not only initiates platform learning, but also cross-platform learning. For example, *iOS* developed in-app billing with the aim of achieving great success, a few months later *Android* had this same functionality¹⁶. Similarly, *Android* implemented the multi-task function which was soon picked up by *iOS*¹⁷. Meanwhile, due to network effects and switching costs in platform markets, entrants normally have to offer revolutionary functionality in order to win substantial market share, according to the platform envelopment strategy (Eisenmann et al., 2011). Developers enter into multiple platform to leverage shared user relationship (Eisenmann et al., 2011). In 2010, *Mashable* reported that 1,412 developers have published apps for both *iOS* and *Android* which equates to 15% for *Android* developers and 3% for *iOS* developers, with the percentage continuing to increase¹⁸. Cross-platform tools like *AppMethod*¹⁹, services like *MBaaS*, and technologies like *HTML5*, simply reduce the boundaries between platforms.

Recombination and Reuse

Recombination characters are spreading all over the development of computing technology (Evans et al., 2006). Computer systems are constructed by stacks style which are laid up by necessary components. More complex products are combinations of even more components. Frakes and Kang (2005) define software reuse as “the use of existing software or software knowledge to construct new software”. Computer scientists have developed all kinds of concepts and technologies to support efficient recombination or reuse. For example, encapsulation or the black-box concept to minimise the dependency between components (Evans et al., 2006; Kiczales, 1996), API (Application Protocol Interface) that is popular for service providers to encourage mashups (Yu et al., 2008) and middleware library that can be used by software developers to distribute compiled functionality to other developers.

API is particularly interesting and popular in the app economy and works as a communication protocol between the service provider and the service consumer. The most widespread type of API is HTTP based, which is called *RESTful* (Fielding, 2000). Developers gain access to API via various tools for the computing power or data resource behind the API which is controlled by the service provider. This way, developers are enabled to reuse the same computing resources in different presentations, and it also encourages peripheral innovation because of the limitation where the service owner cannot fully utilise all the power of resources.

¹⁶<http://www.adweek.com/socialtimes/google-vs-apple-android-in-app-billing-vs-ios-in-app-purchasing-focus-on-developers-vs-customers/13124>

¹⁷<http://www.extremetech.com/computing/112013-how-multitasking-works-on-android-and-ios>

¹⁸<http://mashable.com/2010/07/02/ios-android-developer-stats/>

¹⁹<http://www.appmethod.com/>

This liberation has proved to be extremely true in the last few years due to the popularity of cloud computing and big data trend. Nearly all the big Internet players (like *Google*, *Facebook*, *Apple*, *Microsoft*, *IBM*, etc) are providing their API to the public.

The purpose of this type of opening is to create competition for developers to win in the *API Economy* (Gat and Succi, 2013). API exposes certain aspects of their business, but it allows other developers to disseminate their products through a combination of value-added software and services. Additionally, API allows more companies to compete on equal terms with well-established businesses. This action assists service providers in growing their user base and consequently increases the market size. To achieve this goal, API providers must make every effort to improve their services and create a more user friendly API by reducing the technical difficulties. Subsequently, this drives another trend, called mashup, to enhance the ecosystem around API (Yu et al., 2008). Previous studies (Linares-Vásquez et al., 2013) confirmed that API's instability (change-proneness) and fault-proneness may impact on the success of software applications. They empirically tested and concluded that APIs used by successful apps are significantly less fault-prone than APIs used by unsuccessful apps. In addition, APIs used by successful apps are also significantly less change-prone than APIs used by unsuccessful apps, including when changes affected method signatures and especially public methods.

Innovation by recombination and reuse is sequential and cumulative (Parker et al., 2010), because of the special nature of innovation in the software industry. Yoo et al. (2010) claim that digital artefacts can be flexibly programmed and re-programmed. This nature allows for an unending digital canvas, providing nuanced continuity and a vast expanse for creation. Therefore, apart from an infinite expanse with many "distant" designs, the nuanced multi-dimensional product space affords ample scope for incremental but meaningful "tweaking", reuse and recombination to generate palpably novel offerings. Growing compatibility among digital goods and services creates more scope for the recombination of a range of digital programs, subroutines, services, features and content, perhaps when mapping functions are added to advertising on a social networking application.

It is worth clarifying that recombination requires that developers have certain skills they understand and can put in practice. Previous studies (Chin et al., 2011; Felt et al., 2011b; Hanna et al., 2013) have confirmed and identified that the copying and pasting of vulnerable code exist in the Android market. Some apps are reported by illegitimate authors to repackage and re-brand an already paid or popular app with additional program functionality in order to generate revenue and even execute malicious code²⁰. The community is collectively preventing negative recombination. One solution is to encourage expert manual review for the apps. The

²⁰<http://www.theguardian.com/technology/blog/2011/mar/17/android-market-pirated-games-concerns>

second solution is to encourage users to report on any observed negative behaviours, and meanwhile some automatic detection technologies are developed. Mojica et al. (2012) were the first to report on the volume of reuse in *Android* apps. They found that almost 50% of studied apps are inherited from developer's previous app, and 61% of them are reusing codes from more than two apps. The same study was recently extended in with more than 200k apps (Mojica et al., 2014), and 84% of the classes are reused across all the categories of apps. More and more technology has been developed to automatically detect reuse (Hanna et al., 2013; McDonnell et al., 2013; Minelli and Lanza, 2013; Syer et al., 2011).

Uncertainty and Skewed Outcomes

Innovation is an unpredictable activity, often accompanying uncertainty (Katila and Chen, 2008). Uncertainty makes many "experiments" to fail. In software design, two sources of uncertainty commonly exist (MacCormack and Verganti, 2003). One is platform uncertainty which "*reflects the amount of new design work which must be undertaken in a project*", and the other is market uncertainty which "*reflects the uncertainty faced in determining the requirements that the resulting product must fulfil*". This view on uncertainty advanced the iterative process by emphasising learning and adoption if compared to sequential process and emphasising model which is largely conducted in relatively stable industries (Bhattacharya et al., 1998; Krishnan et al., 1997). They emphasised the importance of investments in architectural design, early technical feedback and early market feedback in order to face the great uncertainty.

Developers have to pay close attention to platform uncertainty, especially in the apps market. Developers' apps rely on the support from platforms. Therefore, if platforms change their technology specification, it would lead to incompatibility or crashing of exiting apps under the new specification. In order to stabilise the market, platforms would try their best to design their original products successfully to avoid changes being made later on (MacCormack and Verganti, 2003). Even though both *Android* and *iOS* have only existed for less than a decade, hardware technology such as better microprocessors and new sensors have been introduced, more efficient software is developed, and users' expectations are future increased. Those factors required platforms to update but, in turn, will introduce new uncertainty. For example, by looking at the history of *Android* ²¹, whose latest operating system is *Android* 5.0, each version would have a separate SDK and developers are required to adopt each SDK into order to release apps for that particular version. By upgrading to the new versions, developers may have had to change the way they implemented certain functionalities due to the platform changes on dispatch or even to modify certain API. Developers may also want to learn the new features

²¹<http://www.android.com/history/>

in new *Android* versions in order to be the first to move on new functionality. As *Android* is expanding ²², this uncertainty will never disappear.

When *Android* updates its system, the market may have devices running on multiple different versions of the *Android* system, and this phenomenon is called *Android* fragmentation (Ham and Park, 2011; Han et al., 2012a). *Android* fragmentation includes hardware-based fragmentation (caused by different processors, sensors or screen sizes ²³) and software-based fragmentation (caused by parallel deployed versions or vendors customised versions (Han et al., 2012a)). The fragmentation of the *Android* ecosystem causes portability and compatibility issues within the entire *Android* platform, which increases developer workload, delays application deployment and ultimately disappoints users, which creates market uncertainty. Even though the whole community is making efforts to reduce the level of fragmentation, developers still have to decide how to arrange resources to develop and maintain for specific versions by fitting this uncertainty into their business strategy and focus.

The cause of this fragmentation is not made in one day ²⁴. *Google* cannot innovate all features of *Android* technology, and it can't even guarantee that their innovations receive positive result. In response, *Google* open sourced the complete *Android* system to the community. This openness greatly increased the innovation scale and scope, allowing *Google* to focus on its core innovation path to control the uncertainty of the coming market. The great flexibility on customisation makes *Android* alliances collectively develop the market size and successfully incubate a few alliances (like *Samsung*, *LG*, *Motorola*, *Xiaomi*, etc, which produced 18,796 distinct *Android* devices in 2014²⁵). These companies are strong enough to compete with *Android*'s rival which is *iOS*. However, it decreases the control of *Google* compared to the closed environment of the *iOS* platform. It actually creates many constraints for developers to deal with fragmentation. *Google* also realised this issue and they adopted two significant steps to fight on fragmentation ²⁶. The first step is to separate *Google*'s proprietary services from the *Android* system, so that users would get *Google*'s service update independent from vendor's customization. This allows developers to have less dependence from *Android* devices. The second step is to reduce the updating rate of *Android* system version, which is to allow developers to have less uncertainty on the *Android* system. Both strategies cannot completely remove the fragmentation, but at least it can slow down to give developers more time to catch up.

²²<http://source.android.com/compatibility/index.html>

²³<http://www.tech-thoughts.net/2012/03/analysis-of-android-fragmentation.html>

²⁴<http://www.businessinsider.com/android-fragmentation-2014-8>

²⁵<http://opensignal.com/reports/2014/android-fragmentation/>

²⁶<http://thenextweb.com/google/2013/08/22/googles-focus-on-in-house-apps-is-making-the-android-fragmentation-problem-irrelevant/>

Consonant with this sentiment, [Leesig \(2014\)](#) argues that thousands of experiments may only lead to only few innovations. With skewed outcomes, sometimes, innovation is popularly considered as a process to explore the “killer app”. [Caves \(2003\)](#) use “nobody knows” to describe similar character of innovation in creative industries. Moreover, [Anderson et al. \(2007\)](#) believes that uncertainty is not the only reason to generate skewed outcome. Skewed outcomes may also happen if a large number of product offered to meet heterogeneous and dispersed niche interests ([Petsas et al., 2013](#)).

Low-Cost, Facilitated Development

Building applications on top of platforms can reduce development costs, speed product creation, reduce coordination costs and facilitate experimentation ([Baldwin and Woodard, 2009](#)). This platform approach requires an initial nontrivial fixed investment but then provides a base for innovation inherently geared toward handling large numbers of variants, supporting various experiments, and providing sufficient flexibility and economies of scope and scale ([Yoo et al., 2010](#)).

A huge effort was made initially to create the platform base²⁷. The huge initial effort makes later development cheaper by lowering technology development boundaries, encouraging developers to enter and increasing the variety of business models. Since 2008, *Android* has seen numerous updates which have incrementally improved the operating system, added new features and fixed bugs in previous releases. Even though those updates sowed the seeds of fragmentation, they quickly improved the the *Android* system and reduce efforts on the app developer’s side ([Taudes et al., 2000](#)). The community provides free tools (like IDE²⁸, API²⁹) for app developers, technical documentation³⁰, detailed tutorials³¹ and conferences³². The comprehensive design of *Android* systems contributes to the growth of the *Android* ecosystem ([MacCormack and Verganti, 2003](#)), and the growth of *Android* ecosystem enriches the variety of tools to improve productivity. Many middleware services are emerging to support app development in *Android* ecosystem³³, for example, user tracking service, mobile data storage services, source coding online management, etc. Meanwhile, by increasing the market size, more people are influenced to learn about app development technology. This increased the available force of app developers, and has significantly lowered the operation cost of

²⁷<http://www.bloomberg.com/bw/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal>

²⁸<https://developer.android.com/sdk/index.html>

²⁹<https://developer.android.com/google/index.html>

³⁰<https://developer.android.com/guide/index.html>

³¹<https://developer.android.com/training/index.html>

³²<http://androidweekly.net/conferences>

³³<https://developers.google.com/games/services/android/quickstart>

development companies. More and more companies have joined to recruit app developers in their operation³⁴. The increase on market size has also stimulated the variety of business models. Recent business model, like freemium and others, have appeared by reducing the boundaries to earn revenue directly from users. Many companies can even provide services by making profiting simply from user-contributed data (Turber et al., 2014).

The low cost has not only happened on the platform owner and app developers side, but also on the users side. The platform war pushed *Google* to continually improve the reliability of the *Android* system to increase the coverage to broader users. *Android One* is such an example to run latest version of the *Android* system but has selected to enhance only the most valuable features for developing country users. The high competition pushes app developers to lower the purchasing price, improve the productivity and place more emphasis on the value of products. The average price for a paid app in *Android* is only a few dollars³⁵, and a large number of paid-for apps cost less than one dollar^{36 37}. In-app purchase technology enables users to only pay for what they desire. This low cost factor has significantly reduced the switching cost for users between apps, and low switching costs encourage developers to focus more on core value development in their apps.

Boundary

The low requirement for technical skills and easier to use development tools has reduced the boundaries for entry to both newbie and professional developers to enter. The low boundary enables ordinary people or even users to transform their idea into a product by using apps. This phenomenon has been termed as “open innovation”, has been popularly studied (von Hippel, 2005) and has the effect of attracting many would-be application software producers (MacCormack et al., 2006). Thus, app platforms generally allow many sorts of individuals to join in the development (Boudreau, 2012). von Hippel (2005) emphasises the importance of the platform role for potentially “democratising” digital innovation base.

App development has become much easier so far. For example, a very basic application for the *Apple iPhone* might only require a couple of weeks of part time development work³⁸, and a survey concluded that the average time to build a mobile app is 18 weeks³⁹. There have been a number of cases reported where novices have created apps using convenient tools such

³⁴<https://nationalcareersservice.direct.gov.uk/advice/planning/jobprofiles/Pages/softwaredeveloper.aspx>

³⁵<http://www.wsj.com/articles/SB10001424127887323511804578300183961009070>

³⁶<http://blogs.wsj.com/digits/2013/03/11/the-surprising-numbers-behind-apps/>

³⁷<http://www.appbrain.com/stats/free-and-paid-android-applications>

³⁸<http://www.popsci.com/diy/article/2009-02/how-make-iphone-app>

³⁹<http://www.kinvey.com/blog/2086/how-long-does-it-take-to-build-a-mobile-app>

as *Google App Inventors* ⁴⁰. One instance was where a student created a 'frequent message informing app' by simply putting three graphic code blocks together. Another example is of the nursing student who created an app to enable phones to send an emergency message or to make a call if someone fell or was taken ill.

The ease of entry does not, however, guarantee the financial success of apps. From a market perspective, easy entry would push the competition into a higher level. This would increase the difficulty of user selection, and damage (especially high quality) developers' benefits to generate a crowding out effect (Boudreau, 2012). A 2014 report from *Gartner* confirmed that 90% of paid apps are downloaded less than 500 times per day, and earn less than \$1250 a day. *"This is only going to get worse in the future when there will be even greater competition, especially in successful market."* ⁴¹. *Gartner* predicted that by 2017, 94.5% of downloads are for free apps. The competition is also rising from the development of other web technologies, like *HTML5*, by having more and more products migrating from other technologies. Therefore, the strategy on how to survive such a highly competitive business environment is worth investing in.

4.3 Signalling Theory

The third part of literature review is on the signalling theory. Including individuals or organisations from two parties means one would have different information to access. The sender party must decide whether and how to communicate or signal that information, and the receiver party must decide how to interpret the signal (Connelly et al., 2011). This is a common problem caused by information asymmetry and also a common solution for effective communication and selection. Since Spence (1973) proposed the signalling theory in his 1973 jobs market paper, signalling theory has received great attention in both academy and industry (Bliege Bird and Smith, 2005). Due to the great impact on various fields, Spence won the Nobel Economics Prize in 2001 with George A. Akerlof and Joseph E. Stiglitz for their contribution on signalling theory ⁴².

The problem identified in this thesis (see Section 3.5) is caused by information asymmetry. *Google* tries to select distinguished developers to promote. Users try to select good apps to purchase. Good developers try to select other good developers to collectively develop the market. However, making those selection decisions requires effective communication and transparency on quality. Meanwhile, the networked nature of the ecosystem leads to the fact that the selection decision is influenced by interaction of multiple parties, which makes it even

⁴⁰<http://www.nytimes.com/2010/07/12/technology/12google.html>

⁴¹<http://www.forbes.com/sites/connieguglielmo/2014/01/13/mobile-apps-may-not-pave-the-way-to-developer-riches-sales-average-less-than-1250-a-day/>

⁴²<http://www.riksbank.se/en/The-Riksbank/Economics-prize/>

more complex for the overall research difficulty of this thesis. Fundamentally, the signalling theory aims to reduce information asymmetry between sender and receiver (Spence, 2002). Although it is not widely applied in the app economy, signalling theory is believed to provide foundation to model the selection mechanism and to explain the impacts of signalling in this thesis. Therefore, the rest of this chapter reviews the signalling theory to provide a clear research background.

4.3.1 Information Asymmetry

The phrase “bounded rationality” (Simon, 1955) expresses the idea that human decision-making is limited by available information, available time and the mind’s information-processing ability. Stiglitz (2002) also claimed that “*information affects decision making in every context - not just inside firms and households.*” Information can be accessed in public being freely available to everyone, or accessed in private, being limited to certain people to access at certain levels. For the same freely available information, people have different abilities in acquiring the data let alone different people having amount of information. Even for the same specific information with the same access levels, different people have different abilities to absorb information and also extract the knowledge out from the information in different ways. Even for the same people and with the same knowledge, but in different contexts, people still make different decisions. All kinds of ability differences lead to the different response for decision-making.

People having too little information make them less likely to gather true knowledge and make the perfect decision. People having too much information face “information overload” issue. “*Information overload occurs when the amount of input to a system exceeds its processing capacity. Decision makers have fairly limited cognitive processing capacity. Consequently, when information overload occurs, it is likely that a reduction in decision quality will occur*” (Speier et al., 1999). The overload of information has increased people’s cost to rank and select the valuable information to process. For example, in the *Google Play Store*, so many new apps are created and so many updates are generated in a single day that it requires from *Google* an extreme capacity to review all of them. If the capacity does not exist, *Google* has to decide to keep and process only selected data and also face the risk of selection with less control on app quality.

Stiglitz (2002) believed that information asymmetry occurs when “*different people know different things*”. To be more specific, when one party has more or better information than the other. This can be due to the fact that some information is private or people have different acquiring or absorbing abilities, or that the context of environment has changed. This generates an imbalance of power in transactions that creates disorder in what should be a perfect competition. Even more, parties having imbalance of information asymmetry would have different

interest and motivation in their decision making context. This leads to the question of how perfect is the information. Perfect information theoretically happens when an agent has all relevant information to make a decision. In contrast, imperfect information is a situation where parties have different information in transition, for example, when a used car seller has more information about the car's quality than the buyer (Akerlof, 1970). Sellers often have better information about a product than buyers because they are more familiar with it. They know more about its quality, durability and other features. A buyer, in contrast, has limited contact with the commodity and thus has less information. Another good example is the insurance market (Healy and Palepu, 2001). Insurance applicants often have more information about the risks they are going to insure.

Existing literatures demonstrates many information asymmetry cases. One example happened between bondholders and stockholders (Miller and Rock, 1985). Bondholders who do not benefit from higher profits typically use a risk-averse strategy. On the other hand, stockholders have an interest in taking more risk. If a risky project succeeds, all the profits go to the shareholders. While if the projects fail, the bondholder has the risk to share in case of failure in the project (although the bondholder has a higher priority than a shareholder for repayment in cases like bankruptcy). Thus, bondholders often have to set up costly and large ex-ante contracts to prevent the management from taking on very risky projects, otherwise bondholders simply raise the interest rate to increase the capital cost of a project.

The situation of shareholders and the CEO (Hendry, 2002) is another example of information asymmetry, generally considered to be a classic principal-agent problem (Hart and D, 1983). The agent (the CEO) is working on behalf of the principal (the shareholders). The CEO, if he only had a small stake in ownership, would have interests differing from firm's owners. Instead of making the company more efficient and profitable, the CEO may be tempted into following: 1) empire-building (Baker and Kiymaz, 2011) (like increasing the size of the corporation, rather than the size of its profits, which usually increases the executives' prestige, perks, compensation, etc., but at the expense of the efficiency and thus value of the firm), 2) not firing subordinates whose mediocrity or even incompetence may be outweighed by their value as friends and colleagues (Bebchuk and Fried, 2005), 3) retaining large amounts of cash that, while wasteful, gives independence from capital markets (Lins et al., 2010), and 4) a maximum of compensation with a minimum of "strings" - in the form of pressure to perform (Bebchuk and Fried, 2005). Agency costs arise because of contracting costs and the divergence of control, separation of ownership and control and also due to the different objectives (rather than shareholder maximisation).

Information asymmetry within societies can be created and maintained in several ways. Firstly, media outlets may fail to disseminate certain viewpoints or engage in propaganda

campaigns due to their ownership structure or political influences. Furthermore, information imbalances between the poor and the affluent can be generated by an educational system that relies on substantial tuition fees. Certain organizational and legal measures can also enhance imbalances, such as document classification procedures or non-disclosure clauses. Exclusive information networks that are operational around the world further contribute to the asymmetry. Lastly, mass surveillance helps the political and industrial leaders to amass large volumes of information, which is usually not shared with the rest of society (Lightfoot and Wisniewski, 2014).

The rise of information computing technology has been used to reduce the gap of information between parties. For example, blogging is playing a more central role in reducing the effects of information asymmetry between corporate insiders and insider trading (Saxton and Anker, 2013). Logging on financial websites such as *Seeking Alpha* and *The Kirk Report* provide bottom up communication from investors, analysts, journalists and academics. Financial blogs prevent people in power from withholding financial information from the public. Saxton and Anker (2013) find that more participation on blogging sites from credible individuals reduces information asymmetry between corporate insiders as well as insider trading. Compared to traditional forms of media such as newspapers and magazines, blogging provides an easy venue to provide information to the public.

By fitting into the *Google Play Store* context, information asymmetry has made the selection decision difficult. It is not easy for each party to observe other parties, because the information to other parties is private. For example, developers do not disclose their real business model to the public to reveal their intention, as the business model is a company secret. It is also difficult for ordinary users to review each line of an app's source code to detect hidden malicious behaviour, as it is too technical for users to understand. *Google* has difficulty in predicting a developer's future performance once they are selected. There are many other imbalances of information in the app economy, and high competition in the app economy leads to information overload which, in return, increases the levels of information asymmetry. To overcome this information asymmetry issue, *Google* also makes efforts to solve the issues by disclosing as much information as possible. For example, developers can show themselves in the *Google Play Store*, with privacy policy, company website and other released apps. Each app keeps track of the download history, user feedback, updating and other similar apps to help the user make a good decision.

Adverse Selection

Having asymmetric information can lead to an adverse selection by pushing customers to select low quality products or services. Adverse selection is originally identified in insurance

([Johnson, 1978](#)). It describes a situation where an individual's demand for insurance is positively correlated with the individual's risk of loss (higher risks buy more insurance), and the insurer is unable to allow for this correlation in the price of insurance. This may be caused by the fact that only one individual knows his private information, or that regulations prevent the insurer from using certain categories of known information to set prices (for example, the insurer may be prohibited from using such information as genetic test results, gender, ethnic origin or pre-existing medical conditions). In this case, the principal is not informed about a certain characteristic of the agent in adverse selection models. For example, health insurance is more likely to be purchased by people who are more likely to get sick.

The principal-agent problem or agency dilemma arises where the two parties have different interests and asymmetric information, especially when the agent has more information. For example, the principal cannot directly ensure that the agent is always acting in the principal's best interests, particularly when activities that are useful to the principal but are costly to the agent. This leads to a conflict of interest by an agency cost. Agency cost ([Jensen and Meckling, 1976](#)) is an economic concept concerning the cost to a principal when he chooses or hires an agent to act on his behalf. Because the two parties have different interests and the agent has more information, the principal cannot directly ensure that its agent is always acting in the principal's best interests. Agency cost consists of two main sources. The first cost is inherently generated by using an agent. Specifically, the principal risks that his agents may use organisational resources for their own benefit. The second cost is generated by monitoring an agent, like gathering more information on what the agent is doing (for example, producing financial statements) or employing mechanisms to align the interests of the agent with those of the principal (like compensating executives with equity payment such as stock options).

Adverse selection may also result in a moral hazard ([Holmstrom, 1979](#)) occurring when the risk-taking party in a transaction knows more about his intentions than the party who is paying the consequences of the risk. More specifically, when the party with more information about its actions or intentions has tendency or incentive to behave inappropriately from the perspective of the party with less information, moral hazard may occur. It happens because the principal usually cannot completely monitor the agent. The agent may have an incentive to act inappropriately from principal's viewpoint if the agent and the principal have different interests to align. For example, if a person purchases insurance to protect from automobile theft, this person may be less cautious about locking his car because the negative consequences of vehicle theft are now (partially) being shared with the insurance company.

George Akerlof's "The Market for Lemons" paper ([1970](#)) is considered as a classic paper on adverse selection which introduced information into economic theory. It influenced two primary solutions to the adverse selection problem, including signalling and screening. [Spence \(1973\)](#)

originally proposed the idea of signalling. He proposed that people may signal in information asymmetry in order to transfer information to the other party and reduce the information asymmetry. [Stiglitz \(1975\)](#) developed the screening theory, and the theory proposed that the party who are under-informed can influence other parties to reveal their information. Screening helps people to make choices depending on other party's private information. [Akerlof \(1970\)](#) already notes that, in such a market with adverse selection, the average commodity value tends to go down because information asymmetry enables “*unscrupulous*” sellers to “*spoof*” items and defraud the buyer. It would thus influence many people to prevent risks by avoiding certain types of purchases, or by not spending as much for a given item. Thus, the market can decay.

In the *Google Play Store* context, “bad” developers tend to “spoof” their low quality apps and defraud the users. Innovation is costly, and “bad” developers may just copy or even add malicious features into the apps to disorder the market. Users' attentions are all on popular product domain, and it narrows the success rates of developers to innovate in a niche domain. As a result, developers may not be willing to innovate since their efforts on innovation may not secure the returns. The development of adverse selection may gradually lead to a conflict of interest, agent cost and hazard model, which would damage the ecosystem of a networked market. In such massive market, signalling is believed to be an action which can be adopted to overcome those problems.

4.3.2 Signalling Theory

In economics, signalling ([Connelly et al., 2011](#)) means one party (the agent) credibly transmits some information about himself to another party (the principal). It was developed as a solution to solve adverse selection and is applied to understand the selection behaviour in many cases.

Market of Lemons

The signalling theory was initially diluted from adverse selection problem and [Akerlof \(1970\)](#)'s study on “Lemons” is the commonly recognised paper to firstly formalise it. This paper explained three key assumptions: 1) goods on the market are of different quality, 2) sellers are better informed about the quality of their goods than buyers, 3) sellers have a reservation price for their goods. This reservation price generates an adverse selection equilibrium. Particularly, [Akerlof \(1970\)](#) shows that a market can have no trade in equilibrium even when: 1) every seller wants to sell at a positive price, and 2) there are buyers willing to pay the seller's reservation price for goods. The reason for no trade is that sellers hold reservation prices above the price buyers are willing to pay for the available goods at that price. Because the owner of an inferior goods is always willing to sell for the price of superior goods, there is an adverse selection of

the goods for sale at a given price. In the extreme cases, there is no equilibrium price where the value of goods on the market at a given price is equal to that price.

Jobs Market

Spence (1973)'s paper on the "jobs market" studied on a labour market with no adverse selection. Akerlof (1970) believed that perverse equilibrium would be produced because of seller reservation prices and that reservation prices result into no trade under adverse selection. But Spence (1973) noticed that even without a reservation price, a perverse outcome would still be possible. The approach is for the owners of superior goods to take observable actions that less-able workers cannot do. Good quality workers often choose to do it, and wages are paid differently for this observed characteristic. Spence (1973) concluded that these actions can occur in equilibrium, even if they are costly and unproductive.

Education

Lang and Kropp (1986) and (Bedard, 2001) find some evidence of signalling on education. Lang and Kropp (1986) look at college attendance as a function of school leaving laws which primarily directly affect secondary school attendance. Bedard (2001) looks at high school dropout rates as a function of the cost of attending college. Both studies found that the "indirectly affected groups" behave as the signalling model would predict: students attend college to "separate" from the high school grads (Lang and Kropp, 1986) and more students drop out of high school because the quality of the high school graduate pool is diluted by greater college attendance and so its not worth the cost of education to pool with them (Bedard, 2001).

Crowdfunding

Signalling effects lead to learning effects. Kim and Viswanathan (2013)'s study on app crowd funding indicate that the crowd indeed learns from the investments made by early investors. However, not all early investors are equally influential. The study found that the crowd is more likely to follow people having more knowledge like *App Developer Investors* and *Experienced Investors*, although each look for a different category of apps that match their expertise. The crowd is more likely to follow *App Developer Investors* for concept apps, and *Experienced Investors* for live apps. The two categories of experts - *App Developer Investors* and *Experienced Investors* - are likely to differ in their expertise, the former with a better knowledge of the product and latter with a better knowledge of market performance. They

demonstrated that, although inexperienced, the crowd is rather sophisticated in its ability to identify and exploit nuanced differences between different signals within the same market.

IPO (Initial Public Offering)

Ahlers et al. (2015) empirically examines the effectiveness of different quality attributes taken from offering documents used by entrepreneurs to induce small investors to commit financial resources in an equity crowd funding context. They found that financial road maps, such as acquisition exit strategies or a pre-planned IPO, are generally less important, because audience regards them more as “cheap talk” than as effective signals. Meanwhile, venture risk, including government grants or patents, had little or no significant impact on funding success. For example, top executives would often signal to capital markets that diversification strategies are in the owners’ best interests, when they increase ownership stakes in their firms (Goranova et al., 2007). Leaders of a young firm in an IPO always stack their board with a diverse group of prestigious directors to send a message to potential investors about the firm’s legitimacy (Certo et al., 2001; Filatotchev and Bishop, 2002).

Structure of Signalling

The above examples demonstrated how a sender undertakes costly actions to signal its underlying quality to receiver. Connelly et al. (2011) summarised that: “*Signalling theory is useful for describing behaviour when two parties (individuals or organisations) have access to different information. Typically, one party, the sender, must choose whether and how to communicate (or signal) that information, and the other party, the receiver, must choose how to interpret the signal*”. The large number of developers and users in the *Google Play Store* drives the difficulty of selection. Therefore, it is important for developers to make actions to signal their quality in order to success in their business.

Connelly et al. (2011) reviewed the primary elements of the signaling theory in the form of a timeline. The timeline includes sender, receiver, as well as the signal itself. The purpose of the signal is to visualise the sender’s unobserved quality, to fulfil observation needs or demands by outside receiver. Signallers are often insiders, who obtain information about a product (Kirmani and Rao, 2000), individual (Spence, 1973) or organisation (Ross, 1977), and information is not available to outsiders. Insiders can obtain both positive and negative internal private information, and they must decide whether to communicate this information to outsiders. The reality is that insiders generally do not send negative signals to outsiders, and the signalling theory focuses mainly on actions insiders take to intentionally communicate the insider’s imperceptible and positive qualities. Therefore, the quality is unobserved, but the

signal should be observable to receiver. In order to differentiate the quality of sender, the signal has a cost to produce, and the opportunity cost to produce the signal is smaller for a sender having real quality but higher for a sender having low quality. This discourages low quality senders to produce less signals. Receivers are outsiders who lack but demand information about the organisation, and would like to receive this information. In reality, both the sender and receiver may have partially conflicting interests, such as successful deceit which would benefit the signaller at the expense of the receiver. For signalling to take place, the signaller should benefit from some action by the receiver. This benefit is normally associated to the selection of signal sender.

In the context of the *Google Play Store*, senders are developers, receivers are multiple parties including users, *Google* and other developers. The great benefit of selection by receivers pushes developers to signal their quality. Those qualities of the developers and their apps are difficult to observe, however, the signal can be observed via a channel on the *Google Play Store*. Those signals are costly to produce, and it is cheaper for good developers to produce the signal. So the question to study is on what signals developers can have to communicate their good quality on innovation?

4.4 Summary of Literatures

More and more scholars have agreed to the concept of "platform thinking". Existing platform strategies literature has discussed on how to attract internal and external complements in order to generate network effects dynamically cross different parties in a two-sided market. The common methods include financial strategy by providing financial subsidising or tying with boundary resources, and exclusivity strategy, which is to uniquely contract important players in the markets. This attraction makes an important contribution to the emergence and growth of the platform.

As the platform has grown, the market size has been greatly expanded by attracting a large amount of participation. The platform attracts both "good" and "bad" complements. "Bad" complements waste platform resources and causes damage to the user experience and platform's brand value. In order to maintain the competitive advantage, the platform has to differentiate the complements and encourage to have more innovative complements. Existing methods to differentiate the quality of complements includes setting up entry barriers and exclusive contracting.

However, those methods are inefficient in an extremely high competitive duopoly context having *Android* and *iOS* combining to take over 90% of the total market share. Both of them have similar participation popularity. They also set up similar price barriers and has evolved in

similar path via platform envelopment. The low cost nature of apps and huge economy scale in the markets disable *Android* and *iOS* from adopting exclusive contracting strategies. The unique nature of app innovation demands a more efficient management strategy by fitting into the current duopoly context of the app economy.

In other aspect, the studies on developers show the behaviour patterns in innovation, which is called strategic behaviour in this thesis. In order to establish competitive advantage, some common strategic behaviour are aiming to create innovative products (like differentiation and diversification), and some are aiming to improve the knowledge (like learning) and process (like business model). The development of innovation strategies has encouraged me to study the strategic behaviour in open source software innovation, open innovation and user entrepreneurs. Those studies have provided a foundation to study the strategic behaviour in app innovation.

Information asymmetry causes the difficulties in differentiating the quality of complements. However, strategic behaviour, is the same in innovation activities. Therefore, I have been inspired to apply the signalling theory in order to select innovative complements based on developers' strategic behaviour in innovation, which forms the signalling selection strategy. The next chapter will describe the design of the signalling selection strategy.

Chapter 5

Model Development

I have described and summarised the problem of this study in previous chapters (see Section 3.5). To solve this problem, this chapter develops a model based on the signalling theory to understand a developer' strategic behaviour on 1) how they strategically signal their quality to be distinguished from competitors for business survival, and 2) how signalling selection would impact on the platform management in the app economy. The contribution of the developed model is to provide an efficient platform strategy in a highly competitive duopoly context of the app economy. It provides management knowledge about developers' innovation motivation and their signalling behaviour by specifically fitting this to an app economy context. It could also help to study the lessons on how the market operators could use the selection mechanism to strategically manage innovation in an ecosystem.

5.1 Model Observation

5.1.1 Openness Develops Crowdedness

The open strategy has developed the crowded nature of the *Google Play Store*. *Google* open sourced *Android* for open collaboration, and attracted a large number of manufacturers to develop devices for a variety of users. *Google* also makes efforts to develop easy-to-use tools, lowering entry barriers, contributes towards technology improvement and shares development knowledge within the *Android* developers' community. This open culture is developing the *Android* community into an innovative ecosystem. Within the ecosystem, open innovation is encouraged to generate a variety of business models, and user innovation is encouraged with lower technology boundaries. Both developers and users are attracted into the market, and the quick expansion has pushed the *Google Play Store* to become a crowded market in just a few years.

The *Google Play Store* is a two-sided market. Developers display their apps on *Google Play Store*, and users choose apps from it. Developers are paying the same cost to enter into the market, that is only \$30 as a one off payment for an unlimited display. However, the displaying order of their apps is different to users in each search context. A user may not be likely to see apps which are displayed towards the very low end of the order, for example users may only see the first few pages of app lists. Apps displayed in high order would be likely to be seen by more users and to trigger more purchase action (see Figure A.16). *Google* decides upon the app orders to display to users. This displaying order cannot be purchased by paying a fixed price, and *Google* has tried to ensure fairness of opportunity to developers to demonstrate their qualities. For example, *Google* would display at the front newly released apps and newly updated apps (see first row in Figure A.16). *Google* would also display at the front the most popular apps and selected apps (see third row in Figure A.16).

5.1.2 Developers Compete For Featuring Selection Opportunity

Developers compete to be displayed at the front. In first case, developers can frequently generate new apps, or update existing apps, and then their apps are automatically displayed at the front (Barrera et al., 2012; Möller et al., 2012). This strategy is not very sustainable due to the crowded nature of the market. Each day there is a huge number of newly released or updated apps, so this order is changed very quickly. This causes each app in this order to have only a short amount of time to stay in the front. Meanwhile, frequently generating and updating may cost a lot of resources for developers, or lowers the guaranteed on the quality of apps.

A better strategy is to compete for accumulative recognition with innovation quality. Developers can innovate to improve the quality of apps and make efforts to add more value to users via their apps (Basole and Karla, 2012). Users would recognise the apps quality and accumulatively increase their popularity. This competitive advantage is less likely to be beaten nor disappeared. Even more, *Google* may recognise the quality by observing the feedback from multiple agents, and then choosing to permanently reward the developers. This reward works as a certificate to the developer, and always helps developers to stand out from the crowd.

5.1.3 Information Asymmetry Increases The Selection Difficulties

Selection is not, however, easy to make in such a crowded environment. Information asymmetry commonly exists between developers, users and *Google*. Only developers themselves know how good they are, and how much value their apps should be worth of. However, in order to compete for selection, developers try their best to claim that their apps have the best qualities (Chia et al., 2012). This inequality of information between developers and others causes the

difficulty for selection. For example, *Google* has difficulty in technically examining the details for each app, as there are so many apps generated and updated everyday. Users have difficulty to search for best apps fitting their desires. Other developers may also have difficulties to filter out good apps that they should learn from. Another issue is on inconsistency, as developers can change their quality over time. So others have to frequently observe the market to ensure consistent selection.

5.2 Signalling Selection Model

To solve this problem, I developed the model to understand how developers would signal their ability to be selected in the app economy. Developers have to strategically make certain actions to convert their unobservable qualities into signals and send them out to others (Connelly et al., 2011). The first path is for a developer to signal their quality to *Google*. Once *Google* observes the signal and chooses to feature the developer's quality, users will adopt the selection as a signal for guaranteed quality by *Google*, and will be motivated to purchase the apps. The second path is for developers to signal their quality to other good developers. Good developers have a higher ability to understand the advanced technology and potential market insights compared to other developers. Therefore, the peer recognition for a particular developer can be a signal to both users and *Google* to help them understand how good the developer is. The two paths are interconnected, because selection is based on the observation of each other. The visualisation of my model is represented in Figure A.5. In this model, the signaller is the developer, and the receiver includes other developers, users and *Google*. However, due to the setting of a two-sided market, it is assumed that users contact developers via *Google*. Therefore, the model is simplified to narrow down to the condition that receivers only include other developers and *Google*. The quality for signalling is innovation ability, and the signal reflects on how innovative the developer is. Other good developers respond to a good developer's signal to collectively develop the market, and this collection can also be an indirect sign for *Google* to determine the developer's innovation quality. Users would look at *Google*'s selection and at other developers' responses to decide their own response to the developer.

5.2.1 What Constitutes A Signal?

In the Literature Review chapter (see Section 4.2.2), I have discussed the literature on innovation strategy. Innovation is to add additional value to the products to meet user demand. Therefore, in order to survive in such a competitive market, a key quality for developers to have is innovativeness. However, innovation is not cheap and not easy to make. In order for receivers

to observe and identify it, developers make costly actions to signal their innovation abilities, and those actions are less likely to be made by developers having no such quality.

A simple path to innovation could mean to differentiate the app from other apps. Apps can be different from existing apps, with a different outlook or user interaction flows. Similar to a web page, an app can have different colour schemes or different layouts for its components. Not all differences make positive contributions for user and market recognition. For example, only differences that simplify user interaction or increase the aesthetic feeling, may be appreciated by users. The *Google Play Store* is crowded and users have very low switching costs between apps. Therefore, simple differentiation may not significantly impact on platform selection results. Developers should look at exploitation which is to improve an app based on previously existing apps. For example, *MobiSystem*¹ has been working on a mobile English dictionary app, and then they create more dictionary apps for other languages.

An advanced path to innovation is diversification. Rather than being closely similar to other previously existing apps, developers may invent a completely new app making the app to solve problems never or seldom targeted by other existing apps. This completely new invention may have the potential to disrupt the market. For example, *Angry Birds*, is a games app based on shooting animals and is considered to be the “*largest mobile app success the world has seen so far*”². Developers can also explore a new product domain they never explored before to expand the number of users they had on their previous apps. This could bring new opportunities for developers to explore an existing successful domain they have never tried before. The exploration can be a complementary to an existing one, or even an additional product domain. For example, one of *Microsoft*’s flagship products is *Office* and it mainly runs on the *Windows Operating System*, but now they have now extended *Office* into the mobile phone market to provide a seamless user experience³. In another example, *Dropbox*’s original product was aimed at file sharing⁴ and now they have also entered into the email domain with new apps⁵.

Innovation is costly. Developers have to fully understand the existing market in order to innovate (Katila and Chen, 2008). For example, they may have to carry out research to discover out problems not explored by existing apps and then figure out gaps to improve upon. This research may need to rely on market observation over a long period of time. Making R&D research requires advanced staff with skills in technology and marketing, so the human resource cost can be high (Jung et al., 2013). Developers devoting so many resources on R&D may have returns inferior to their expectations. It is likely that three new apps are only slowly being

¹<https://play.google.com/store/apps/dev?id=4895393381068725503>

²<http://miller.mit.edu/articleangry-birds-will-be-bigger-mickey-mouse-and-mario-there-success-formula-apps/>

³<https://play.google.com/store/apps/details?id=com.microsoft.office.officehub>

⁴<https://play.google.com/store/apps/details?id=com.dropbox.android>

⁵<https://play.google.com/store/apps/details?id=com.mailboxapp>

accepted by users, or are not popular at all among them (Santos and Eisenhardt, 2009). A major risk of innovation is to make mistakes. By not being aware of what other developers are doing, a developer's innovation efforts may be a duplicate of others' existing or ongoing work. They could even duplicate mistakes already made by other developers (Katila and Chen, 2008).

Therefore, it is important for developers to learn from others, including other developers, users and *Google*. Other good developers already devote their innovation efforts by observing uncertain product domains, and by learning from them would avoid repeating mistakes other developers have already made (Katila and Chen, 2008). Users would also feedback information for improvements to aggregately reflect user demands (Felt et al., 2011b), and learning from them could reduce uncertainty on the demanded product (Fu et al., 2013). User innovation is also a source to assist the developer in achieving and enhancing the innovation (Remneland-Wikhamn et al., 2011). *Google* expresses to the public how they expect developers should innovate via selection. Catering to *Google*'s expectations would increase the chances for developers and their apps to be selected. Put simply, innovation efforts should be made by being aware of others' work in the market.

Innovation influences on how much the product range can be extended. How quickly having the next new app would affect on how quickly the product range can be expanded. In other words, how quickly producers can follow the latest products to generate a new product would decide the rate of growth for product variety. A mobile app's playing time can be short due to its low switching cost. Especially on gaming apps, since many of them are free or very low cost, users may wish to try new apps after a short period of playing the previous app. Quicker following from developers leads to more products and it helps users to meet their needs for easy switching. Platforms also wish to see a quicker following, as it increases the frequency in trading and increases the total revenue. Slow following causes impatient users to leave the market if they find desired products in other competitor markets. This would damage the advantages of both the producer and platform, a result that would be very undesired.

Another aspect is the length of time required to develop a quality app. To create an app, a developer would spend time on research, development and testing. There needs to be the right amount of time to allow the app to be developed to ensure top quality. A rushed job may not ensure consistent quality; too slow a production would mean that it receives the best quality but followers might not pick this app in such a crowded and highly competitive market. An innovative idea may be implemented early by competitors. This can also be explained by the innovation path on exploring and exploiting (March, 1991). Both activities need time to be made, however, excessively slow actions allow other competitors to catch up, so the early effort is negatively influenced (Katila and Chen, 2008). Therefore, developers have to balance the following time to ensure it is not too rushed and also not too delayed. For platform owners like

Google to achieve this goal, it takes either having developers to rapidly follow up with new apps, or attracting new developers to work on new versions.

In summary, developers can innovate by exploitation, exploring or disruption. The higher return of innovation would be accompanied with higher input and risk. Developers need to invest in a lot of resources for innovation, including length development times, advanced staff, careful market insight observation and the ability to analyse effectively. However, the investment is not always a guarantee for return. Learning is an efficient way to avoid risk, and also to reduce the input of resources by reusing the knowledge of others. Therefore, the developer needs to be aware of other good developers and follow them. Although it is costly, this would enable good developers to be better able to make positive innovations, which means in the long run that they are cheaper to do than they would be for developers who have little or no innovative ability. So it is possible to assume that some actions can only be carried out by developers with high quality, and those costly actions constitute the signal sent by developers for receivers.

5.2.2 How Are Signals Observed?

Signals can be observed and absorbed differently by different receivers (Connelly et al., 2011). Innovation requires technological skills, market research skills and business prediction. Therefore, low ability developers may not be able to understand the signals from high quality innovating developers. It is possible that only other good developers can figure out the truth of long time investment return for an innovation. This pushes the signalling in a specific direction rather than targeting on everyone in the *Google Play Store*. In other words, good developers are more aware of other good developers having similar or better abilities rather than of all kinds of developers. This question can be seen in Figure A.6.

This inequality is the same for users. The opinion of leaders would have a large impact on a user's decision. Lead users may have demand to investigate into technical details to test the quality of apps. That is why *Google* and *Apple* like to invite top developers to hack their products^{6 7}. Some users may identify and ask questions after intensively using the apps, and those questions may attract attentions from other ordinary users (see Figure A.17). Developers signal their ability via replying to those lead users⁸.

The observation of signals will call for collective efforts. When developers innovate, especially by opening up a new market domain, it takes a lot of effort to increase the growth of

⁶<http://www.forbes.com/sites/andygreenberg/2013/01/28/google-offers-3-14159-million-in-total-rewards-for-chrome-os-hacking-contest/>

⁷<http://www.zdnet.com/article/mac-hacking-competition-winner-mocks-apple-security/>

⁸<https://support.google.com/googleplay/android-developer/answer/138230?hl=en>

the market domain into a large size. Large size markets would attract more developers to join them and provide a variety of apps for this new domain. For some apps that heavily rely on the user community, like Facebook, more users mean better user experience. A signal from good developers could increase the confidence for users to join in the new domain. Therefore, a collective effort between developers and users would develop a new domain into a larger size and all of them would benefit from it (Hyrnsalmi et al., 2012b).

The benefit of collective efforts is verified by looking at good developers as it reduces the risk of uncertainty. Each developer has a number of users who love using their apps and subscribe to their updates. Therefore, good developers joining in a new domain bring their subscribed users into the market, which then signals to good developers who can secure the opportunity for market success and to claim the new market (Santos and Eisenhardt, 2009).

Collaboration introduces crowding, but good developers choose the right time to follow new developers to balance out competition. The time of entry can influence the successfulness in the market (Zhu and Iansiti, 2012). An early entry helps to gain an initial advantage, but faces great uncertainty about the market growth. However, the early mover also spends more resources to incubate and train their users. Late entry takes the benefit out of large users who have already joined and accepted the market, but the competition also needs to be much higher at this stage. Therefore, knowing the right time to enter into the market is a knowledge that not all developers possess. Developers who have more resources and understand the business better are more likely to find the right time for entering the market.

Google observes the signals and collective efforts to decide whether to select developers and to promote them to users. The objective of *Google* is to increase the market size by attracting both developers and users. *Google* has an interest in ensuring the market has sustainable growth, thus, good developers and lead users can gain priority attention in the crowded market. Both of them are more interested in innovation, which in turns secures the competitive ability of the market to competing with other markets such as *iOS*. It may be difficult to filter out all good developers, but it is possible for platforms to filter out relevant new platforms. Hence, the strategy would be to select relevant good developers and let them compete and learn how to be selected.

5.2.3 Summary of Model

In summary, the *Google Play Store* is an extremely competitive two-sided market based on the *Android* platform. It is a virtual market place, which means both developers and users have little to no geographic restrictions. The market uses been using open strategy, which has attracted a large number of developers and users and pushes competition to the highest levels. This crowding effect brings difficulties for market governance and product selection by

enhancing information asymmetry. In order to survive in such extreme competition, innovation is the key for sustainable success that means including developers to compete with developers, the market to compete with other market and users to compete for good developers to solve their problem in a more innovative way.

Google adopts a selection strategy to manage the sustainable development of the *Google Play Store*. In order to be selected, good developers have to make certain costly actions to make their innovation ability easy to observe, and then signal it to others in the market for a chance to be selected by users. Those costly actions may also be made by developers who are not highly qualified. It means they have to pay more in terms of opportunity costs to compete for a chance to be selected. Thus, signals are more likely to be generated by developers having genuine high quality. The model has explored three costly actions to constitute a signal, including differentiation, diversification, and following.

Signalling is observed differently by receivers having different abilities. Good developers have a higher ability to interpret the correct signal and quality, similar to how *Google* would interpret it. Thus peer recognition by good developers works as another signal to transmit to the developer community. By clustering together, good developers can claim the market and collectively increase their user base to benefit each other in the cluster. *Google* observes this peer recognition signal to understand a developer's quality, and users would take *Google's* selection as a guarantee for quality. Each party in the *Google Play Store* collectively contributes with their knowledge to reduce the information asymmetry. This collaborative effort helps to develop the *Google Play Store* into a sustainable ecosystem.

Chapter 6

Research Method

This chapter discusses the research method designed to examine the signalling selection model developed in the last chapter. The signalling selection model is observed and inspired from the featuring selection mechanism in the *Google Play Store*. So the thesis aims to visualise the history of featuring selection in the *Google Play Store*. To achieve this goal, multiple online data sources were determined to scrape for this research. The scraped data sources include the apps' creation time, developers' competitive behaviour, *Google*'s selection decisions, and app market business performance. They form a panel data set with the connection between apps, developers and *Google* in the *Google Play Store*. Two groups of developers are filtered out for analysis. The first group includes all the featured developers who received the "Top Developer" award. They are classified as innovative developers. By tracing other apps that are similar to the apps created by these innovative developers whose developers did not receive the "Top Developer" award, these are then classified as less-innovative developers and forms the second group. The thesis compares and contrasts the strategic behaviour by the two groups of developers in the next two chapters. According to the nature of this data, some strategic behaviour needs to be studied in a qualitative method and some can be quantitatively studied.

6.1 Data Source & Collection

In order to examine the model, I collected data from the *Google Play Store* to make inductive qualitative (see Chapter 7) and statistical quantitative analysis (see Chapter 8) on the *Google Play Store*. This section introduces the data of interest to this study, and then explains how the data was collected.

6.1.1 Fundamental Data in App Level

Each app is uniquely identified with *app_id* in the *Google Play Store*. The *app_id* is fixed when a developer decides to publish the app to users. Developers cannot change their *app_id* after it is released, unless they decide to completely remove the app from the store. After purchasing a particular app, a user can reinstall this app or continuously receive updates for this app, according to this *app_id*.

There have been several articles published which have experimented and identified that apps can copy each other or try to confuse users with similar app names ([Chia et al., 2012](#); [Felt et al., 2011b](#); [Pearce et al., 2012](#); [Ramu, 2012](#); [Zhou et al., 2012a](#); [Zhou and Jiang, 2012](#); [Zhou et al., 2012b](#)). A unique *app_id* can help user to avoid such confusion.

The *app_id* is used as a key to map other useful information or attributes about a particular app in my study (see Figure [A.8](#)). So I am using this unique variable to structure the relationship between app attributes for an app. I am listing some of the most valuable attributes for an app that can be used in my study (most of them can be found in Figure [A.9](#)).

- *app_cate*: Each app should fit into a specific category and developers decide when they publish the app on the market. *Google* defines the categories into 18 game categories (see Table [A.13](#)) and 23 non-game categories (see Table [A.12](#)) in the store. The apps in each category have some similarities on functionalities. *Google* regularly promote and feature apps in each category, and users can also browse top apps based on *Google*'s category classification. This provides the fundamental criteria to group apps in the study.
- *app_price*: Each app can be either free or paid. The decision is made by the developer at the time of publishing the app. The purpose of developing an app is to generate revenue. A paid app can immediately generate revenue when users purchase it with premium strategy. However, many developers utilize a freemium strategy ([Anderson, 2009](#)), which means that the revenue may generate from continuous use or may even come a long time after the original download. It is worth mentioning that this attribute can be changed by the developer after publishing the app, as some developers prefer to set the app to be free temporarily or permanently. This change is made to attract users in the short term or to completely give up a premium strategy. However, as I cannot track each change on price for each app, and most changeable apps only alter their price strategy for short periods, I assume the time I captured the *app_price* could be used to represent the pricing strategy for most cases.

- *inapp_billing*: This utilises a special permission used in the *Google Play Store* ¹ and was only introduced two years after the invention of Android ². Originally, a developer can only have free or paid apps. So if the developer would like users to try the app before spending money, they have to create two apps, one free app with limited functionalities and the other one a paid-for app with complete functionalities. *inapp_billing* solved this problem by enabling users to try first and then pay for it later in the same app. It achieved great success as "23 of 24 top-grossing apps in the Google Play use In-app Billing" ³. After one year of introducing *inapp_billing*, "the total revenue generated from in-app purchases exceeds revenue from traditional app purchases". This attribute provided additional information of the freemium strategy. Therefore, I am using a binary to represent whether an app is using this strategy or not. *1* means use, and *0* means not in use.
- *app_install*: The early published app may have a larger number of installations. However, a good app quality may greater influence an app's installation figure. A larger number of installations represents the popularity of the app, and it would theoretically bring more revenue, regardless of whether the premium or freemium strategy is utilised. Meanwhile, a large installation base would signal to platform owners and end users a positive signal, potentially resulting in a snowball effect as the installation base would continue to expand. *Google Play Store* only shows the installation number for an app in categorical data format, for example, *app_install* is in the order of: *0-5*, *5-10*, *10-50*, *50-100*, ..., *50,000,000-100,00,000*, and so forth (see Table A.14 for the complete list). The range is very wide and therefore not very useful in statistics, but it does gives a general indication for observation.
- *reviewer_count*: This is the total count of users that have reviewed the app. *reviewer_count* is much smaller than *app_install*, however, users who add a review usually really care about the performance and quality of an app. As *app_install* is in categorical data format, I used *reviewer_count* as another variable to proxy the installation base. An illustration of this attribute can be seen in Figure A.9.
- *avg_review_score*: Users who downloaded the app can post their review opinion to generate a review score. At each time, users can choose to put a review score from 1 to 5 with [*1*, *2*, *3*, *4*, *5*], *5* meaning good and *1* meaning bad. For each review,

¹http://developer.android.com/google/play/billing/billing_overview.html

²<http://android-developers.blogspot.co.uk/2011/03/in-app-billing-launched-on-android.html>

³<http://android-developers.blogspot.co.uk/2012/05/in-app-subscriptions-in-google-play.html>

Google multiplies the score and user count to get a value for each score, and then sums it all and then divides by the total count of users in all scores. This results in the final average *avg_review_score*, see Figure A.9. As an example, *avg_review_score* = 4.3, and *reviewer_count* = 545462. Seen as a calculation this is: $4.3 = (352260 \times 5 + 94348 \times 4 + 36851 \times 3 + 16868 \times 2 + 44555) \div 545462$. Due to how the calculation is modelled, the *avg_review_score* can basically represent the user satisfaction.

6.1.2 Fundamental Data In The Developer Level

Similar to the as *app_id*, each developer has a unique identification in the *Google Play Store*. It is normally equivalent to developer's name but encoded for Internet URL readability⁴. For example, a developer called **Dropbox, Inc.** is assigned with a unique ID called **Dropbox,+Inc.**. Developers can not change their *developer_id* if their apps are developed under this developer name. With *developer_id*, the *Google Play Store* has developed several ways for anyone to easily identify the developer behind an app, and even to check all the apps belonging to a particular developer, (see figure:A.18). Like a branding, the *developer_id* becomes the representation of a reputation for each developer to maintain on. As discussed before, apps can be very similar by name so knowing who developed this app may reduce the confusion by letting users know whether a certain app is the particular one that they were looking for. As each app belongs to a particular developer, I can count the number of apps (*apps_count*) the developer has to represent their productivity.

Developers have many useful attributes as well. Featuring (*is_featured*) is the first attribute and most important character to study. It represents an award given by platform owners (the *Google Play Store* team). From Figure A.18, it is easy to see that **Dropbox,+Inc.** has a blue badge called **Top Developer**. It is a sign to confirm that this is a *featured developer*. A *featured developer* can attract lots of attention and bring confidence to users. It is an attribute each developer is dreams of achieving. This is a life time award, so once obtained, all of his following apps display this badge in the website. In other words, a *featured developer* is recognised by the platform owner at some point in time, and a *non-featured developer* is somebody who has never received the “**Top Developer**” award from *Google*. All developers can be either *featured* or *non-featured* in a particular moment. There is another kind of award called **Editor's Choice** in the *Google Play Store* which can also be seen in the Figure A.14. **Editor's Choice** is not useful in this study for two reasons. The first reason is because there are less than 40 apps that have this award after data collection, so this award is not seen by this study to be a good representation of the status of designers. For many complex reasons, the *Google Play Store*

⁴http://www.w3schools.com/tags/ref_urlencode.asp

would often change the list of apps obtaining the second award. I am unable to obtain the complete changing history of the *Editor's Choice* list in the market as it is not a life time award. The second reason is because developers who obtain the second award already have the first award. So I assume that studying the first award already covers the second award.

The second attribute is privacy policy (*has_privacy*). Developers can declare the privacy policy for the apps they have developed. Users may feel confident to see the privacy policy statement before they install an app. However, having a privacy policy can restrict what developers can do with their apps. Therefore, it is a trade off attribute.

6.1.3 Fundamental Data In Permission Level

Permission is the more granular component of an app. The great thing about *Android* devices is that it opens permission for developers to create fancy functionalities by using fundamental resources. Those fundamental resources belong to users but are necessary to have for certain functionalities in the app, and they are defined as permission by Google ⁵. Permission is a text declaration in the app and can be recognised by an *Android* device at the time of its installation. It can be related to sensors or even about user data in the phone, and basically enables developers to touch every component of a user's mobile phone. Figure A.19 is a screenshot showing that a particular app would request a certain combination of permissions. Figure A.26 shows the distribution of permissions requested by apps in the *Google Play Store*. Users have to agree for the apps to use those permissions, otherwise, an app cannot be installed and run on the user's *Android* mobile phone. Later updates would not request same permissions unless additional permissions are required by an app. Meanwhile, permission cannot be individually granted by users, as an app would use a combination of permissions.

Permission has been developed with the very development of *Android* SDK. Some new permissions can be added when new hardware components become popular, but some permissions have disappeared because of the loss of popularity and the identification of security issues. Each permission has a unique name (named as *perm_id*) in the SDK ⁶. Developers have to explicitly declare them in the source code in order to use them. Permission descriptions are designed for ease of understanding (Felt et al., 2011c), however, it is still very technical for average users (Felt et al., 2011a). Some studies (Felt et al., 2012) found that people usually ignore reading the permission before they grant permissions in an app. This would lead to a security problem and damage the health of the ecosystem in the app market (Sarma et al., 2012). So *Google* provided a higher level of classification to group jointed permissions ⁷ and assigned

⁵<http://developer.android.com/guide/topics/security/permissions.html>

⁶<http://developer.android.com/reference/android/Manifest.permission.html>

⁷http://developer.android.com/reference/android/Manifest.permission_group.html

a friendly icon to each group. The purpose is to make permission easier to understand by users and help to easier decision making on purchases⁸. There are 17 permission groups (see Table A.15) including in-app purchases. As I already recognised the *inapp_billing* variable, I only consider the remaining 16 permission groups and named them as *perm_group*.

6.1.4 Additional Data Variables From *AppBrain.com*

Regarding time information, the *Google Play Store* has only disclosed the latest update date for each app, but it is worth knowing when an app was initially released to the market. By having this initial release information, I can investigate apps according to their release time, so I can order and recreate the market evaluation history. This information can also help me to understand the complete history of developers.

After research, I found that *AppBrain.com*⁹ has been monitoring the *Google Play Store* since the very beginning, and could provide a comprehensive database of initial release data for most apps. By passing *developer_id*, *AppBrain.com* returns a complete list of apps made by this developer, and the initial release date with *app_id* is included in the results. By matching to the *app_id* in the *Google Play Store*, I was able to add this *release_date* as a temporal attribute to the relevant apps.

6.1.5 Additional Data Variables From *WayBack Machine*

As previously mentioned, the *Google Play Store* states whether a developer has obtained the “Top Developer” award. However, it does not disclose when a developer obtained this award and what is the order for apps to make the status transition. This transition of status from *non-featured* to *featured* developer is a useful variable to understand the developers’ behaviour in this study. There is no direct way of obtaining this transition information. A possible solution is to compare the multiple time stamps in the app history to check whether the *is_featured* has been changed over time for each apps belonging to currently *featured* developers. *Wayback Machine*¹⁰ archives Internet webs pages and also frequently archived webs pages from the *Google Play Store* and even previously known *Android Market*. By passing the web URL for each app by *featured* developers to *Wayback Machine*, I have been able to get all the historical archives of these websites (see Figure A.20), and then for each archived web I checked the *is_featured* in each historical time for each app. With some manual work, this process determined which apps made the transition of status for each developer. The algorithm

⁸https://support.google.com/googleplay/answer/6014972?p=app_permissions&rd=1

⁹<http://www.appbrain.com/>

¹⁰<http://archive.org/web/>

and *Python* code have been demonstrated in Appendix A.2. For *featured* developer, transit apps are coded as 2, apps released before transition are coded as 1, and apps released after transition are coded as 3. For all *non-featured* developers, their apps are coded as 0. It is worth mentioning that some *featured* developers have their first app with a coding of 2, those developers are identified as born to be featured.

6.1.6 Collection Method

The *Google Play Store* can be accessed from either the web ¹¹ (see Figure A.16) or a mobile app (see Figure A.16). Both formats display the same information, but the layout is different due to the screen size. The web version of the *Google Play Store* contains all public information visible to users. So the data collection was targeted on the web based *Google Play Store*. As I am only interested in apps, I did not collect other irrelevant data (e.g. Music, Books, etc.). I carefully designed the whole data collection process and obtained ethical approval from Nottingham University Business School.

Each app has a profile web page on the *Google Play Store*. The profile provides almost all fundamental data, for example, who is the developer, what rewards it has received (see Figure A.8), how users review the app, updating, price, user downloads, permissions, privacy policy and the link to the developer (see Figure A.9). It also lists the links to other apps similar to the app (see Figure A.10) (which is determined by *Google*). Links to similar apps help to visit more profiles for apps in the *Google Play Store*. The app profile web page contains a link to its developer's profile. The developer's profile (see Figure A.18) contains links to everyone of their apps. This helps to find out every app for a specific developer. I have used this procedure to find out every app in the *Google Play Store*. Additional data has been obtained from *AppBrain.com* and *WayBack Machine*. Those data can be connected with unique *app_id* and *developer_id* from the *Google Play Store*. *AppBrain.com* also listed all the apps it was watching, so it provided a second source to make sure I have collected as many apps as possible.

The purpose was to collect as much information about apps and developers as possible. So I developed a *python* program to scrape the *Google Play Store* web page automatically. The program takes the *Google Play Store* front page as a starting point, and then scans each web page to filter out the links to apps contained and also the links to developers. For each app profile web page, I record all necessary public information by ignoring any private information that could identify users in the *Google Play Store*. Each app is identified by a unique *app_id*, and each developer has a unique *developer_id*. I used those unique IDs to prevent duplication of data collection. Meanwhile, I also scraped the sites *AppBrain.com* and *WayBack Machine*.

¹¹<https://play.google.com/store/apps>

The program continues to scrape until there is no new *app_id* and no new *developer_id* in the job queue.

The program has been set up to run twice. The first collection finished in September 2012 and collected about 567,730 apps and 112,223 developers. The second collection finished in October 2014 and collected about 1,340,505 apps and 232,397 developers. Public information disclosed that the *Google Play Store* should have 675,000 apps¹² in September 2012, 1,300,000 apps in July 2014 and 1,430,000¹³ apps in December 2014. It means the first data collection reached 84% of total apps ($567730/675000=84\%$) and the second data collection reached at least 93% of total apps ($1340505/1430000=93\%$).

I decided to use data from the second data collection process for this study. The first reason for this decision was because *Android* was still not fully developed during first data collection period (see Figure A.7 and Table A.6). *Android* SDK has been frequently updated in 2011 and 2012 and the fragmentation issue was a big problem at that time. *Google* started to separate its proprietary service from the *Android* SDK by announcing the *Google Play Services API* 1.0 on the 26th September 2014¹⁴. During the second data collection period, there were significantly less updates to *Android* SDK. The second reason is that *Google* has changed its permission mechanism¹⁵ so the data obtained the second time was more accurate. The third reason is that the second data collection has reached a higher percentage of coverage and the market selection effects have been observed for longer time. My interest is to focus on the later crowded stage rather than on the early stage of the market. The priority on the early stage for *Google* is to attract developers and the priority during the later stage is to control the quality of apps. The market is also much more crowded in later stage. By simply comparing the number of apps and developers in the market for two times that becomes clear. Therefore, this study is based on data from second collection time.

6.1.7 Data Overview

Based on second collected data, there are only 365 developers selected to be featured as “Top Developer” out of 232,397 developers. This highlights the extremely difficulty for obtaining the featuring opportunity and that the competition is extremely high. Those featured developers have in total produced 3542 apps and those apps have a special badge in their app profile. The average number of apps by a general developer is 5.7 ($1340505/232397=5.7$) and the average

¹²<http://officialandroid.blogspot.co.uk/2012/09/google-play-hits-25-billion-downloads.html>

¹³<http://blog.appfigures.com/app-stores-growth-accelerates-in-2014/>

¹⁴<http://android-developers.blogspot.co.uk/2012/09/google-play-services-and-oauth-identity.html>

¹⁵<https://support.google.com/googleplay/answer/6014972?hl=en>

number of apps by a featured developer is 9.7 ($3542/365=9.7$). There are 95 featured developers only having one app produced and the largest number of apps by a featured developer is 142.

Because good developers are the targets to study in this thesis, I have mainly focused on featured developers. From Figure A.21 and the summary we can see the average number of categories one developer crosses over. Nearly 3.43% of developers can only work on one category and 92% of developers work on less than 8 categories. The maximum number of categories developers can work over is 15 but only very few developers can do that.

The next chapter inductively analyse how developers signal their innovation ability. The chapter after that presents the statistical analysis for the model.

6.2 Qualitative Research Methods

In order to understand the developers' strategic behaviour in the app economy, a qualitative study has been made (see Chapter 7). It has filtered out all "Top Developers" in the dataset (totalling 365 featured developers). Most top developers have a privacy policy to the public. A large amount of literatures (Kalapesi, 2013; Schwab et al., 2011) has studied the importance of user data to a Internet business as a unique resources. This privacy policy discloses how a developer requests user data via his apps.

The qualitative analysis started with the research on app permissions. It studies the relationship of app permissions requesting behaviour in app innovation, and then classifies app permissions into three groups (each group can access a type of user data). Privacy policy enables me to study the relationship between user data behaviour in app innovation. This classification of app permission and user data enables me to group developers with their apps in different scenarios to understand their business models and reuse behaviour. So at the end of this study, I was able to compare and contrast five special scenarios to find out why does a developer demand the same user data in diversification, why developers do not use user data, why developers only use a specific kind of user data, and why developers use all types of user data in their apps, and why developers do not diversify.

This study method enables me to improve the traditional business model analysis by specifically fitting it into the user data context for the app economy. It confirms that developers are signalling in app innovation. However, it is still not clear what are the details of signalling patterns and what is the effect of platform selection. A quantitative method is designed to answer this questions.

6.3 Quantitative Research Methods

A quantitative analysis is designed to examine the pattern and efficiency of a signalling selection model (see Chapter 8). The purpose is to study the relationships between different parties in the *Google Play Store*, so the first job was to identify three comparison groups for each featured developers.

It then projects each app chronologically according to the release time in each comparison group. This forms a statistical dataset in long form panel data format to examine the platform selection and peer recognition.

6.3.1 Panel Data Analysis

Statistical modelling adopts panel data analysis (Wooldridge, 2010) in this thesis. Panel data (also known as longitudinal or cross-sectional time-series data) is a dataset in which the behaviour of entities are observed across time. These entities could be states, companies, individuals, countries, etc. Panel data allows for the control of variables that cannot be observed or measured like cultural factors or differences in business practices across companies; or variables that change over time but not across entities (for example, national policies, federal regulations, international agreements, etc.). This is accounted for individual heterogeneity.

When individuals form groups or clusters, it might be expected that two randomly selected individuals from the same group will tend to be more alike than two individuals selected from different groups (Singer and Willett, 2003). By a similar argument, measurements taken on the same individuals at different occasions, e.g., physical attributes or social attributes, tends to be more highly correlated than two measurements from different individuals. Such dependences can therefore be expected to arise and multilevel models are needed to analyse data with a hierarchical structure. Throughout this model, the lowest level of observation in the hierarchy (e.g. app measurement) is referred to as level 1, and the group or cluster (e.g., developer) as level 2.

In my model, each app is a repeated measure of occasion when a developer has a new app. The data frame for the repeated measures structure is often in long form (it has one row per measurement occasion) or wide form (one row per individual with a column for each measurement occasion). The wider form tends to correspond to be the traditional multivariate framework for analysing repeated measures data (where all individuals must be measure at the same time points) and the long form tends to correspond to the multilevel framework for analysing repeated measures data (where individuals can be measured on a different set of occasions from each other). Importantly, the multilevel approach does not require that each individual is measured on every occasion, that is the reason why a multilevel framework does

not need to be balanced. The spacing between occasions can also differ between individuals. So it would benefit to the nature case of my dataset that the time duration of two consecutive apps can be different. So the data is represented in long form, and the time occasion is represented by two consecutive apps according to release time.

Fixed Effect or Random Effect Estimator

I need to choose an estimator to use in panel data analysis and the most commonly used two estimators are fixed effect estimator and random effect estimator. Fixed effects are suggested to analyse the impact of variables that vary over time. Fixed effect explores the relationship between predictor and outcome variables within an entity. Each entity has its own individual characteristics that may or may not influence the predictor variables. When using the fixed effect, it is assumed that something within the individual may impact or bias the predictor or outcome variables and it needs to control for this to happen.

Unlike the fixed effects model, the rationale behind the random effects model is to measure that the variation across entities is assumed to be random and uncorrelated with the predictor or independent variables included in the model. Random effects assume that the entity's error term is not correlated with the predictors which allows for time-invariant variables to play a role as explanatory variables. [Greene \(2003\)](#) claimed that *"the crucial distinction between fixed and random effects is whether the unobserved individual effect embodies elements that are correlated with the regressors in the model, not whether these effects are stochastic or not"*.

To decide between fixed or random effects, the *Hausman* test has been invented where the null hypothesis is that the preferred model is random effects, and the alternative is fixed effects ([Greene, 2003](#)). It basically tests whether the unique errors (ui) are correlated with the regressors and the null hypothesis is that they are not. *Stata* has a specific command called *hausman*¹⁶ to make this test.

¹⁶<http://www.stata.com/manuals13/rhausman.pdf>

Chapter 7

Qualitative Analysis

The last section has introduced that there are a total of 365 featured developers in the data source. Each developer has disclosed his company website and privacy policy in his app profile. This chapter discusses the qualitative analysis on their companies' websites and privacy policies, in order to understand the trajectory of how they behave when selected by *Google* as featured developers. This inductive study provides the foundations for quantitative analysis in the next chapter.

The study starts from the app permission because permission is the key for developers to have certain functions in the apps. More importantly, it is the foundation for developers to acquire and use users data in their apps. Based on *Google*'s classification and developers' privacy policy, I have classified user data into three groups, including *personal* user data, *social* user data and *system* user data. This classification enables me to find that business models in the app economy are different from the traditional economy due to the fact that user data is an important resource in the app economy. The study on privacy policy helps me to list the components of user data business model based on [Morris et al. \(2005\)](#)'s work. With this model, I then empirically study three interest groups of featured developers in the data set. The first group contains every featured developer who has more than one app but keeps using the same user data in all of their apps. This group of developers choose the diversification strategy, but only have one type of interest in the user data. The second group contains every featured developer who has only one app, which is a strategy to avoid diversification.

7.1 App Permission

Section [6.1.3](#) introduced what is permission. For the distribution of permission requested by apps, please refer to Figure [A.26](#). In simple words, a permission is a "restriction limiting access

to a part of the code or to data on the device”¹. *Android* has listed out all standard permissions with technical explanations in the documentation². Some app functions have to be implemented with a specific permission in order to carry out certain functions; for example, an app has to request *CAMERA* permissions in order to have camera features. Some permissions would bring extra functionality. For example, *READ_CONTACTS* could bring apps extra data about a user’s contacts to provide social features or to save time for users to input their friend into the app. Developers can thus combine different permissions to stack up different functionalities. For example, developers can use both *CAMERA* and *READ_CONTACTS* permissions in the same app to provide photo sharing features. In other words, permission is a foundation for innovation on app function.

However, figuring out how to use permissions requires technical skills. Using permission requires technological leaning ability. Developers have to understand how to use permissions from various sources. Developers also have to keep an eyes on the technological developments as permission rights keep changing when *Android* upgrades SDK. Innovative developers also pay attention to new permissions and aim to understand how to use new permission to bring fresh new functionality in quick succession to obtain a first-move advantage in the market. For example, *BODY_SENSORS* permission “allows an application to access data from sensors that the user uses to measure what is happening inside his/her body, such as heart rate.”. It is a newly introduced permission to welcome the new trend of the smart watch. Figuring out how to use this new permission provides opportunities for developers to provide attractive features with innovative wearable technology.

It also requires developers to optimise user permissions. Permission request is the foundation to provide its functionality, however, requesting too many permissions would reduce the user experience. Too many permissions simply means too many activities. It would cause rapid battery consumption in users’ mobile devices³. It also introduces the risk to expose the app to malicious attack⁴. Some lead users may also doubt on an app’s intention, which would reduce the confidence to install the app⁵. If some lead users identify improper use of permission by developers and exposure this to the public, a developer’s reputation is significantly impacted⁶⁷.

Another reason for app permission requests is to allow the developer to work on user data. Even in some cases, (for example, GPS related permission looks like the interaction between

¹<http://developer.android.com/guide/topics/manifest/manifest-intro.html#perms>

²<http://developer.android.com/reference/android/Manifest.permission.html>

³<http://www.cnet.com/uk/news/whats-killing-your-battery-androids-top-10-performance-sapping-apps/>

⁴<http://www.wired.co.uk/news/archive/2014-04/01/x-privacy-android-app>

⁵<http://www.bbc.co.uk/news/technology-25258621>

⁶<http://venturebeat.com/2012/02/14/iphone-address-book/>

⁷<http://www.businessinsider.com/path-app-stores-contacts-on-servers-2012-2>

device and hardware), the device hardware is still responding with user data back to the app via permission (which is user's location data in this case). Data can be written to or read from a user's device. The granted permission allows developers to interact with user data in various forms. Recently, both industry and academics have realised the value of user data (Kalapesi, 2013; Schwab et al., 2011). So the next section focuses on user data in the apps.

7.2 User Data

It is useful to have a higher level of classification on user data to understand what kind of user data one developer is interacting with. After studying the technological explanation on app permissions^{8 9 10}, my interest is to find out whether developers are more interested in working on a user's personal data to individually identify users, or on social data to have a collective insight by knowing how a user interacts with other people, or device system data to understand the background operation of an app. Based on each individual permission, Google also classifies permission into 17 groups¹¹. This official classification already provides the fundamental criteria for groups based on the permission operation, but is not specific for user data level. By adding extra criteria of how the data immediately relates to a user, I can classify the permission groups into three user data types. They are *personal* user data, *social* user data and *system* user data. *Personal* data represents user data that is only related to the user, but *social* data refers to user data not only about the user, but also about other people such as their friends, and *system* user data concerns the rest of the data, which is more relevant to system devices.

This classification criterion on user data also comes from the inductive analysis of privacy policy statements by *featured developers* which is introduced in later section. Developers demand user data in different purposes and methods. An example¹² of the statement from a featured developer on what information they are collecting in their app is¹³:

*“When using our Applications and Services, we may ask that you provide us some **personally identifiable information** – information about you that can be used to contact or identify you, and information on your use or potential use of the services and related services.*

⁸<http://developer.android.com/guide/topics/security/permissions.html>

⁹<http://developer.android.com/guide/topics/manifest/permission-element.html>

¹⁰<http://developer.android.com/reference/android/Manifest.permission.html>

¹¹<https://support.google.com/googleplay/answer/6014972?hl=en>

¹²<https://newsrepublicstatic.blob.core.windows.net/html/cgu/privacy.htm>

¹³<https://play.google.com/store/apps/details?id=com.mobilesrepublic.app>

*Personal Information that we might collect includes information like your email address. Personal information may also include other information you supply to us concerning your **preferences and interests** expressed in the course of using the service. We use Personal Information to: **administer** our Application & Services, enable you to **easily navigate** the services, **provide, improve and optimize** the services, **better understand user needs and interests**, **personalize your experience**, provide you with **software updates** and/or **product announcements**, and provide you with further information and offers from us that we believe you may find useful or interesting.*

*When you use our Application & Services, our servers will automatically **record information** that your mobile device sends or transmits, including, but not limited to, the device ID number, user settings, location and information about your use of the services. We only store, track and monitor general usage and not information specific to any user. We use this information to **improve** the Services, **develop** new products and **offer** products and services that may be of interest to you.*

*We also use **cookies** on our Application & Services to collect information. We use cookies to save your preferences and **personalize** the service for you. **Third-party advertisers** may also place or read cookies through the Application & Services. You can instruct your device or browser, by changing its options, to stop accepting cookies or to prompt you before accepting a cookie. If you do not accept cookies, however, you may not be able to use all portions or all functionality of our Applications & Services.”*

From this statement, I can see that developers need personal data to identify who is the user and then provide customised services to each specific user. It should need system data for technical purpose. Some other developers are relying on social data, especially for social networking features. For example, *FourSquare* claims in their privacy policy ¹⁴ that: “Information you provide to one of our Services may be used with other Services you or your friends use. For example, location information we receive in connection with your use of one of our apps may **be used to customize the content you and your friends see in another of our apps.**”

As mentioned before, *Google* defined fundamentally more than 100 permissions to be used in app development. Based on this permission base, *Google* then classified 17 permission groups to help users easily understand the purpose of permissions requested by an app. In order to study the demand and value of user data in an app, I proposed a higher level of classification

¹⁴<https://foursquare.com/legal/privacy>

of user data based on permission groups, and my classification criteria is coming from inductive analysis of the developers' privacy policy statements. With this three tiered relationship, I could explicitly map the user data for each app and then make the study of user data in apps. Each app can be linked to a particular developer, so further more, it enables me to study what user data are demand by developers. The relationships among permission, permission group and user data are listed in Table A.15.

7.3 Privacy Policy

In order to systematically understand the motivation of user data demand, I have studied developers' privacy policies. A privacy policy is a statement or a legal document that discloses some or all of the ways that a developer gathers, uses, discloses and manages users' data. It fulfils a legal requirement to protect a customer or client's privacy. Nearly all featured developers disclose their privacy policy in their apps' profile. Conversely, there are many non-featured developers having no privacy policy and the majority of them are individual developers rather than company developers. A real example of privacy policy could be checked here ¹⁵.

The content of a developer's privacy policy is dependent on the applicable law and may need to address requirements across geographical boundaries and legal jurisdictions. Every developer is trading their apps on the *Google Play Store* and their users can be spread around the world. However, user data may be physically stored in one or multiple physical locations. Therefore, the content is restricted to the laws in each country. Many featured developers are even applying and using online privacy certification programs like *TRUSTe*. *TRUSTe* ¹⁶ is an online privacy seals that accesses, monitors, and certifies websites, mobile apps, cloud, and advertising channels to allow companies "*to safely collect and use customer data to power their business*". *TRUSTe* accepts consumer complaints to enforce their privacy rules. Those actions demonstrate that developers are very serious about user data and their use of use data.

The purpose in analysing a developers' privacy policy is to study the relationship between user data and innovation. This study helps to understand how developers signal their ability to innovation via permission. The privacy policy provides a safe harbour for companies to use user data for innovation as users have to accept privacy policies in order to use an app. However, it also restricts developers to act only within the boundaries of privacy policy statements. I have randomly selected 100 developers to look at their privacy policies. Their apps cover every category in the *Google Play Store*. In addition to saying what information they need, developers

¹⁵<https://evernote.com/legal/privacy.php>

¹⁶<https://www.truste.com/>

generally also disclose how they use the collected information. Bellow is an example quoted from Apalon ¹⁷:

“ We may use your information we collect to:

- Deliver the Apalon Apps to you and ensure their optimal functionality and operation;*
- Provide you with customer support;*
- Perform research and analysis about your use of, or interest in, our products, services, or content;*
- In accordance with applicable law, communicate with you by e-mail, postal mail, telephone and/or mobile devices (including through in-application advertising) about products or services that may be of interest to you either from us, our parent or subsidiary companies or other third parties. If you do not wish to receive in-application advertising, you can upgrade to the paid version of the app.;*
- Develop and display content and advertising tailored to your interests and to measure its effectiveness;*
- Verify your eligibility and deliver prizes in connection with contests and sweepstakes and in order to conduct polls and surveys;*
- Enforce our terms, conditions and policies and to communicate with you;*
- Manage and improve the Apalon App. For this purpose, we may use and combine your information which we collect across different Apalon Apps used by you;*
- When we collect your precise GPS geolocation data (subject to your consent), we do so for the purpose of providing you with our location-related products and services, for example to provide you with forecast and weather alerts for your location.*
- When you share your information with us via a social network provider like Facebook, we use this to show information on your social network profile (subject to your privacy setting and preferences for such social network) and to allow you to share information about our services with your connections or on your social network profile.*
- When you use the Apalon App, we may use the App Identifiers to facilitate access to the Apalon Apps, help us recognize you across different Apalon Apps, to enable you to use the Apalon Apps, to personalize your experience,*

¹⁷http://www.apalon.com/privacy_policy.html

to increase security, measure the use and effectiveness of our Apalon Apps and service advertising. You can block certain App Identifiers by changing the settings of your device.

”

7.4 How Do Developers Innovate With User Data?

Apalon’s statement in the last paragraph already tells a lot about how developers innovate with user data. After reviewing the privacy policies for other developers in the sample, I have summarised in bellow how developers use user data for innovation.

- *User data demands are significantly dependent on what the business developers are doing and on the functionalities that developers provide in the app.* For example, cloud apps always want to connect to the Internet, apps having locating functions always collect location data, and social network apps always want to access to users’ contact books, etc. By looking at an app’s permission request, we can “guess” a lot about the functionality that the app is providing. Alternatively speaking, if other developers want to learn how to implement a particular function in a particular app, they can always first learn about how an app requests permissions. In other words, permission provides a way for developers to learn from each other.
- *Game apps require significantly less user data than other apps.* The reason could be that game apps have less demands on user data exchange, if we compare them to other apps like social networking ones. Another reason is that the *Google Play Services API* already provides support for game data exchange. This API works as a common infrastructure for every game developer. Developers can worry less about technology requirements, and users have less chances of data leaks because most data is exchanged within the *Google Play Service* rather than on developers’ own servers. Developers, especially game developers, have to keep an eyes on how do such third-party APIs work.
- *User profile is always demanded.* Nearly all apps are connecting to users. Social network app need to connect based on user identification. Game apps also need to store and restore game sessions based on user identification. Even apps that only control a flashlight on the phone need to identify users to some degree for the purposes advertising ¹⁸. User profiling is for technology implementation but also for a business model. A common technology for user profiling is *cookies*. For example, *Dropbox* explained in their privacy

¹⁸<http://www.theguardian.com/technology/2014/oct/03/android-flashlight-apps-permissions-privacy>

policy ¹⁹: “We use technologies like cookies and pixel tags to provide, improve, protect and promote our Services. For example, cookies help us with things like remembering your username for your next visit, understanding how you are interacting with our Services, and improving them based on that information.”

- *Developers use user data to add value to their apps by customising their services to each user.* User profiling provides the first step to understand who the user is. The more user data is collected and analysed, the better is the position for developers to understand their users in order to provide tailored services. Understanding a user is a time consuming process which requires users to continually use the app so that later the developer may analyse the user’s digital footprint. Even on the first use after installing an app, the developer can use existing data to efficiently understand users to provide a customised service and improve the customisation as the user continues to use the app. For example, *SwiftKey* aims to learn from users to help them type on a touchscreen faster and easier. It explains in its privacy policy that: “When you are using our Products, our prediction technology learns from the way you use language. Upon first installation we need to establish a base for your personalised language model. To achieve this, *SwiftKey Keyboard for Android* accesses your SMS messages.” ²⁰
- *Developers use user data to connect and interact with other developers’ products.* Many developers would demand the same type of user data. That data needs time to be generated, and it also takes motivation for users to generate it. Therefore, an easier method is to reuse the user data already captured by other developers. For example, a photo editing app may want to access users’ *Flickr*’s photo storage to have photos to edit. Some developers start to open some of their user data to attract collaboration to maintain their competitive advantage. For example, *Dropbox* claimed in their privacy policy: “Some of our services let you access your accounts and your information with other service providers”. Another example is *Facebook*. Other developers can enable automatic filling for their user registration form when connecting to the *Facebook* app ²¹. *FourSquare* also explains in its privacy policy that: “When you use the Services, certain information is public, such as your public user profile information, tips, likes, saves, public photos, lists that you create and save, tastes, badges/stickers, mayorships, and lists of friends and/or followers (as applicable to a particular Service), and may be searchable by search engines and be shared with or disclosed to third parties, including through our API to a wide range of users and services. We may also share or disclose aggregated and anonymous data, such as de-identified check-in data (that is, not tied to

¹⁹<https://www.dropbox.com/privacy>

²⁰<http://swiftkey.com/en/privacy/>

²¹<https://developers.facebook.com/docs/facebook-login/v2.3>

individual users), to third parties, including users you may follow and users of our API, for analytical or other purposes.”

- *In order to provide better services, developers may have to integrate more user data from other developers’ products. One developer may not only interact with other developers’ apps, but also integrate more user data from them. User data enables the bridge for integration. For example, SwiftKey claimed in their privacy policy that: “The SwiftKey personalisation service, which is a feature of SwiftKey Cloud, accesses your recent content from online services that you specify, such as Gmail, Facebook and Twitter. It uses this content to build a personalized language model on our servers, which is then transferred to your device. This is an optimized view of the words and phrases that you use most often, and reflects your unique writing style.”. FourSquare also confirmed that “We receive and store information from third parties that interact in some way with the Services or that provide services to us in connection with the Services. For example, if you unlock a Foursquare special that is redeemed through a credit card swipe, the credit card processor may tell us information about the redemption such as the time that your card was swiped. In addition, you may choose to use apps that share your Personal Information, activities and/or content with Foursquare. Please read the privacy policy of any such app so that you understand its sharing practices.”*
- *Developers use user data to ease their marketing. This happens a lot when a developer wants to invite user’s friends to join the app. For example, Dropbox claimed that: “If you give us access to your contacts, we’ll store those contacts on our servers for you to use. This will make it easy for you to do things like share your stuff, send emails, and invite others to use the services.” Spotify also claimed in its privacy policy ²² that: “We use these cookies to analyse how the Spotify Service is accessed, used or is performing. We use this information to maintain, operate and continually improve the Spotify Service. We may also obtain information from our e-mail newsletters, to learn whether you opened or forwarded the newsletter or clicked on any of the content. This information tells us about our newsletters’ effectiveness and helps us ensure that we’re delivering information that you find interesting.”*
- *Developers use user data to determine where and how to innovate. User data normally provides insight on where to improve the product and how. For example, NewsRepublic already stated in their privacy policy that: “We use this (collected user) information to improve the Services, develop new products and offer products and services that may be of interest to you.”. In addition, developers can get feedback and user experience reports from user reviews in the Google Play Store review system. Many developers also state in*

²²<https://www.spotify.com/uk/legal/privacy-policy/>

their privacy policy that they keep the right to contact users when the service crashes or future development is needed.

- *User data is demanded by other users.* For example, *Dropbox* claimed that: “Our Services display information like your name and email address to other users in places like your user profile and sharing notifications. Certain features let you make additional information available to other users”. Users data provide the foundation for users to communicate, collaborate and share. It is also for developers to increase their user base via user data interaction and to enhance their competitive advantage.
- *Developers use user data to improve their other business services.* It normally happens when apps need user participation. For example, the *BBC* enables users to contribute to news sources or evidences in their apps. Another example is *WeatherBug*²³. *WeatherBug* is a free app and users can use it to correct mistakes in weather broadcasts. The corrected information provides feedback to its core business which is to provide chargeable weather services to enterprises.
- *Developers generate revenue from user data.* For example, *A+E Networks*²⁴ claimed that: “Your IP location information may be used during your visit to the Websites to serve you advertisements for goods or services available in your specific location, however, such information will not be collected or maintained by A+E except in aggregate form.”. Another example is *Duolingo*²⁵, a language learning app. *Duolingo* is advertisement-free and single users are not charged for its services. It uses data and process from users learning English to collectively translate for business purposes²⁶.

The privacy policy provides the evidences that user data is an important source for innovation. Developers can innovate based on user data to provide customised services, expand their market size and to connect to other developers’ products (Kalapesi, 2013). It has confirmed that user data helps developers to make internal or external decisions (MacCormack and Verganti, 2003), marketing (Von Krogh et al., 2003), communication (Jeppesen and Frederiksen, 2006) and may be an important component for functionality implementation (Boudreau, 2012). Developers can implicitly generate user data when users are using apps, or integrate external data sources when users disclose these in the app, or share user data among the others apps by developers.

Many developers claim in their privacy policy that user data is an important asset to them. For example, *FourSquare* claims that “If Foursquare or substantially all of its assets are acquired, or in the unlikely event that Foursquare goes out of business or enters bankruptcy,

²³<http://weather.weatherbug.com/about-us/privacy-policy.html>

²⁴<http://www.aenetworks.com/privacy>

²⁵<https://www.duolingo.com/privacy>

²⁶<http://www.technologyreview.com/news/506656/the-cleverest-business-model-in-online-education/>

user information would be one of the assets that is transferred or acquired by a third party.”. After Microsoft bought Skype, Skype now enables Microsoft to integrate services based on user data²⁷. Both Gamevil and Com2uS are developing mobile games and based in South Korea. They were competing against each other, but on 4th October 2013, Gamevil and Com2uS merged²⁸ and set up a new social network game platform called HIVE. Nowadays, they cooperate rather than compete by sharing user data on the platform²⁹. For example, Gamevil states that “Registration Information We strongly encourage you to create a HIVE Account that enables you to receive customer support services with ease and enjoy various features of the GAMEVIL Services.”³⁰, and Com2uS states that: “Registration Information. We strongly encourage you to create a HIVE Account that enables you to receive customer support services with ease and enjoy various features of the Com2uS Services.”³¹

7.5 Business Model On User Data

After studying privacy policies, I found that user data would influence a developer’s business model. The first influence is that user data is considered an important resource to companies, especially for Internet business. Developers can directly monetise from user data. For example, NewsRepublic claimed in their privacy policy that: “We may share aggregated information that does not include Personal Information and we may otherwise disclose aggregated application usage information with third parties (including suppliers, advertisers, research, companies and other organizations) for industry analysis, demographic, profiling and other purposes.”. Some companies were well evaluated thanks to their user data when they were acquired. For example, some companies already had a good evaluation in IPO with their huge amount of user data, including Twitter³², Facebook³³, LinkedIn³⁴, etc. Facebook even paid \$1 billion in cash and shares to acquire Instagram for its user contributed photo data³⁵.

The second influence is to change the way developers think about their business model. Developers can provide free services to users to acquire user data and charge from enterprises or third parties who demand user data. Developers can also charge from premium users who

²⁷<https://www.microsoft.com/privacystatement/en-us/skype/default.aspx>

²⁸<http://toucharcade.com/2013/10/04/gamevil-just-bought-com2us-for-roughly-65m/>

²⁹<http://terms.withhive.com/terms/policy/view/M9>

³⁰<http://terms.withhive.com/terms/policy/view/M14>

³¹<http://terms.withhive.com/terms/policy/view/M14>

³²<http://www.cnbc.com/id/101103596>

³³<http://www.wsj.com/articles/SB10001424052702303448404577409923406193162>

³⁴<http://www.forbes.com/sites/georgeanders/2013/11/07/a-twitter-user-is-worth-110-facebooks-98-linkedin-93/>

³⁵<http://www.theguardian.com/commentisfree/cifamerica/2012/apr/09/facebook-instagram-1bn-storify>

demand user data service (there is generally less than 20% of premium users in an app, and 80% of users pay nothing). For example, *Kabam* confirmed in their company profile ³⁶ that: *“Unlike traditional console video games, Kabam games are available to play for free virtually anywhere gamers look for their entertainment. Kabam’s “freemium” business model enables players to access games at no cost or obligation, unlike console games that are expensive to buy or inconvenient to rent, and are played only on stationary and costly hardware devices that are upgraded about every seven years. players who enjoy Kabam games can pay for premium content in the game to enhance their playing experience. The average Kabam player is highly engaged, playing more than two hours a day.”*

The third influence is to have innovation on a business model. User data greatly expands the possibility for companies to make a profit. For example, *King Game* confirmed ³⁷ that *“At King, we design casual games with a broad appeal, which are easy to learn and challenging to master, allowing people to play for a moment, then move on with their day and pick up their game later. Our games are also synchronized across platforms, allowing players to switch seamlessly between devices and platforms and continue their game wherever they left off, so they can play anywhere, any time and on any device. For us, this is encapsulated in the idea of bitesize entertainment. We make our games available for free, while players can purchase virtual items priced relative to the entertainment value they provide. We embed social features in our games that enhance the player experience. User data enables them to create such good user experience, and also innovative features.”* Another example is *Talking Tom and Friends* app. The app was free to download as a way to expand its market popularity, which later contributed to earning a partnership with Disney ³⁸.

7.5.1 Monetisation on User Data

I now summarise how developers monetise with user data in their apps via three channels after I studied 100 featured developers. The summary is visualised in Figure A.27. The first monetisation channel is for users to pay for the app. It is traditional that users pay for what they have consumed. The difference is that most developers would provide an opportunity for users to trial the app first, and then encourage them to pay. This would make users happy as they only need to pay for what they need. Some developers sustain their business by only relying on a small percentage of users to pay and letting the majority of users use their products freely (Anderson, 2009). The second channel is to let advertisers pay for users. It is like the search service in *Google*. Ordinary users receive free search services, but advertisers need

³⁶<http://corporate.kabam.com/display/company/Company+Home>

³⁷<http://company.king.com/about-us.aspx>

³⁸<http://outfit7.com/2012/05/outfit7s-talking-friends-apps-to-star-in-disney-web-series/>

to pay for it. Developers make efforts to attract users and then sell advertisement positions in the app for revenue. Those advertisement positions in the app could be a small banner, product recommendation to users, or a soft production mention in the content, etc. The last channel is to let enterprises pay for users. For example, employees can use an app for free in personal contexts, but their employers have to pay for them in a business context. Developers can also sell aggregated information from their user data to enterprises who need it. For businesses in the crowd source sector, such as *Duolingo*, pay *Duolingo* for translation services, and *Duolingo* distributes the translation task into its large amount of users to collectively work on the translation job for free.

Most developers would adopt a mix of channels for their business model. For example, *Dropbox* mixes user payments and enterprises payments in the app. It provides advertisement-free online storage services. The first 2GB of storage is free, but would charge higher premiums for users who require more than 2GB storage space. *Dropbox* also charges for business accounts and provides services specifically to business users. Many games apps adopt a combination of user payments and advertising by providing free apps to users with an advertisement banner. Users can pay a small amount of money to remove the advertisement in the app. *WeatherBug* adopts a mix of advertisement payment and enterprise payment strategy. It provides free weather service to users, but puts an advertisement banner in the app. Users can report the mistake and correct the weather information and this correction enables developers to charge companies for its enterprise weather service. Some developers can use all three channels for monetisation. For example, *NBA Game Time* ³⁹ is an official app for the *NBA*. It provides in-app purchases for users to buy premium features like *NBA League Pass Premium Subscribers*. It provides advertisements for embedded content and also charges from team sponsors for users market insights. The stronger monetisation abilities, the stronger it is the competitive advantage and therefore the developer's competition performance may also be better.

7.5.2 Value Creation On User Data

User data helps developers to create value in their apps. User data is the fundamental component for an app. Users can express their demand in the app and developers can also monitor users' usage to figure out where and how to improve. Put simply, user data helps developers to create or have a new version for an app. Developers can use user data in marketing or to invite friends to join in the app. This would increase the app's user base. For most social networking apps, the more users joining in the app, the better value users enjoy. The contributions on new app, new versions, and a larger user base will, in return, contribute to monetisation. It brings the

³⁹ <https://play.google.com/store/apps/details?id=com.nbaimd.gametime.nba2011>

possibility that users can purchase more products, or that higher advertisement fees can be paid due to the broader exposure, or even that developers could increase charges for enterprises who demand higher quality of enterprise services. The model of value creation on user data has been represented in Figure A.28.

7.5.3 User Data Business Model Components

The business model is emerging as a new unit of analysis that emphasises a system-level, holistic approach to explain how firms “do business” (Zott et al., 2011). I have explored how to monetise based on user data and explained how value is created with user data. I then researched to understand how to make business model analysis with user data. Based on the framework developed by Morris et al. (2005), I listed the components that should be considered to analyse the user data business model in the app economy.

- **data source:** This component means understanding what user data are needed in the business. The data source may be either internally captured or externally integrated. Examples of internal data are customer support data, user contributed data including both human created (like birth location, memories, birthday, etc.) or system captured data (like GPS data), etc. Examples of external data are companied acquired data, customer externally provided data or data freely available from Internet. This component is additional to Morris et al. (2005)’s method.
- **data purpose:** This component shows what a developer can do with user data. Examples of usage can be: 1) for analytical purposes to predict the business future and provide evidence for decision making, 2) for service customisation purposes to enhance user experience, 3) for integration purposes to combine multiple data sources for value creation, 4) for complementary purposes to add additional levels of data accuracy or support for other core businesses, 5) for sharing purposes to widely spread the branding, 6) for storage purposes to provide storage service for users, 7) for advertising purposes to sell advertisement position, etc. This component is developed specifically into a user data context in comparison with Morris et al. (2005)’s method.
- **offering:** This component reveals what products the developers can offer based on user data. The offering could be data products, for example, providing some data to users that are demanded or difficult to fetch, to help users to manage the storage of their data, or to share data among users in demand. The offering could also be product knowledge, and that knowledge is normally extracted from data. For example, health conditions, fashion trend predictions, financial market sentiments, etc. The offering could also be another product. For example, providing other product ordering services or photo

printing services or even people matching services, etc. In comparison to [Morris et al. \(2005\)](#)'s method, the offering in user data business is generally a virtual service rather than a physical product.

- **target users:** This component provides an understanding of who should be targeted on by the developer. A typical classification is B2C (business to customer), B2B (business to business) and B2M (business to mixed). Each type would have a different influence on monetisation and marketing. This component is similar to [Morris et al. \(2005\)](#)'s method.
- **revenue source:** This component reveals how developers make profits. The revenue source can be 1) app purchases, 2) subscription fees, 3) premium fees, 4) third-party payments, 5) commission earnings, 6) advertisement fees, etc. User data business has generated many new monetisation models if comparing to [Morris et al. \(2005\)](#)'s method.
- **competency:** This component shows how developers can survive from the competition. Developer's need to research on whether the business idea may be innovative or not. Whether the app has a competitive advantage based on advanced technology or highly skilled human capital or established user community. This component is similar to [Morris et al. \(2005\)](#)'s method.

7.5.4 Business Model Analysis On 100 Featured Developers

I then studied three key components in the sample with 100 randomly selected featured developers and presented the results in Table [A.17](#). Those developers came from 19 different countries including developed and developing countries. The companies have also been established over 28 different years, from 1920 to 2012. Some of them were established a long time before *Android* was invented, and they were extending their services to the mobile world. Mobile apps enabled them to interact with users in an easier way to complement their core business. Some developers were growing up as *Android* was developing. A large percentage of those developers are game developers, and a common path for them has been to originally develop games for PC and video game consoles before *Android* and then to quickly move to mobile gaming to claim the market with their existing technology and human capital after *Android* was introduced. Other developers were established after *Android* became popular. Those developers are specially focused on mobile users and may also provide services on other access medias like PC. They are focused on mobile users, understood the potential of mobile market and always have an attractive idea before releasing an app. Their apps have always started with a simple function to solve user needs and then was then extended to other functions to complement other user needs.

The first component to study is *data source*. There are 22 featured developers' businesses purely relying on user data (see column *P*). In their apps, user contributed data are also the

major resource for users to consume, so a common strategy for them is to attract a large amount of users by encouraging users to benefit from the large amount of user contributed data. There are 80 featured developers trying to integrate user data with other data in their app to create value (see column *O*). Users may have difficulties to combine data together due to the technology challenge, or developers can help users to reduce the gap through integration services. It may be also impossible for users to look at other users' data directly for privacy issues, but developers can work as information brokers to provide insights from all users to each user without crossing the boundaries of privacy. There are 18 developers providing user data as a service. In this category, developers create tools to help users to manage their data, store their data and share their data on demand. There are 90 featured developers with social network services integrated into their apps (see column *M*). Users already leave large amounts of digital footprints in social networks like *Facebook*, *Twitter*, *LinkedIn*, *Google+*, etc. By integrating to those services, developers can obtain a user's data quicker and finish some jobs easily by including using user profiles on social networking sites to finish user registration, to predict user trend, to reuse existing tools for data storage and processing, etc. By providing API for external access, social networking sites can acquire more users from those developers, and also enhance the lock-in effects for users to choose as a social identifier (Egelman, 2013). Therefore, it is a mutual benefit for developers to cooperate.

The second component is *data purpose*. There are 58 featured developers using apps to complement other core businesses (see column *H*). It can be a complement to an existing large user community, a software tool or other services. It is worth mentioning that mobile apps have been popularly used by TV companies for user interaction during TV broadcasting in the sample. There are 53 featured developers having competitive resources (see column *I*). Examples of competitive resources include unique resources like copyright, unique position like affiliation or franchise, unique ability like advanced technology, etc. Developers who have competitive advantages attract users more easily due to their unique and strong competitive ability. There are 84 featured developers who continue what they did on their first app (see column *J*). Developers want to explore new domains once they have established their positions in the market. Having more user data increases the ability and possibility for diversification. However, this may not be reliable as some developers were on the way to diversification during my observation which was not captured due to the effects of time change. There are 31 featured developers providing API to other developers and users (see column *K*). Opening an API enables other developers to access user data and services in their own apps. This open strategy would attract other developers to create complementaries to its own property product. This would benefit users with a variety choices and would improve usability. As mentioned before, other developers will also contribute with new users to API providers, especially for social

network site. There are 50 featured developers who enable users to access their data offline. Enabling offline brings the difficulty on how to synchronise data between online and offline systems or among users. However, offline features provide a better seamless user experience, as the mobile Internet is not always available and mobile data services are not cheap.

The third component is *revenue source*. There are 21 featured developers heavily relying on the advertisement banner (see column *D*) and 33 featured developers relying on recommendations in an app's content (see column *E*). Those developers would motivate users to continually use the app to increase the chances for the advertisement to be seen. There are 47 featured developers doing in-app billing (see column *F*) and 47 featured developers providing free trial versions in addition to a paid version (see column *G*). Developers who generate revenue from subscriptions may not find it necessary to pay in the app. Users already paid for a subscription from somewhere else, and log into the app to continue to use those services. However, as the use of mobile payments like *Apple Pay*⁴⁰ or *Google Wallet*⁴¹ increases, users feel it is easier to pay for subscriptions on mobiles, therefore, more and more subscription developers have begun to enable users to pay in their apps.

There are some other lessons that have not been listed in the table. For example, sharing features would always bring a larger user base. Recommendations and personalisation increase the user experience and better granularity is always welcome to users. Communication features between users increases the user active time in the app. Social integration motivates users to contribute with their data and to continually use the app. The increasing interest on the user base and functionality generally contributes to the possibility and to the increase in revenue.

7.6 Why Does The Developer Demand The Same User Data In Diversification?

Different functionalities require different user data. However, some developers adopt diversification strategies but only demand on the same types of user data. It is interesting because diversification generally requires different types of resources to differentiate or innovate different functionality. I filtered out all such developers from all featured developers and then studied them in Table A.18. There are 94 developers in total from all featured developers. The number of apps ranges from 2 to 32, and in totally there are 509 apps.

40 of 94 featured developers have all their apps in the same category. Those developers create new apps exclusively in the domain they are already familiar with. Those developers are

⁴⁰<https://www.apple.com/apple-pay/>

⁴¹<https://www.google.com/wallet/>

commonly in the gaming or entertainment categories, for example puzzles, music, education, travel, etc. In those categories, it is easier to duplicate apps with different content.

90 of 94 featured developers demand *personal* user data in their apps. 78 of 94 featured developers demand *system* user data in their apps. Only 9 of 94 featured developers demand *social* user data in their apps. This unbalanced distribution on user data demand may be caused by the method of innovation they adopt. Most developers differentiate on the content in an app, rather than on the functionality of an app. Therefore, developers would require more *personal* and *system* user data but less *social* user data.

7.6.1 Developers Do Not Use User Data

There are 2 developers who do not demand any user data in their apps, as they did not use any permission. Both of them are independent developers and not employed by any commercial company. *Sarah Northway*⁴² is a couple who travels around the world and have developed 4 mobile games across PC, *iOS* and *Android* platform. In *Google Play Store*, 2 of their games are paid apps and the other 2 games are free apps. They are using paid strategy. 4 games cover three categories and each game is quite different from the other. *RabugenTom* is an individual developer having a PhD in Physics qualification. He has two apps doing exactly the same function on music chord, but one app is free with limited functionality and the other one is paid with all functions. He is using a freemium strategy. Both developers received high user reviews.

7.6.2 A Story Of An Independent Developer

I emailed to interview *RabugenTom* and he told me about his experience as an individual developer:

“I thought about the app in early 2010. I wanted to make apps mostly by curiosity. At this time, the offer was scarce. There were a few apps, but they were very poorly executed.”

“To make a chord fingerings app, you need fingerings. I was looking for a database of them on the Internet, but I didn’t find one. But these fingerings don’t come from nowhere. They follow more or less clear musical and physical principles. So I came up with the idea of computing them since I wasn’t able to find an existing list. Here lies the real innovation of the app. It opened a lot of new possibilities. Some of them are in the app right now, but other existing ones are still to be developed.

⁴²<http://northwaygames.com/>

So it's initial weakness (the lack of database) was turned into it's main strength (it doesn't need one)."

"I published the app in July 2010, after 3 months of development, and 6 months later, I made a huge update which outlines the current version. In February 2011, I was contacted by Google. I made a tablet version and was invited to present the app during Google I/O as a sandbox developer. In January 2013, I made another huge update to the app and started developing the iOS version. It took me one year to do the iPhone version and almost the same time to do the iPad update (with a lot of new features). I'm still working on the app. I'll get back soon to the Android version to modernize it a lot."

"About user data: if you're speaking about non app specific user data, I didn't use it. I had just set up some basic analytics (Google Analytics) to have a few stress metrics in case I want to set up some kind of backend service. I did not retrieve any personal information. I did not check user engagement or acquisition. The only data which matter to me are the installed number of users and the current number of them running the app at a given time. There are no social stuffs yet within the apps."

"I'm still working on the app, trying to make it better. I think that it lives in some kind of super niche (guitar players) and there is still a lot of them to reach. That's why I'm not trying to raise the revenue/user right now. It's a full time occupation. I have a lot of other ideas, but I lack the time to work on them."

"For info, I have a PhD in Theoretical Physics. I learnt all I know about programming by myself, on the Internet. Life circumstances had brought me in a very remote place of Brazil, without any possibility to continue academic research, so I started coding like that. I did not have any contact at Google and Apple at this time."

"I would rather say Artisan style Entrepreneurship. I need and want to communicate more. My thesis advisor was of the kind (speak only if you have something relevant to say) and I'm kind like that now. Exposing myself doesn't bring something to the app nor to the user experience. The relevant bits take some time which I don't have as much as I want. I have a lot of ideas or technical stuff to publish, but again, I don't have the time to do so. I also need to write in English which is not my first language and tends to slow things down. I think I'm working more or less like a novel writer would do. There are a lot of developers like me in the market."

“All my revenue came from my paid apps. On Android, the free version has advertisement but I’m using them as in-house ad for the paid app. The paid app conversion from these in house Ads seems to earn me slightly more than the paid advertisements, but it’s very dependent from the number of users and their engagement. I think I earn as much as I would do if doing academic research in France (where I’m from). This is much less than what a developer working for a company or studio would earn in the US.”

“About the life circumstances, a few months after defending my thesis, I’ve met a Brazilian girl, and I’ve followed her from France to the middle of Brazil (she can’t work outside Brazil). At this time, I was looking for a regular job I could do with my diplomas. I was in the middle of nowhere with a 13 inch computer with 1MB Internet. I started to learn Java and then made my first app, mainly to have something to do. We’re still living in Brazil.”

“Both times I was featured, it was only because of the intrinsic quality of the app. I didn’t have anyone to contact at Google or Apple when I submitted the app to raise their attention about it.”

“I don’t plan to recruit anyone right now, but it may change in a few months. The PhD itself didn’t bring anything, but being able to solve problems by yourself is invaluable.”

7.6.3 Developers Who Only Use *system* User Data

There are 2 developers who only use *system* user data and not the other two types of user data. It is interesting because they did not use *personal* data as it is commonly used in apps for user identification and service customisation. *Hulu* has two apps in the *Google Play Store* and both of them are free. *Hulu* is an American online company providing TV shows, clips and movies. Users can subscribe to *Hulu* for high quality video content or enjoy free general content with advertisements. The first app is a mobile interface for users to complement its core service on the desktop platform. The second app is doing the same function as the first app, but is specifically designed for the Japanese market. Users have to login first in order to use the services, and *Hulu* did user registration on desktop rather than mobile. *Hulu* disclosed in its privacy policy that ⁴³ *Hulu* is using social network data but mainly integrating from social network website like *Facebook*. This integration is done when users register online and not from a mobile app. *Hulu* uses *system* user data to understand the distribution of users and even the pattern of user behaviour.

⁴³<http://www.hulu.com/privacy>

Storm8 Games is the other developer only using *system* user data. It has 10 games apps and all of them are free with in-app purchase options. 7 out of 10 apps are in the *Arcade* category. Half of its game apps are designed for offline playing and the rest is encouraging users to play for competition. However, the social networking features are done by the *Google Store Service API* which does not require additional *social* user data by developers. The *system* user data requested by *Storm8 Games* is mainly focused on accelerating the graphic speed and improving user experience. *Storm8 Games* listens a lot to user feedback, and users always review their apps with high scores. One of its reviews said “*Thank you for rewriting the notifications. They are so much better. I like that you added a count indicator telling how many babies are needed left to save towards the end. Now if you could just add a music switch in the settings, this will be the perfect game.*”. In June 2011, *Storm8 Games* was one of the first free-to-play mobile game publishers to announce that they had earned \$1 million dollars in a single day ⁴⁴.

7.6.4 Developers Using All Types Of User Data

There are 9 featured developers using all types of user data in their apps. *MediSafe Project* ⁴⁵ is an app to remind people to take pills on time. The motivation for developing is that “*MediSafe’s co-founding brothers started the company when their diabetic father accidentally double-dosed on insulin and suffered an emergency. They understood if they were more involved in reviewing their father’s daily medication and insulin regimens, it could change his behaviour for the better and help him get healthier.*” Both apps are completely free, but they charged from enterprises “*The MediSafe Enterprise Analytics Dashboard provides a menu of standard and custom reports which include medication usage, adherence, effectiveness, geo location, population management and more (anonymous). This gives you real time insights into your customers which can drive marketing programs more broadly than just adherence. MediSafe’s unique combination of data, algorithms and analytics helps you identify medication consumption problems as they happen in real time. From this you can develop actionable solutions that ultimately drive enhanced outcomes. You can develop and deliver individualised messages at the right time to change behaviour and positively impact appropriate consumption.*”

The other 8 developers are using all types of user data because of their sociable features and their diversification strategy. For example, *textPlus* has 2 apps with in-app purchase options and social data allows users to send messages or call their friends. *Twitter* has an app for tweeting and another one doing the same function but specifically designed for TV content. *theScore* has 2 free apps, one for online sport gambling and another specific for football news. *TripIt* has 2 apps, one is free with advertisement and the other is a paid app that provides the same

⁴⁴<http://www.gamezebo.com/2011/06/10/storm8-has-their-first-million-dollar-day/>

⁴⁵<http://www.medisafe.com/>

functions as the free one. *BiCore* has 3 free game apps and all of them have in-app billing options. *Dropbox* has 3 free apps, one app runs the core business that *Dropbox* initially started and the other free apps are complement to the core business with extended functionalities. *Evernote* uses the same strategy as *Dropbox*, so one app is running the core business with in-app billing options, and other two apps are completely free for user sharing content as a complement to the core app. *SwiftKey* is a tool for typing, so the major revenue mainly comes from the selling of technology. *SwiftKey* apps were not free originally, but this strategy changed in later 2014 to make every one of its apps free ⁴⁶.

It can be summarised why developers want to diversify using the same types of user data. One motivation is that they want to target the same types of users but in different categories. This is mainly to increase the economic scale. There are many reasons for targeting the same types of users in different categories. For example, *Hulu* has to develop a separate app specifically for Japanese users due to the restriction of copyright. Another motivation is to collect as much as possible for a target type of user data. For example, *MediSafe* would like to generate user medication behaviour data as much as possible, so they develop apps for different scenarios but all collected data are used to charge for the same enterprise clients. The other motivation is to have as many duplicate apps as possible. For example, developers would use the same technology to create an ebook so their users data demands could also be same.

7.7 Why Do Developers Not Diversify?

[Lim and Bentley \(2012b\)](#): Studies of individual runs show that although S3 Copycat dominates the top developers lists, individual developers from other strategies can become the most successful developer in the ecosystem, achieving either the highest number of downloads or the highest average downloads per app. However, the occurrence is by chance: the developer has to build the right app at the right time (e.g., the app has features preferred by many users, appears on the New Apps Chart, is downloaded by many users, appears on the Top Apps Chart, and continues to attract downloads). Developers with high Top20AvgDI tend to develop one app in the entire three years. This shows that it is difficult to repeat success (a fact well-known to app developers in the real world¹⁶). The more apps a developer builds, the lower the average downloads tends to be. As such, the same developers rarely appear on both average and top downloads lists. In many runs, we observed that developers who built many apps can have the highest total number of downloads, but often have a low average.

It is also interesting to study why developers do not want to diversify their offering of apps to the market, by only working on one app. The strategy for them is to continually focus,

⁴⁶<http://swiftkey.com/en/blog/why-we-set-swiftkey-free/>

improve and maintain only one app. There are 94 such developers from all featured developers with only one app and I presented the result in Table A.19. There are 15 out of 94 featured developers (see column A) who were still in the early stage of their business during my analysis, as it is not easy to find out their business model and how they planed their future development. A common strategy for those early stage developers is to focus on user base growth.

There are 20 out of 94 featured developers (see column B) working on games. Users can switch between game apps quite easily with short longevity, so game developers commonly have new game apps or new features to maintain their user's attention. Therefore, it is interesting to understand why those developers only want to focus on one app. 8 out of 20 developers have a competitive advantage to work on multiple platforms. It means their game can be played and battled across different platform, including *iOS*, *Android* or even *XBox*. This platform crossover feature really extends user coverage. Users can play the game or battle with their friends without restriction on any device hardware. Developers can then focus on how to improve the app content for play. This competitive advantage is not easy to acquire, and different platforms require different types of technology development skills. 5 out of 20 such developers are independent individuals. They have limited resources compared to company developers and they want to show them off with the game apps they have developed. Some of them are hackers, for example, *Dejobaan Games*⁴⁷ who was invited for talks many times to talk about their development experience. In addition, there is one game developer using *social* user data, who developed "Football Manager" style games.

74 out of 94 featured developers are working on non-game apps. 15 out of 74 developers have their apps to complement other core businesses outside of mobiles. So their priority is still in their core businesses in somewhere else. Among them, 4 out of 15 developers are using *social* user data for complementary purpose. 20 out of 74 developers have advanced their technologies. For example, *FxGuru* has an easy-to-use mobile movie effect editor, *Picadelic* is a leading developer of innovative online and mobile image based products. 3 out of 20 developers use *social* user data. 34 developers have a large user community and many of them (20 of 34) use *social* user data. This also drives the fact that developers using *social* user data are overall much more common than the list of developers in previous groups.

The focusing strategy rather than diversification enables developers to spend more resources on a single app with better quality. Some developers have a large dedicated business to focus on, but some developers work as agencies to continually change their business by reusing their technological skills. This is a main reason for developers to diversify or not. For example, *Digital Legend* developed a famous game engine to support multiple platform development. They have to continue to support their technology, which does not allow them to diversify.

⁴⁷<http://www.dejobaan.com/>

Developers using focusing strategy always have apps with greater longevity compared to developers using the diversification strategy. They can profit enough from a single app and the market for this single app is also huge. Focusing enhances their market position to compete for a greater user base. Cost is another issue. The cost of having new users to install and use the app becomes more and more expensive due to the rise in competition ⁴⁸. Thus, focusing on one app concretes spending and save costs for user acquisition.

7.8 Summary Of Qualitative Study

App permission is the key for developers to implement certain functions. Combing different permissions would enable developers to innovate on unique functionalities. Therefore, developers can learn from each other on how to implement functionality via observing how they use permissions. However, using too many permissions would increase the power consumption in an app and reduce the user experience. Therefore, developers are motivated to optimise user permission.

The reason for developers to use permission is for acquiring user data. User data plays as an important resource in the app economy. Developers want to acquire enough user data to create more value for their users. In this section, I have classified user data into three types, including *personal* user data, *social* user data and *system* user data. The study found that *personal* user data is mainly common, as it is the key for user identification and service customisation. *System* user data is also popularly demanded to provide optimised app performance or to be key for certain advanced technologies. *Social* user data is less commonly demanded, due to the technology complexity and privacy risks. However, successfully using *Social* user data brings great benefits for marketing and social networking which would increase the user base.

The priority for developers is to increase their user base in the app economy. Monetisation in the app economy is heavily reliant upon user data and user base. Therefore, developers are motivated to acquire more and of better quality user data as well as to innovate to encourage users to continue using apps to generate more user data. Developers should not only focus on the technology and product innovation, but also on business model innovation to compete in such a competitive app market.

I have improved the traditional business model analysis by specifically fitting it into a user data context. I then applied it to study an example of randomly selected 100 featured developers. I found that developers should adapt their strategy to collaborate with and learn from other developers rather than to only treat them as competitors. I also studied every featured developer

⁴⁸<http://venturebeat.com/2014/06/26/9000-apps-and-397m-users-show-cost-per-install-is-the-worst-mobile-user-acquisition-method/>

adopting a diversification strategy but demanding the same types of user data. They diversify to target the same types of users but are distributed in different categories due to many reasons, such as boundaries on languages, demography and content. Concentrating on same types of user data with diversification would let them increase the economy scope. The last group of developers I studied shows featured developers who only have one app. They focus on a single app rather than on diversifying because they want to concentrate all of their resources on the longevity of the product because they want to demonstrate their ability by deeply developing a product. Most featured independent developers only have one app.

This chapter has explored how developers compete with app permissions, user data and business models. In this context, the featured developers' competitive advantage can influence other developers to learn and mimic. It provided evidence that developers are competing against each other, but also are aware of each other. However, it is still not clear how developers signal their competitive advantage for selection and what is the impact of such selection. The next chapter will quantitatively analyse the selection and market performance regarding the developers' signalling behaviour.

Chapter 8

Quantitative Analysis

In Chapter 5, I have developed the model to describe how developers would signal to *Google* for selection and signal to other good developers for peer recognition. In order for a receiver to identify the sender's quality, developers have to make some costly actions to transform their quality into observable signals. The Literature (Chapter 4) and Qualitative Analysis (Chapter 7) chapters have demonstrated that featured developers can demonstrate their innovation ability via some unique costly actions. This chapter continues to analyse how developers signal for platform selection with three particular quantified signals and how the market share is impacted by their signalling behaviour.

To achieve this goal, two questions are designed for statistical test in this chapter:

- **Question 1:** what constitutes a signal for platform selection?
- **Question 2:** how does market performance respond to peer recognition?

8.1 Comparison Groups

Both questions study the relationship between different parties in the *Google Play Store*, so the first job is to be clear about what to compare. One app can compete with other apps created by the same developer for resource allocation or compete with apps by other good developers for market recognition or even compete with other apps having a very similar functionality. In order to study these competition relationships in the app economy, three comparison groups have been identified. Each group is to understand the motivation and performance of developers' strategies in a particular competition context. Featuring is a key state for developers to compete for platform selection. So I am primarily focusing on apps by *featured* developers and they are the **target** app. In other words, the **target** app is the app released by a *featured* developer and the *app_id* of a **target** app is the primary key to link all three groups.

For each **target** app, the comparison groups are selected with specific criteria. In first comparison group, I discovered their **target** app's *developer_id* and then selected every app by this *developer_id*. This group is named as **dvlp** which shows for comparing every apps by the same developer. In second group, I found that **target** app's *app_cate* (category) and then selected every apps created by *featured* developers in that *app_cate*. This group is named as **cat**, which represents every app by all *featured* developers in that particular *app_cate*. The third group selects every recommended app to the **target** app. This group is called **funct** which represents apps very similar to the **target** app. It should be noted that the apps in the third group can share same or different *developer_id* and *app_cate*.

In its web version, the *Google Play Store* recommended no more than 16 apps similar to a particular app. Figure A.10 is a screenshot of an app recommendation for a particular app and it shows four apps recommended as similar apps to that particular **target** app in the *Google Play Store*. By actually clicking the button, users can see no more than 16 similar apps recommended to each **target** app. These similar apps are identified by *Google* with a complex algorithm and would be heavily used as recommendation in the *Google Play Store*. By listing them together, users can easily compare them and this increases the competition among similar apps. This information has been used to make the *funct* group, which is the third comparison group.

The first comparison group compares how a developer distributes resources to his apps. The second comparison group sees how *featured* developers are competing to and learning from other good developers. The third comparison group understands how apps compete when they are very similar. Three groups can be studied in isolation, but can also be combined together to gain a comprehensive reasoning for actions taken by developers, as each comparison group provides an insight from a particular angle. For example, if a *featured* developer diversified their product lines (insight from *dvlp* group), is it because they are facing the high competition from other *featured* developers working in some product market (insight from *cat* group), or is it because they learned from similar apps to expand its boundaries into other markets (insight from *funct* group)? The specific combinations in this study are introduced later when I discuss on analysis.

Once a comparison group is selected, every app in the group is aligned according to its *release_date* to obtain an order (*sequence*) for each app in that particular group. This *sequence* represents the market entry order of an app and also for the developer (see illustration in Figure A.29b). I would then compute the days difference (*days_lapsed*) on *release_date* between any app and any app immediately prior to it in the group (see t_i in Figure A.29a). Each measurement in each comparison group is discussed in details in next subsection. Remember,

each group contains many apps to the **target** app and every app in the group is ordered from 1 to n according to its release time. So each comparison group can be represented as a list like:

$$apps = [app_1, app_2, \dots, app_i, \dots, app_n] \quad (8.1)$$

8.2 Platform Selection

Platform selection is a change of status from a non-featured developer into a featured developer. So far, there are only 365 developers selected by *Google* to obtain this status change. The competition for selection is extremely high, as there are 232,397 developers in total in my collected data set. It means the probability for selection is only 0.15% ($365/232397=0.15\%$). However, once selected, developers would stand out from the crowd with such a selection and enjoy many benefits with this special “label”. Each developer would try their best to signal to *Google* for recognition on their ability.

Google wants to select good developers with real and high qualities. Such a large number of apps in the market would increase users’ search costs finding their desired apps. In such a crowded market, many apps may look similar to each other. Users may have to try several apps before finding the exact app they require (Boudreau, 2012). More importantly, not every app has a guarantee of high quality (Chia et al., 2012). Some low quality apps would damage the user experience and impact on the motivation to continually and actively use the *Android* platform. This negative impact is extremely true due to the existence of malicious apps (Sarma et al., 2012). Good developers are interested more in innovation and long term benefits. Lower quality developers are more interested in short term benefit, so rather than innovate, they would like to copy existing popular products to earn “quick money” and this copycat action increases the crowding (Yadong Luo et al., 2011). This crowdedness would damage the financial return to high quality developers’ investment on innovation, which would discourage developers’ motivation for innovation. In the long term, this would damage the health of the ecosystem. A market operator like *Google* has a responsibility to protect developers’ innovations and make sure that their investment on innovation could get fair returns.

Google needs to be very careful on selection. *Google*, as the market operator, needs to maintain fair competition, which is to make sure the investment on innovation can get the expected return. Selection is a kind of guarantee for investment return on innovation. Only developers themselves know their real quality, but all developers “pretend” to tell the public about their “high” quality. Due to the information asymmetry, *Google* has to “guess” their real quality, which makes the selection a difficult job. Improper selection would discourage the motivation of developers for having real quality. Meanwhile, “Top Developer” is a life-time

award. So *Google* needs to observe and monitor developers for enough time before building enough confidence in them to make a decision on selection.

Therefore, the question is on how *Google* observes developers to make selection decisions. Put another way, the question is on how developers act to signal to *Google* for selection. In other words, the model is built to answer what constitutes a signal for selection.

8.2.1 Dependant Variables

In the chapter on Data Collection (see Chapter 6.1.5), I explained how to understand which app enabled a developer to obtain their selection status by using an algorithm to compare the date of release for each one of their apps on historically archived web pages (see Appendix A.2). After this work, each app gains a variable called *status* to represent whether the app was released before this developer got selected or not. It is a binary data. If the app was released before selection (or before the developer was featured), the app's *status* is 0. If the app was released after selection (or after the developer was featured), the app's *status* is 1. It is easy to image that every non-featured developers' apps' *status* is 0. Since *status* records a developer's transition, it is adopted as the dependent variable for selection prediction.

Due to information asymmetry, *Google* has to interpret a developer's real innovation quality by observing their actions. These actions are costly to developers, but in fact a reduced opportunity cost is good for developers (Connelly et al., 2011). Therefore, high quality developers are willing to make such costly actions but not low quality developers, as they have to pay much more to act. These actions are only done by high quality developers, who are observed by *Google* to interpret their innovation abilities. From the Qualitative Analysis chapter (see Chapter 7), I have identified good developers' distinct actions for three qualities, including uniqueness, following and diversification. These three qualities became the independent variables to predict the probability for selection.

8.2.2 Uniqueness

I discussed in Chapter 7.1 that app permission is the key for app functionality and for acquiring user data. Combing permission in a unique way can create unique functionalities in the app and create competitive advantage. Innovativeness can be demonstrated in a proxy of the unique use of app permission. Therefore, the first distinct action is for good developers to uniquely use app permission.

The way to calculate uniqueness in this chapter is developed from the Levenshtein distance (Levenshtein, 1966). The Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the

minimum number of single-character edits (i.e. insertion, deletion or substitution) required to change one word into the other. Therefore, it is also sometimes called edit distance. It has been widely used (Heeringa, 2004) to compare the similarity between two objects.

Uniqueness is calculated based on the distinction of permission lists requested by two apps. Each app requests a list of permission when users install it. The *Google Play Store* disclosed this information in its web version (see Figure A.19) and I scraped it in my data collection. Uniqueness measures how permissions are used distinctly between two apps. So by comparing two apps, it is possible to filter out the permissions commonly used in both apps, which shows the overlap of permission by these two apps. The other permissions are either only used by the first app or only used by the second app, which means they are unique to an app. In other words, the *unique* is the portion of distinctly used permissions in every permission owned by two apps (see Equation (8.2)). An example of calculation is demonstrated in (8.1). The equation enables the *unique* to be normalised in a range between 0 and 1. A special case would be when both apps are not using permissions at all, then the *unique* would be 0 as they have the same permissions in use. This method is not exactly the same as the original Levenshtein method, as the order for “character” in a “string” is not important. So, there aren’t any substitutions, but only insertion and deletion operations to change from a “word” to another “word”. This method is more closer to the solution for the longest common subsequence problem (Majumdar and Nechaev, 2005).

$$count_{pc} = count(perms_comm) \quad (8.2a)$$

$$count_{ua} = count(unique_a) \quad (8.2b)$$

$$count_{ub} = count(unique_b) \quad (8.2c)$$

$$count_{ab} = count_{pc} + count_{ua} + count_{ub} \quad (8.2d)$$

$$unique(app_a, app_b) = \begin{cases} 0 & \text{if } count_{ab} = 0 \\ \frac{count_{pc}}{count_{pc} + count_{ua} + count_{ub}} & \text{otherwise} \end{cases} \quad (8.2e)$$

$$unique_i = \begin{cases} 0 & \text{if } i = 1 \\ unique(app_{i-1}, app_i) & \text{if } i > 1 \end{cases} \quad (8.3)$$

Based on this method, *unique* can be calculated for any two consecutive apps that are ranked by release date for each comparison group. The equation can be found in Equation (8.3). A special case is to calculate the score for the first app in the sequence. Even if there is no app to

$$\begin{aligned}
app_a &\Rightarrow perms_a = [perm_id_1, perm_id_2, perm_id_3] \\
app_b &\Rightarrow perms_b = [perm_id_1, perm_id_3, perm_id_4, perm_id_5] \\
&\quad \nabla \\
perms_comm &= [perm_id_1, perm_id_3] \\
unique_a &= [perm_id_2] \\
unique_b &= [perm_id_4, perm_id_5] \\
&\quad \nabla \\
count_{pc} &= count(perms_comm) = 2 \\
count_{ua} &= count(unique_a) = 1 \\
count_{ub} &= count(unique_b) = 2 \\
&\quad \nabla \\
unique_{ab} &= unique(app_a, app_b) \\
&= (count_{ua} + count_{ub}) / (count_{pc} + count_{ua} + count_{ub}) \\
&= (2 + 2) / (2 + 1 + 2) \\
&= 0.6
\end{aligned}$$

Table 8.1 An illustration of how *unique* is calculated for two apps

compare to, the *unique* value is set as 1 for consistence for the first app. Uniqueness is a relative term that depends on what to compare. The comparison groups represent three dimensions of competition. Therefore, I have calculated *unique* for each group and they are *unique_dvlp*, *unique_cat* and *unique_funct*. To visualise the change of uniqueness for apps in each category, I have created two figures to represent every apps by featured developers in game categories (see Figure A.25) and non-game categories (see Figure A.24).

This uniqueness is costly to act. I have discussed before that using permission requires technical skills. There are two kinds of uniqueness. The first kind of uniqueness means using newly introduced permission. When a new permission is introduced into *Android* SDK, not many developers can figure out how to use it, so its first use is unique to other developers who have no ability. The second kind of uniqueness is uniquely combining permission in a way that other developers have not tried before. This uniqueness requires developers to have strong insights and then to generate innovative ideas on how to use the permission differently. Both kinds of uniqueness require developers to recruit highly skilled employees to learn, understand and monitor permission technology development. This normally leads to the higher operation cost of a company.

However, uniqueness is not guaranteed for business success. Using permission differently may produce bugs and unstable function in an app. It may also lead to the unexpected mobile power consumption or the slowing of operations. Uniqueness should not generate unnecessary waste or complexity. For example, if developers may uniquely implement some functionalities in a complex way and other existing methods can implement it in an easier way, then this is wasteful and brings about unnecessary complexity.

In summary, uniqueness makes the app different from existing apps to gain a competitive advantage, but the cost associated with uniqueness would also bring the negative impacts on company operation and app performance. Therefore, the relationship between uniqueness and selection should be curvilinear. For each comparison group, a hypothesis is proposed as below. A signal is generated during the development of an app and transmits this signal during the app release. It requires technical skills to interpret this signal, and *Google* has these permission data to analyse in the *Google Play Store*.

Hypothesis 1 *In comparison to a developer's previous apps, the relationship between app uniqueness and platform selection is curvilinear (inverted U-shaped), with the highest probability for platform selection occurring at an intermediate level of uniqueness. Specifically, a developer is more likely to acquire a featured status at medium levels of uniquely use of permissions, and decreases at either the lower or higher levels of uniquely use of permissions.*

Hypothesis 2 *Comparing to prior apps by a featured developer from same app category, the relationship between app uniqueness and platform selection is curvilinear (inverted U-shaped), with highest probability for platform selection occurring at an intermediate level of uniqueness. Specifically, a developer is more likely to acquire a featured status at medium levels of uniquely using permissions, and such likelihood decreases at either lower or higher levels of uniquely using permissions.*

Hypothesis 3 *In comparison to previous apps with the same functionality, the relationship between app uniqueness and platform selection is curvilinear (inverted U-shaped), with highest probability for platform selection occurring at an intermediate level of uniqueness. Specifically, a developer is more likely to acquire a featured status at medium levels of uniquely use of permissions, and such likelihood decreases at either lower or higher levels of unique use of permissions.*

8.2.3 Diversification

[Lim and Bentley \(2012b\)](#): Studies of individual runs show that although S3 Copycat dominates the top developers lists, individual developers from other strategies can become the most

successful developer in the ecosystem, achieving either the highest number of downloads or the highest average downloads per app. However, the occurrence is by chance: the developer has to build the right app at the right time (e.g., the app has features preferred by many users, appears on the New Apps Chart, is downloaded by many users, appears on the Top Apps Chart, and continues to attract downloads). Developers with high "Top20AvgDI" tend to develop one app every three years. This shows that it is difficult to repeat success (a fact well-known to app developers in the real world). The more apps a developer builds, the lower the average downloads tend to be. As such, the same developers rarely appear on both the average and top downloads lists. In many runs, we observe that developers who built many apps can have the highest total number of downloads, but often have a low average.

The second distinct action is to diversify. In economics and marketing, diversification is the process of distinguishing a product or service from others to make it more attractive to a particular target market ([Matsusaka, 2001](#)). This involves differentiating the product from competitors' products as well as the firm's other own products. Resource endowments ([Barney, 1991](#)) allow firms to be different. It reduces competition and makes it possible to reach new segments of the market. Thus, diversification is the process of distinguishing the differences of a product or offering from others to make it attractive to a particular target market ([Kotler, 2000](#)).

Diversification is an ability. Developers diversify to cover more users. It would bring users a seamless experience by providing apps in multiple categories. For example, *Facebook* users can chat and also play games with their online friends. Diversification generally requires developers to have staff working on different product categories, to satisfy different types of users, and also to innovate on different functionalities. Having more apps would increase the exposure rate and the chance to be searched by users. This can be considered as an indicator for high productivity, which would positively contribute to the reputation and branding of the app. Diversification is also a strategy to battle in a rapidly changing business environment by diversifying the company risk into multiple apps. If one app fails in the market, other apps may still be able to operate to cover the failure of that app.

Diversification is costly. Developers have to spend more resources to maintain the apps. For example, developers have to recruit more staff, spend more on user data storage and analysis and allocate more resources on marketing and communication. Diversification is not simply a process of copy and paste or duplication. Having more apps simply means having more competitors and developers have to keep their innovative ability to flight against competitors. In comparison with the focusing strategy, diversification may not allow developers to continually exploit in a single product which may allow a particular app gain a deeper market position.

I have created two visualisations on how developers diversify in app categories. The network in Figure A.22 shows that a large portion of developers are using a diversification strategy by producing apps in multiple categories and this clustering is more obvious in game related categories. The reason may possibly be because game sector is easier for app diversification, and the investment return may be more guaranteed as it has a very large user base. The chart in Figure A.23 highlights the distribution of diversification and focusing strategies. In the *Google Play Store*, the number of apps (*blue line*) in each category is normally higher than the number of developers (*green line*), and this ratio (*red dashed line*) is extremely high in certain categories like *Casual* in *GAMES* and *Book & Reference* in *APPS*. It partially proves that some developers prefer to focus on a small number of apps and some developers prefer to have a large number of apps to diversify the market opportunity.

Diversification is measured by the number of distinct categories in statistic that developer has already created apps for. The variable is called *scope*. With the above analysis, I would predict that the relationship between diversification and selection should be curvilinear and the hypothesis is:

Hypothesis 4 *The relationship between diversification and platform selection is curvilinear (U-shaped), with lowest probability for platform selection occurring at an intermediate level of diversification. Specifically, a developer is less likely to acquire a featured status at medium levels of diversification by adopting a focusing strategy in the early stages or a diversification strategy after the business has been established to increase the chance of acquiring a featured status.*

8.2.4 Following

The third quality is based upon how rapidly a follow on app can occur. When an app is already introduced into the market, how quickly a developer can create a new app to follow that app in the market is an interesting question. If a developer follows quickly, his new app still obtains a lot of first move advantage, but it may have less time dedicated to app creation compared to the original app, so the quality may not be guaranteed. If a developer follows the market slowly, his new app should take enough time to achieve a good app quality, but it may lose the benefit of early entry. Therefore, an optimistic following speed is important for strategic innovation.

Following has been studied by organisational learning theory as search timing on innovation in existing literature (Katila and Chen, 2008). Search is a problem-solving process in which organisations recombine, relocate and manipulate existing knowledge to create new knowledge (Nelson and Winter, 2009). Innovation search takes place in a knowledge pool by sampling opportunities from the pool of technological possibilities. This pool of knowledge has two

distinct types. One is knowledge familiar to the searching firm (its current knowledge base) and the other is knowledge that is new to it (Katila and Ahuja, 2002). The firm may introduce new products because it actively explores an entirely new area that departs from its current knowledge base or because it productively translates knowledge from its existing knowledge base into new products. Being in a highly competitive environment, innovation search has an impact on learning and competition.

Competitors can influence the search of the focal firm through learning. The firm can learn from its competitors and the competitors can learn from the firm. For example, in competitive industries, large firms in particular often adopt a “fast second” strategy to commercialise the discoveries of their smaller competitors (Markides and Geroski, 2004). Firms also observe a variety of R&D attempts by their competitors to link their experiences to consequences (Katila and Ahuja, 2002). Overall, the implication is that the **target** firm learns what works and what does not by observing the results of their competitors searches, with a likely positive effect on the frequency of the *target* firm’s introductions of new products. On the downside, competitors also learn from the *target* firm. They typically start searching in the same area and crowd it. The area quickly becomes picked over and innovative opportunities begin to disappear. This is because the best technological opportunities are typically exploited first and the remaining opportunities are more difficult to find. In contrast, if the **target** firm searches first and alone, it does not have to invent around competitors. The knowledge space is less crowded and there are ample opportunities to draw on knowledge that does not overlap with competitors. Overall, the implication is that when competitors start learning from the **target** firm, there is a likely negative impact on the innovative nature of the **target** firm’s introductions of new products.

Search by competitors can also induce racing behaviour in which the **target** firm searches simply to keep up with its competitors. This behaviour is particularly likely when firms engage in simultaneous searches with similar resources. In an extreme case, competitors’ achievements provide a continuously moving target for the focal firm, establishing a “Red Queen Effect” that the firm has to run just to stay in place (Barnett, 1997). Because racing is likely to focus the firm’s efforts on matching those of its competitors, it may make it harder for the firm to introduce new products frequently. Also, firms that race often start thinking alike about implementation instead of adopting unique R&D paths, thus killing the variation of search within the knowledge pool. As a result, innovation may become stifled. The overall implication is that racing makes innovation more difficult because the firm’s efforts are focused on staying even rather than on innovating, with a likely negative effect on both the frequency and innovativeness of the focal firm’s introductions of new products.

Katila and Chen (2008)’s study discovered that firms introduce more new products if they search after what their competitors do and they introduce more innovative new products if they

search ahead of their competitors. *“The most innovative firms combine these two approaches, bridging their own and their rival’s hitherto isolated clusters of knowledge, but avoid engaging in leaning contests in which they search at the same time as their rivals. The key insight for innovating firms, then is not necessarily to strive to perform as well as possible in absolute terms, but to be different from the competition.”*

Developers learn to innovate by following each other in the app economy. Permission use is publicly disclosed by the *Google Play Store* for each app. Although developers may not know how to exactly implement some functionality, permission use discloses a lot of tips. Therefore, developers can learn from each other. Developers innovate to enter into or open a new market. Other developers observe that ground breaking innovation activity and learn to follow. Following other’s innovation activities saves the costs by reusing the lessons from other’s innovation investments. Following is also a safer strategy. By following, there are already other developers who have devoted resources into the market and those following developers are not, there, exploring the new market alone. It is also worth mentioning that the user base is the key for monetisation in the app economy. They are collectively working to increase the market size which benefits each other. Following is not only to follow other developers’ apps, but also a result of exploiting based on a developer’ own prior apps. When other developers follow your previous apps, you gain inspiration from their following and increase the innovation space for the next app.

However, following is not a radical innovation path, so the innovation space may be crowded. Early developers move first, so they have the first move advantage in the market. Following developers require extra resources to learn from the early developers first and later they compete with other following developers to quickly implement their learning results in order to generate follow on apps. Early move developers also have more knowledge and insights about the product, but it would also be risky for early move developers to exploit faster to have new apps quicker than their followers. Therefore, developers can become distracted by racing against each other in the group that is following.

To balance the benefit and cost of following, the following speed should be optimistic. I have calculated the following speed by counting the difference of release data between two consecutive apps in each comparison group. Three independent variables are calculated for each comparison group: *days_lapsed_dvlp*, *days_lapsed_cat* and *days_lapsed_funct*. Each variable predicts the selection in relevant comparison groups by counting how many days are needed by a developer to generate a new app to follow a previous app. I did not distinguish following a good developer or a normal developer in this model on platform selection, but I continued to study the effects of following on different types of developers in Question 2 on peer recognition, which will be discussed later.

The above description basically explained that excessive rush to follow and an excessive delay to follow are not efficient. *Google* wants to see an optimistic following speed to guarantee the innovation quality. Therefore, I predict the relationship between following speed and selection should be curvilinear. For each comparison group, a hypothesis is proposed as bellow.

Hypothesis 5 *In comparison to a developer's previous apps, the relationship between following to enter and platform selection is curvilinear (inverted U-shaped), with the highest probability for platform selection occurring at an intermediate level of following to enter. Specifically, a developer is more likely to acquire a featured status at medium levels of rapid following prior to app to release, too rapid a following or too slow in following will both decrease the chances of acquiring a featured status.*

Hypothesis 6 *In comparison to previous apps by a featured developer from the same app category, the relationship between following to enter and platform selection is curvilinear (inverted U-shaped), with the highest probability for platform selection occurring at an intermediate level of following to enter. Specifically, a developer is more likely to acquire a featured status at medium levels of rapid following prior to app to release, too fast to follow or too slow to follow will both decrease the chances of acquiring a featured status.*

Hypothesis 7 *In comparison to previous apps with the same functionality, the relationship between following to enter and platform selection is curvilinear (inverted U-shaped), with the highest probability for platform selection occurring at an intermediate level of following to enter. Specifically, a developer is more likely to acquire a featured status at medium levels of speed in following prior to app to release, too rapid in following or too slow to follow will decrease the chance to acquire a featured status.*

8.2.5 Control Variables

There are some variables needed to control in both app level and developer level for obvious reasons as seen bellow.

- *Product Type*: There are two types of apps, either game apps or normal apps. Game apps and normal apps are different on competition strategy. Users would also expect functionalities and qualities that are differently from each type of app. Therefore, I controlled this variable in the developer level with three distinct categorical values. One value is *game-only* which I assigned as 1 for developers with every one of their apps in game categories. The second value is *non-game only*, which I assigned as 0 for developers with none of their apps in game categories. The third value is *mixed*, which I assigned as 2 for developers with apps in both game and non-game categories.

- *Price Strategy*: Apps adopting different price strategies would perform differently. Users would also expect different qualities within the apps, for example, users expect more to see advertisements in free apps compared to paid for apps. Therefore, I controlled this variable on the app level with three distinct categorical values. One value is *free*, which I assigned with 0 for the completely free app. The second value is *in-app purchase*, which I assigned as 1 for apps allowing in-app purchase options and letting the user pay for only what they need. The third value is *paid*, which I assigned as 2 for apps that charge for a price for them and users have no choice to try them before they pay.
- *App Sequence*: The sequenced ranking of apps by release date would also need to be controlled. Developers should acquire more skills and resources when they are producing later released apps. This is a chronology order of apps released by a single developer. The value *sequence* starts from 1. The purpose is to control the differences from a learning perspective.
- *Immediately Featured*: Developers are fighting to compete for selection opportunities. However, there is a group of developers who have been immediately featured with their very first app. These developers should be controlled from developers who are gradually recognised by *Google* for featuring. The variable is *immediate* and is controlled in the developer level. 0 means developers who are immediately featured and 1 means developers who are gradually recognised by *Google*.

8.2.6 Unit For Analysis

Despite having three comparison groups, the focus for platform selection is on the consequent changes taken by a single developer. Therefore, the unit of analysis is for each *featured* developer. It predicts in *dvlp* group with additional information from *cat* and *funct* groups. Figure A.30 shows the underlying structure of data in terms of individual units. The nodes on the diagram are specific population units. In this case, the units are apps and developers which form two levels (or classifications). The lower units form the app classification and the higher units form the developers classification. This unit diagram is just a schema to convey the essential structure of apps nested within developers. In the dataset, there were 3409 apps belonging to 365 featured developers (271 developers have more than 2 apps and 94 developers have only one app). The hierarchical structure means that an app only belongs to a developer. Such a structure may arise when I am interested in a developer performance and I made repeated measures of this by assessing app performance for multiple apps from each developer. This structure is likely to give rise to correlated or non-independent data in the sense that apps from the same developers often have a tendency to be similar on such variables. Therefore, multilevel modelling is attempted in this thesis.

8.2.7 Statistical Method

After determining the unit of analysis, the group variable is *developer_id*, and the time variable is *sequence*. Because each developer has a different number of apps, so the repeated measurement is unbalanced. The distribution of panel set up can be found in A.1. All the statistic analysis in this thesis are conducted by *Stata* 13 package ¹.

All variables used in statistical analysis have been listed in Table A.1. Table A.2 displayed the descriptive statistics results on mean, standard deviation, min and max. Table A.3 displayed correlation matrix for platform selection model.

The predictor, *status*, is a binary variable with either 0 or 1. Therefore, a *logit* model is suitable to predict the probability. A *logit* model, or logistic regression, is a direct probability model and a binary *logit* is used to predict a binary response based on one or more predictor variables. *Stata* has a special command called *xtlogit* ² to analyse *logit* models in panel data.

In my thesis, there are a few time invariant variables. For example, *coverage*, *immediate* and *scope* are attributes in the developer level, and they will not change for apps from the same developer. However, these variables are different across entities and would have some influence on dependent variable. Therefore, theoretically, a random effect should be adopted. In practice, I ran a fixed effects model and saved the estimates, then I ran a random model and saved the estimates, then I performed the *hausman* test. The *Hausman* test returned a strong non-significant result, which indicated that the final model should be a random effect. So, the final model is based on a random effect estimator.

8.2.8 Statistical Results

Three models have been attempted and the results are presented in Table A.4. Model 1 reports the results with the control variable included. Model 2 reports the results with the addition of uniqueness, diversification and following. Model 3 reports the results with the addition of squared independent variable. The results show that three curvilinear relationships (uniqueness, diversification, following) are supported by three independent variables (*unique*, *days_lapsed*, *scope*).

Uniqueness

unique_dvlp is strongly significant ($p < 0.001$) with a positive coefficient (1.492). It is an evidence that a developer can increase their selection opportunity by differentiating from their own apps. Rather than doing simple duplication, a developer can innovate their apps to make

¹<http://www.stata.com/>

²<http://www.stata.com/manuals13/xtxtlogit.pdf>

them unique comparing to their previous apps. This “unique to self” action would be captured by *Google* as a signal for innovation ability. However, *unique_dvlp_2* is significant ($p < .05$) with a negative coefficient (-2.877). *unique_dvlp_2* is the square of centralised *unique_dvlp*. *unique_dvlp_2* proved that too much uniqueness to his own apps would actually damage the selection opportunities for a developer. This negative effect is due to nonlinear increasing on cost and technology difficulties as the increase of uniqueness. Therefore, *unique_dvlp* and *unique_dvlp_2* proofed the curvilinear (inverted U-shaped) relationship between app uniqueness and platform selection comparing to a developer’s prior apps. The highest probability for platform selection occurs at an intermediate level of uniqueness. Specially, a developer is more likely to acquire a featured status at medium levels of uniquely using permission and to decrease at either the lower or higher levels of uniquely using permission.

unique_cat is moderately significant ($p < 0.01$) with a positive coefficient (1.016). It means that higher uniqueness to good developers would increase the chances of being selected by *Google*. Good developers are targets for everyone who wishes to learn and compete. Benchmarking against good developers is an indicator for signal quality. However, *unique_cat_2* is a weak significant ($p < 0.20$) with a negative coefficient (-1.687). The same as *unique_dvlp_2*, *unique_cat_2* should indicate that too much uniqueness to good developers would associate with high costs and technological difficulties. Therefore, I still interpret this is an evidence for the curvilinear (inverted U-shaped) relationship between *unique_cat* and platform selection.

unique_funct is not statistically significant. Therefore, Hypothesis 3 is not supported in this thesis.

Diversification

scope is strongly significant ($p < 0.001$) with a negative coefficient (-0.452). It means that if a developer attempts to create apps in a larger number of distinct categories, they are less likely to be selected by *Google*. In other words, a less diversified developer is more likely to be selected. The *Google Play Store* has such a large number of developers who have apps to solve every kind of user need that only specialised apps are recognised in such extreme competition. *Google* encourages developers to be specialized in what they are good at. The curvilinear test (*scope_2*) is moderately significant ($p < 0.01$) with a positive coefficient (0.0421). It means that if developers are very good at and are confident on diversification strategies, they are also likely to be selected by *Google*. These developers are normally big companies with strong finance and teams. Their diversification strategy is not to simply duplicate apps or to copy from others, but is driven by company expansion. Therefore, *scope* and *scope_2* proved the that curvilinear (U-shaped) relationship between diversification and platform selection. The lowest probability for platform selection occurs at an intermediate level of diversification. Specially, a developer

is less likely to acquire a featured status at medium levels of diversification, and their chances increase at either the lower or higher levels of diversification.

Following

days_lapsed_dvlp is strongly significant ($p < 0.001$) with a positive coefficient (0.00552). It means the slower release of following own apps are more likely to be selected. Developers need to spend enough time on app creation to guarantee quality. Exploiting based on a developer's own knowledge happens with enough time and resources to devote into research. It also means quality is very important for selection. The curvilinear test (*days_lapsed_dvlp_2*) is moderately significant ($p < 0.01$) with a negative coefficient (-0.00000484). It means developers who are too slow in following on with their apps are less likely to be selected by *Google*. Developers who are slow at following on would have already made their previous app take the monopoly on the market position, and it is difficult for slow followers to compete. Therefore, *days_lapsed_dvlp* and *days_lapsed_dvlp_2* proved the curvilinear (inverted U-shaped) relationship between following speed and platform selection, comparing to a developer's previous app. The highest probability for platform selection occurs at an intermediate level of following speed. Put simply, a developer is more likely to acquire a featured status at medium levels of following speed and their chances decrease at either lower or higher levels of following speed.

days_lapsed_cat is not statistically significant and *days_lapsed_funct* is weak significant ($p < 0.2$). I considered that both Hypothesis 5 and 6 are not supported in this thesis.

8.3 Peer Recognition

The last section examined how developers follow other developers to learn from and race with each other to search for innovation. However, it is still not clear whether developers should follow good developers or ordinary developers, and how market performance responds to this following decision.

Innovation requires ability, and only good developers would be able to understand the technological challenges and the long term market potentials. Good developers are leading the innovation and everyone (including users, *Google* and developers) in the market pays attention to them. Therefore, I assume that developers follow good developers to trigger a signal or to signal to attract good developers to follow them. The collaboration taken by good developers would help to explore and claim the market. I study this "peer recognition" effect in this section.

Following good developers would have a lot of benefits. Firstly, it would be less risky in terms of market failure. Good developers normally have stronger research ability to determine the market potential. Even when the market is new, good developers have a stronger ability

to explore and increase the market size. Meanwhile, *Google* would prioritise recommending good developers to the market and users have more confidence to following good developers into a new market. Therefore, a market section having good developers who have entered into it can grow quicker. Secondly, good developers, especially big companies, are less likely to focus on niche market section, which provides opportunity for small companies to catch up and obtain the position. Thirdly, good developers have higher value to learn. Good developers have a higher innovation abilities. Their innovation activity should always indicate the probability for future advanced technology innovation. Good developers may also respond quicker to the market and to technological change. Therefore, following good developers can help secure a lot of useful information.

Attracting good developers to follow you also has benefits. Good developers would bring their large amount of subscribers or users to their following apps. These users may also take the opportunity to look at the other apps including your apps. This is extremely beneficial, especially when you are not “famous”, the market section is new or it is difficult for you to increase your user base. Good developers following you also attract *Google*’s attention to your app and promote the market section, which also help you to increase your user base. Your solo innovation is risky, as “many new ideas are bad ideas”(Katila and Chen, 2008). Having good developers following you is a kind of confirmation on the market value and potential you are currently working on, therefore, it should enhance your confidence to continually exploit innovation. In addition, good developers following you also bring your more inspiration and knowledge on innovation and your following app is more innovative by benefiting from reusing the knowledge learned from good developers. In other words, attracting good developers to follow you help you to claim the new market (Santos and Eisenhardt, 2009).

The downside of peer recognition is the increase in competition. The joining of both good and ordinary developers increases the competition. However, the total number of good developers is not that large, so the threat from good developers is not on the total number of apps, but on their ability to be competitive. Good developers have all kinds of strong abilities, their joining would normally take a large part of market share. It is a big threat to developers who are not ‘famous’ yet. The joining of ordinary developers also increases the competition. Due to the large number of ordinal developers, the market can become very crowded. Ordinary developers are less interested in innovation, so their actions such as copying would disorder the market ecology.

It is also interesting to understand how this peer recognition would effect on developers in other way before or after they are selected. Peer recognition helps a developer claim the market when he has less ability. However, when a developer is a good developer and gains enough ability, the help from other developers’ involvement becomes smaller but generates

more competition. After becoming one market leader, whether he should continually follow other developers is an interesting question to answer. This section studies the interaction effect between following and selection.

8.3.1 Dependant Variable

The interest of this model is to see the market response to a developer's signal. Therefore, the dependent variable is the market performance. The *Google Play Store* already disclosed some market performance data for developers to monitor their operation and for users to make decisions on purchase. The disclosed market information includes *avg_review_score*, *reviewer_count* and *app_install*. The *app_install* is in categorical data format, so it is not very useful to compare the rational changes in details. I decided to use the other two variables to measure the market performance which comes from user reviews.

Each product aims to achieve a better market performance. However, there are costs incurred to achieve this. The market performance can be proxy by app installation and user satisfaction. Installation represents how many people have already purchased the product and satisfaction proxies the product quality and branding with users' willingness to recommend or purchase again. I do not have the exact installation data, but the number of people who reviewed the app can approximately represent the portion of installations. I assumed that, for each app, a certain average portion of users post their reviews to let other users know about the apps. This assumption has been used in several marketing research studies (Fu et al., 2013; Lee and BradLow, 2011). Meanwhile, each review represents a score of satisfaction by the user. Therefore, the combination of installations and reviews can represent the existing and potential market performance (see Figure A.31). I defined an equation for market performance (see Equation 8.4) as the sum of review score timed by the number of reviewers. The variable is called *performance* and contains count data.

$$performance(app_i) = \sum_{s=1}^5 COUNT_s \times s \quad (8.4)$$

8.3.2 Independent Variables

To study the difference of following between good developers and ordinary developers, I have to narrow the comparison into apps having similar functionality. *cat* group only has featured developers, so it is not suitable in this study. *dvlp* group is used to set up panel data structure and *funct* group is used to count for the following by different kinds of developers.

Panel data can only predict the future with past data rather than with future data. Due to this limitation, I can only predict the number of apps to follow and not with the number of apps attracted for following after an app. Therefore, in each *funct* group, I have calculated the number of similar apps by featured developers (*apps_featured*) before the release of the **target** app and also the number of similar apps by non-featured developers (*apps_nonfeatured*). They are the independent variables to predict the differences on market performance by following different types of developers.

As discussed before, it is beneficial to follow good developers compared to following ordinary developers. The cost is to increase competition. Therefore, I made following two hypotheses between peer recognition and market performance:

Hypothesis 8 *The relationship between peer recognition by good developers and market performance is curvilinear (inverted U-shaped), with the highest market performance occurring at an intermediate level of peer recognition by good developers. Specifically, a developer is less likely to obtain high market performance at medium levels of peer recognition by good developers and such chances decrease at either the lower or higher levels of peer recognition by good developers.*

Hypothesis 9 *The relationship between peer recognition by ordinary developers and market performance is negatively linear and lower market performance occurs at higher levels of peer recognition by ordinary developers.*

To study the differences before and after selection, I also tested the interaction effects between two independent variables and *status*. Once developers are selected or featured by Google, they need less help from following, especially from ordinary developers. Therefore, I made the following two hypotheses on interaction effects:

Hypothesis 10 *The relationship between peer recognition by good developers and market performance is not affected by whether a developer has obtained the featured status or not.*

Hypothesis 11 *The relationship between peer recognition by ordinary developers and market performance is affected by whether a developer has obtained the featured status or not. Specially, after a developer is featured, the market performance will decrease with higher peer recognition by ordinary developers.*

8.3.3 Control Variables

Besides all control variables in the platform selection model, the peer recognition model adds one additional control variable. This additional control variable is *productivity*. It is an

attribute in the developer level representing the total number of apps created by a developer. Developers can produce and maintain a different number of apps, therefore, their resources and diversification strategies are also different. It would influence their decision and behaviour on following. The model wants to remove the bias on prediction caused by this *productivity*, so I decided to control it.

8.3.4 Statistical Method

The statistical method in peer recognition is very similar to the method in platform selection, except that peer recognition adopted a negative binomial regression model in panel data. It happens because the dependent variable *performance* is a count variable and over-dispersed. From descriptive statistics (Table A.2), we can see that *performance* has a standard variance (2,388,899) much larger than the mean (331,151). Negative binomial regression is good to deal with such over-dispersed count predictors.

Negative binomial regression needs to check the existence of zero-inflated issues. By checking the data, there are 16 out of 3409 apps having *performance* equal to 0, which is only counts as 0.5% ($16/3409=0.5\%$). Therefore, it is safe to ignore the zero-inflated issue in the analysis.

Stata has a specific *xtnbreg* command³. With the same steps and reason, I decided to use random effect in the final model. Table A.5 displayed correlation matrix for peer recognition model.

8.3.5 Statistical Results

Three models have been attempted and the results are presented in Table A.6. Model 1 reports the results with the control variable included. Model 2 reports the results with the addition of independent variables. Model 3 reports the results with the addition of a squared independent variable and interaction effect. From the results, we can see that a curvilinear relationship (peer recognition by good developers) is supported by *app_featured* and an interaction effect is supported by *featured X apps_featured*.

app_featured is strongly significant ($p < 0.001$) with a positive coefficient (0.0752). It means that following good developers would be likely to generate a good market performance. *Google* acknowledges the behaviour to learn from good developers which is believed to be a strategic mechanism to manage the innovation in the *Google Play Store*. Curvilinear test (*app_featured_2*) is significant ($p < 0.5$) with a negative coefficient (-0.00322). It means that too many good developers entering into the market would significantly increase competition and

³<http://www.stata.com/manuals13/xttnbreg.pdf>

their strong ability would take the most of the market share. It would decrease the follower's market performance. Therefore, the relationship between good peer recognition and market performance is curvilinear (inverted U-shaped) with the highest market performance occurring at an intermediate level of peer recognition by good developers. Specifically, a developer is less likely to obtain high market performance at medium levels of peer recognition by good developers and such chances decrease at either the lower or higher levels of peer recognition by good developers.

app_nonfeatured is strongly significant ($p < 0.001$) with a negative coefficient (-0.0246) in model 2. It means that following negative developers may negatively impact upon market performance. It is due to the fact that lower achieving developers may have higher chances to predict the market by mistake, or make wrong decisions. Average developers may also be interested in copying rather than on innovation, so their products would damage the user experience and disorder the market. However, curvilinear tests on *app_nonfeatured* is not significant. Therefore, it confirmed the linear negative impact by following average developers.

The next step is to look at the interaction effect between following and selection. *featured X apps_featured* is not significant. It means that even if a developer is selected or featured, the impact on following good developers is the same as there is no interaction effect. Because *app_featured* is significantly positive, a featured developer still receives the benefit by following other good developers. *featured X apps_nonfeatured* is strongly significant ($p < 0.001$) with a negative coefficient (-0.0348). It means following low to average developers negatively impacts market performance after a developer has been featured. *app_featured* is not significant in the model but has a positive coefficient (0.0102). *featured X apps_nonfeatured* is significantly negative, or it means that after a developer has been featured, their market performance is negatively impacted by following non-featured developers. The reason would be that non-featured developers crowd the market and increase a user's search costs. This is another indication that developers should follow more good developers more and follow average developers less. So Hypothesis 10 and 11 are supported.

8.4 Discussion

This chapter studied how a developer constitutes the signal to *Google* for platform selection and how market performance responds to developers following. I have plotted the statistical data and prediction after statistical analysis. In this section, I discuss the findings with these plotted charts.

Figure A.32 demonstrates the relationship between uniqueness and platform selection. The chance for platform selection (see *red* line) firstly increases with the expansion of uniqueness

until it reaches the highest point, and then it decreases. Uniqueness is a foundation for innovation. Developers innovate to have new apps that are different from existing apps. This differentiation enables developers to enter into the new market section to serve new users or provide existing users with a seamless user experience. *Google* wishes to see more uniqueness. More unique products would solve broader user needs, which would attract more users to *Android* and increase the competitive advantage to compete with other markets. Therefore, *Google* would observe the signal from these developers creating unique apps and select them for featuring as a way to protect their innovation. *Google* aims to use this selection to encourage developers to generate more uniqueness. However, it is costly for developers to uniquely use permission, which generally requires strong technology skills and market knowledge compared to other actions like coping or duplicating. Being too unique can lead to technology complexity and market uncertainty, which generates negative user experience and waste resources. That is why chances for selection actually decrease with very large uniqueness. The statistical analysis has found evidence for this curvilinear (inverted U-shaped) relationship from uniqueness to self and uniqueness to other good developers. Uniqueness to similar apps is not supported due to insignificant statistical results.

Figure A.33 demonstrates the relationship between diversification and platform selection. The chance for platform selection (see *red* line) firstly decreases as increases on diversification scope take place, it then reaches the lowest point, and then increases. Diversification allows developers to compete in different market sections, which is either to attract new users with a new app or to serve existing users with a different app. This would diversify the risk and cost of signal product failure. However, diversification distracts a developer's resources. The developer has to diversify their resources to develop and maintain several different products. More importantly, competition in the *Google Play Store* is so high that only really good apps can stand out from the crowd. Diversification would prevent developers from being dedicated to a single product innovation, which would not enhance and deepen the innovation into what it should be. *Google* wishes developers to focus on their specialties and keep innovating their products. Such a large number of developers would always have a chance to have someone working on a niche market, but innovation is a rare resource and should not be distracted. That is why *Google* pretends to reduce the selection opportunity for diversification. However, when a company becomes big enough, it has enough resources to balance on both diversification and innovation. In this case, diversification is a strategy for big companies to continue to grow, and should be encouraged. Therefore, the relationship between diversification and platform selection is curvilinear and is supported by the statistical results.

Figure A.34 demonstrates the relationship between following speed and platform selection. The chance for platform selection (see *red* line) firstly increases along with the increase of

following time and as it reaches the highest point, it decreases. Developers can learn from each other to innovate. Developers need to spend time observing other developer's innovative activities and to absorb the knowledge from lessons and then convert them to implement a new app. Therefore, a longer time to prepare a new app is more likely to lead to a good quality and highly innovative new app. *Google* does not want to see rapid followings in apps because too quick a follow-up is more likely to be a copied or duplicated apps. These apps damage the user experience, and also damaged the investment returns of other developers who are interested in innovation. *Google* wishes to give innovative developers a longer leading time. That is why the chances for platform selection decrease when there is an excessive rush to follow. However, excessive slow follow-ups is also not good, since it may reflect lower learning ability. Developers may also lose the good chance to enter and secure their market position. Excessively slow follow-ups also brings about the risk that other developers may create new apps that are similar to your idea before you; then your product may be less competitive and their product would already be iterated. Developers should avoid this. This curvilinear (inverted U-shaped) relationship has been supported by self following. Following good developers and following similar apps are not supported in the statistical results.

Following speed is important for platform selection and who to follow is an important for market performance. Figure A.35 demonstrates the comparison of peer recognition between good developers and average developers. Following more good developers would secure better market performance (see *green* line) compared to following more ordinary developers (see *yellow* line). Developers can learn a lot from good developers to improve the innovative behaviours. Following good developers secures against market failure and promotes development. *Google* want developers to learn from, and to follow, good developers. One reason is that good developers are worth learning from for innovation and good developers always lead the market. The other reason is probably due to the small number of featured developers which makes innovation easier to manage. By encouraging the following of good developers, *Google* can focus on influencing these good developers and drive the learning culture in the ecosystem. However, following too much leads to high competition which will negatively contribute to market performance. That is why the relationship in Figure A.36 is curvilinear (inverted U-shaped). Even more, Figure A.37 demonstrates that following average to below average developers significantly negatively impacts market performance, especially after the developer is selected. Average developers have less ability to determine the market trends and less interest in innovation. Featured developer's market performance would be negatively impacted by lesser quality developers' disordering behaviour. It suggests again that developers should follow good developers rather than poor ones.

In summary, the statistical results suggest that developers should make costly actions to constitute signals and these signals are observed by *Google* to interpret for innovative qualities to make its selection decision. Costly actions include the uniquely use of app permission, less diversified apps to enhance higher levels of specialty, and proper following to learn how to innovate. Developers should follow other good developers to maximise their market performance. The findings confirmed that *Google* can efficiently manage the innovation in such a highly competitive market via the selection of developers. With its selection mechanism, *Google* signals the innovation selection criteria to the developer community, and this encourages them to learn from each other for competition. Without scanning all of its developers, *Google* can concentrate more on influencing good developers to innovate, and good developers spread *Google's* desire to every other developer to push the *Google Play Store* into a healthy ecosystem.

Chapter 9

Conclusion

This chapter firstly discusses the implications and contributions of this research, and then provides an overall summary. It then explains its limitations and proposes the future works.

9.1 Discussion

9.1.1 The Platform Owner's Reciprocity Function

In a two-sided market, producers sell their products via a market and consumers buy products from the market. Consumers and producers are reciprocally connected via the market. Consumers listen to the market operator to make their purchase decisions, and producers would want market operators to recommend their products to consumers. It brings the responsibility for the platform owner to control and maintain the quality of the complements in order to ease the interaction between producers and consumers.

This reciprocity function pushes the development of "platform thinking" ([Cusumano, 2010](#)). "Platform thinking" promotes to change the treatment of competition into collaboration by identifying common interests. Three parties, in this context, share a common interest to developing the market into a healthy ecosystem ([Gawer and Cusumano, 2013](#)). The network effects of a healthy ecosystem would bring users more products to meet their various demands and better products at a cheaper cost. Producers also benefit by having easier-to-use tools, by reusing publicly available knowledge and then enjoying the increasing revenue from a larger user base. Meanwhile, the development of an ecosystem expands market operators' commission revenues, and enhances their competitive advantage over other rival markets.

To achieve this, existing platform literatures argues that function of the platform owners is to develop the network effects across different parties in the market ([Gawer and Cusumano, 2013](#)). Since both producers and consumers are reciprocally influenced, the platform owners

should protect on what they should protect the market. Protection from existing platform strategies are in macro scope with the assumption that members on one side behave same as the protections. For example, the platform should modify the price structure to attract the price-sensitive user side to increase the development of other sides (Rochet and Tirole, 2006), or develop boundary resources to ease the technical development works (Eisenmann et al., 2011). This assumption ignores the micro difference between members on the same side. This thesis approves that developers have an unequal response to the platform strategy.

I argue the importance for the platform owner to understand the micro difference of a particular side in its market. The understanding of their unequal response in one side would generate an opportunity to differentiate members on the other side. This shift of focus helps to control the network effects at a deeper level. This thesis demonstrates the power of innovation management by solving the problem of how a producer (or complementor) signals to compete in an extremely crowded (platform based) two-sided market. By differentiating developer's strategic behaviour, the signalling selection model developed in this thesis enables the platform owner to develop the peer effects in learning which can ease the platform management work.

The implementation of the platform owner's reciprocity function would be different in a different business context. When a two-sided market is in an early stage, the priority of the platform owner is to increase the growth to amplify the network effects. *Android* adopts an open strategy since its beginning. After early development, the market shares are equally occupied by several major platforms. The platform owner envelops (Eisenmann et al., 2011) the popular features from his rival platforms to reduce the motivation for developers to switch to other platforms. These two kinds of business context have been studied in literature, and the third kind of business context is rarely studied which is the duopoly. The duopoly is recently formed by *Android* and *iOS* recently and is believed to last for a long time by looking at the ecosystems they have established. In this business context, there are only two platforms and the developers have to compete directly as switching behaviour is inefficient. The platform has gained a massive number of consumers and producers, so the priority of platform attraction is on "good" developers. I argue that the platform owner should differentiate to protect more "good" developers rather than treating all developers as the same in this context.

9.1.2 The Chaos Of Crowdedness

However, the differentiation at a micro level is not easy. The crowded nature of the duopoly introduces the chaos. Many producers are profit-maximum driven (Carlin et al., 2001). Producers spend different amounts of efforts on R&D by adjusting to their expectations of profit returns and time planning. Some less responsible producers engage in unsustainable actions which disorder the market. For example, copycat (Yadong Luo et al., 2011), imitation (Enkel

and Gassmann, 2010), free-riding (Evans, 2003; Han et al., 2012b), malicious behaviour (Zhou et al., 2012b), legitimate products (Zhou et al., 2012a), etc.

Chaos is serious in the duopoly, and the user data studied in the qualitative chapter is particularly hardest-hit. Some technical research (Felt et al., 2011a,c, 2012; Sarma et al., 2012) on app permission confirms the misuse of user data in the app economy. I argue the motivation of misuse, because the value of user data is recognised as a unique resource for innovation and competition (Lim and Bentley, 2012b). Mobile apps have generate and capture a great amount of user data, and this data becomes an important asset in the digital economy (Kalapesi, 2013; Schwab et al., 2011). Users can stay with the business which has richest user data (Koch and Kerschbaum, 2014). It drives the popularity of many new business models in the app economy, like *freemium* (Liu et al., 2014). This also pushes the development of API strategy (McDonnell et al., 2013), which performs as an efficient tools for internal or external communication and collaboration between apps and developers (Aitamurto and Lewis, 2013; Chin et al., 2011; Li et al., 2014), and attracts other externalities (Boudreau and Lakhani, 2009), like mashup (Yu et al., 2008).

Chaos also leads to the competition between developers for user attention. User attention is important to the business performance. The apps displayed on the front pages of the *Google Play Store* receive more user attention and, therefore, higher customer access, which would potentially contribute to the increase of sales. In order to attract user attention in app markets, some app producers frequently updates (Möller et al., 2012) their apps to introduce new features or just to compete for the chance to display on the front pages of the *Google Play Store*. Competition in updating leads to the increase of platform operation costs, data traffic on users' mobile devices, and search costs by users. More importantly, it increases the chance for developers to "treat" on the request of their app permission. Some apps perform well and leave a "good" impression in its early versions, but start to "steal" user data in later versions, because users can lower down their precautions in later versions (Barrera et al., 2012).

The third perspective of chaos is to motivate the piracy (Rasch and Wenzel, 2013). To save the cost on R&D, some producers repack (Zhou et al., 2012a) existing apps by giving them a new title or look. Some of them looks very similar and confuse the consumers at the point of search and selection. It also damages other producers' benefits, because the original creator may lose profit and then lose the motivation to continue with R&D. In the long term, it generates a negative impact for developers to put investment on innovation (Mantena et al., 2010) which would reduce the platform's competitive advantage. Therefore, the platform owner has to make extra works to identify the original work, and then provide adequate protection.

9.1.3 The Communication In Platform Thinking

I study the cause of chaos and find that information asymmetry develops chaos even more which is very serious in the app economy. Due to the unequal information held by different parties, only producers themselves know their real quality, but all the producers "pretend" to show their "good" qualities, and some of these qualities are not "real" (temporal or fake). This brings the difficulty for the market operators to understand their innovative nature. Without the knowledge of developers, the platform owner has the difficulty to select developers for recommendation to the users and the consumers have difficulty searching for desired products with very good qualities from such a huge amount of products.

It also increases the cost for good developers to stand out as they are distracted by making some costly actions to let the public know their real ability. Rather than focusing on innovative activity, developers have to spend large amounts of resources on marketing. Even when marketed, the targets (like final users) would have a different understanding about the developers. It brings lots of uncertainties to the developers and may discourage the developers to invest in the actual production and innovation. The platform owner has the responsibility to create a clear channel for communication.

The thesis attempts to review the existing platform strategies in two-sided markets to examine their efforts for easy communication. It concludes that previous platform studies do not specific pay attentions to solving the communication problems in their strategies. Many strategies are proposed in theoretical modelling ([Cusumano, 2010](#)), and they emphasise the setting of barriers. By setting up different pricing ([Hagiu, 2006, 2009b](#); [Lin et al., 2011](#)) to different groups of complementors, the developers have different competences and will adjust their decision on whether to stay or leave the market. The barrier works as an indicator for developers to communicate to themselves. Some empirical studies of platform strategy focuses on the exclusive contracting and subsidising ([Armstrong and Wright, 2007](#); [Hagiu, 2009a](#); [Hagiu and Lee, 2011](#); [Mantena et al., 2010](#)). They rely on the platform owner's manual picking of "good" developers, which works in a small number of communities like video game developers, but is not feasible in the app developer's community. The most relevant discussions are from the scholars ([Economides and Katsamakas, 2006](#); [Schiff, 2003](#)) who encourage the opening of the platform and the market. It gives complementors more ownership and equal chance by lowing down the technical requirements. This would increase the sharing culture and encourages members to talk in the market. However, their interest is on how to encourage collaboration, and open strategy would lead to crowding which would make communication even more difficult.

The lack of focuses on communication problems is particularly due to the market nature of the platforms they studied. The increase of communication and operation costs would motivate

developers to move, which generates the crowding out effects (Boudreau, 2012). It means the massive new entrants would crowd out the existing "good" developers. Boudreau (2012) studies this phenomenon in PDAs age. The chaos of crowdedness will make a platform lose "good" developers when they have many similar rivals. However, this condition is different in the current duopoly. Even the communication cost can be high, the developers have to stay to compete for a position in a platform. I argue that some developers would engage in certain costly actions to communicate with the purpose of reducing information asymmetry.

I believe it is necessary and valuable to study for an efficient way for communication of innovativeness in the app economy. I argue that the differentiation of complements is important to control the average innovativeness when a market is already established with strong network effects. The current strategies on complements differentiation is to set up entry barriers and open the platform. This kind of strategy is useful when there are many alternative platforms to switch to (for example, during the age of PDAs as studied in Boudreau (2012)'s paper), but not efficient in the current duopoly. I argue that the current platform strategies would narrow their space to communicate, because a common barrier will increase everyone's opportunity costs to innovate and grow, and is particularly unfriendly to new entrants who are potentially "good". I promote that both platform and developers have the motivation to overcome the information asymmetry to identify the innovativeness, especially to protect new entrants who are potentially "good".

9.1.4 Google's Strategy

The *Google Play Store*, the official online two-sided app store for the *Android* platform and also the market leader of one of the largest app economies in the world, has a priority to solve this problem. The virtual exchange nature in the *Google Play Store* liberates the geography limitation for developers and users. *Android* is rapidly developed with the adoption of open strategy. Open strategy not only develops the market size rapidly, but also generates crowding. According to my collected data set in 2014, there are 232,397 developers and 1,340,505 apps in the *Google Play Store*. Thousands and thousands of new versions and new apps are generated each day, which makes it difficult for users to search for a desired app. *Google*, as the market operator, has the responsibility to control the average quality of apps in the *Google Play Store*. The overwhelming number of apps makes it difficult for *Google* to check the quality of each and every app. At the same time, *Google* wants the market to be full of visiting traffics, since the numbers of traffics is the basic of a strong online platform that enjoys a competitive advantage from other competitors like *iOS*. Therefore, *Google* needs many more number of users and experienced developers along with quality apps in the store.

To achieve this target, *Google* has to encourage and protect innovation in the *Google Play Store*. One strategy *Google* adopts is to reduce information asymmetry. *Google* discloses the profiling information of apps and developers to the public and this profiling information includes user reviews, similar apps recommendation, developer's privacy policy, etc. This profiling information enables people to study a developer's background and look at how other people evaluate the developer. Some users may also have the interests and skills to deeply research other information traced from this profiling information (Fu et al., 2013).

Another strategy used by *Google* is to adopt a featuring selection mechanism to promote a learning culture. *Google* will periodically select some apps for featuring. The apps selected by *Google* would get some extra benefits, for example, users see the selection by *Google* as a guarantee of quality, so selected apps would receive more attention from users and then have a boost in sales. *Google* will also place greater priority on recommending and promoting apps from some particular developers. After being featured, there could be more opportunities for the new developers to ask for collaboration. This process will certainly make the selected developers unique and outstanding as well as ensuring ensure the success for them in the *Google Play Store*.

Google did not explicitly disclose the detailed criteria for selection. As a result, the unselected developers have to learn from selected developers in order to compete for selection opportunities. Thus, the learning motivation and learning channels are established. As a result, *Google* will have more time to influence the learning by focusing more on managing the innovation of selected developers, as it would indirectly influence non-featured developers rather than spending a huge amount of efforts on governing all apps and developers in the market. In my collected data, there were only 365 developers were selected, and the chances for selection are only 0.15%, which means that competition for selection is extremely hard in the *Google Play Store*.

I studied this featuring selection mechanism and developed a signalling selection model. In this model, developers are motivated to engage in innovative activities according to their competence. Innovative developers have lower opportunity costs to make innovation, so they are more likely to communicate this to the platform owner as signals. The platform owner will observe the signals and feedback with great benefits, for example, the platform owner will award the developer a featuring status which would potentially bring a large number of customer. The platform owner takes this selection opportunity to set up a communication channel. The developers can communicate their innovative skills via the channel. The platform owner can observe the developer's innovative via the channel. The selection feedback will also provides users with a criteria for searching, as users can simply listen to the platform owner's selection to understand how good a developer really is.

By looking at the innovation, the developers' experience and market status receives less consideration. It means the platform owner emphasises more of the actual actions done by a developer, no matter whether a developer is new or not. It would encourage a broader range of user innovation by increasing their opportunities of selection. A valuable benefit for the platform is to ease their management job by generating the peer effects in learning. A non-selected developer will be motivated to learn from selected developers to increase their selection chances. It would encourage developers to keep learning. A benefit of learning is to solve the "fragmentation" issue caused by open strategy. The signalling selection model will accelerate the innovation speed, because developers are encouraged to learn and compete for the chance to be selected, new technology can be rapidly spread and adopted.

The signalling selection model solves the limitations of other platform strategies. Firstly, it differentiates developers' quality but does not reduce the size of the developer community. Barrier based strategies will block the less-competent developers out of the market which will have a negative impact on network effects. In contrast, the signalling selection model ensures all the developers to maintain the network effects, and lets the market select the developers. Secondly, it protects the new entrants who are potentially innovative. Barrier based strategies will increase the opportunity costs for everyone in the market, and newly entrants will easily "die" from this increase. The signalling selection model is based on developers' intrinsic innovation which removes the interference of other factors. Thirdly, the signalling selection model promotes the learning effects. Non-selected developers are motivated to learn from selected developers on how to generate innovative signals. It is not the intention to block out the non-innovative developers, but it provides them with space to learn and improve. With these learning effects, the platform owner can use a small group of innovative developers to influence a large community of developers.

9.1.5 Developers' Strategy

For developers, the question is how to be selected. *Google* periodically monitors the market for selection. Developers, especially good developers of a high quality, need to communicate with *Google* to let *Google* understand their real ability. The communication is interfered with in such large market due to the information asymmetry. Therefore, developers of a high quality need to make some costly actions that developers of a low quality will not be willing to make. The reason for developers having low quality being unwilling to take such actions is because they have higher opportunity costs (Connelly et al., 2011). It costs much more for developers of low quality to make such actions relating to their investment return. In other words, these costly actions are only distinctly taken by developers who have the right level of quality, so these actions serve as the indicators of quality or ability. *Google* will observe these costly actions to

indicate the developer's real quality. This is the channel for good developers to communicate their real quality. This explanation is developed from signalling theory (Spence, 1973) that had won a Nobel Prize in 2001. The signalling theory has been applied to the job market (Spence, 1973), "lemon" market (Akerlof, 1970), education market (Bedard, 2001; Lang and Kropp, 1986), IPO (Ahlers et al., 2015), etc, but it is the first time it has been applied to study platform selection in such a highly competitive app economy during the writing of this thesis.

The thesis reviews the literatures to understand some commonly identified strategic behaviour in innovative activities. Many scholars have studied innovation strategies to improve the competitive advantage via differentiation, diversification, business models and learning. Some scholars devoted special interests to the open source software developers to understand their motivation of "free contribution". Open innovation has been successfully adopted by many industries to make great business, by sharing some monopoly control on some proprietary components. The open culture has significantly improved the variety of products and barriers to innovate, which enables some ordinary users to "accidentally" become inventors or entrepreneurs and to join the competition. This thesis argues that the platform strategy should also protect and encourage these "weak" competitors in the markets, as they often bring innovative behaviours and freshness into the market. It promotes the research on how such strategic behaviour can identify the inventiveness, no matter whether the developers are well established players or new entrants.

Some strategic behaviour has been found to be particularly popular in app market. For example, the study of a developer's privacy policy confirmed the importance for developers to accumulate user data as a unique competitive resources. The use of user data is strongly associated to the developer's business model. *Freemium* is a new business model and popular to see in app economy. In order to acquire more user data, developers may diverse their apps. However, some developers use very little user data to build the trust. The extreme competition also motivates the illegitimate behaviour, like malicious, repacking, and updating. Malicious will cheat users with some fake functionality and "steal" user data. Some developers will repack other developers' apps and "resell" under their own name. Some developers will frequently "update" their apps with the only purpose to display their apps in the front, however the "updating" does not update any functionality or fixed any bugs. Those malicious has confused the selection in the market.

Three particular costly actions have been identified to constitute signals. The first action is to uniquely combine app permission. App permission is the key to implement function and to obtain user data. Combining permission in a different way will produce different app function. It improves the competitiveness with new attractive features, or improved technology efficiency. However, app permission is technical work. Uniquely using permission requires

developers to monitor the development of permission technology and to observe the market trend, which will increase the company's operation cost. Even more, the uniqueness (uniquely combining app permission) means newness, that is always accompanied with unstable function and is less community supported. In a crowded market, uniqueness is also not easy to achieve as the innovation space is narrow. Thus, uniqueness action is costly. *Google* will encourage uniqueness, as it would increase the product variety, comparing to copying or duplication. Therefore, *Google* will encourage uniqueness in *Google Play Store* by selecting developers who are willing to produce unique apps.

The second action is to diversify. Developers can diversify to spread the risk of single product failure into multiple products. Diversification lets developers have broader users groups or enhance the existing user satisfaction. Diversified apps will help developers to collect broader user data and to analyse for service and product innovation. However, diversification will distract developer's resource into multiple products, rather than continually improving a single product. Each single product may not be able to improve as much as it could without diversification. The market is so crowded, and there would be always some developers working on some user needs. If the app quality is not impressive, each app in the diversification may have a tiny market performance comparing to the high quality apps. *Google* wants developers to focus on improving an app into a really good quality one, unless developers are really strong enough to confidently adopt diversification strategy. Developers can learn via diversification, but they should at least have a specialist and dedicated app to obtain large market share. In crowded markets, users can have large product choices, but only product that have really good quality to solve user demands can get user population. Therefore, *Google* would monitor developers' diversification strategy to make selection decision to influence their professionalism.

The third action is to follow properly. Developers are not solely responsible for innovating. This is because they are developed by depending on others, like a fabrication. Some developers are using other developers' services or products in their innovations. Developers are also learning to search for innovation space. Furthermore, reusing other developers' R&D or lessons would save the cost on a developer's innovation experiment. Others' innovation attempts will also inspire a developer's innovation scale and scope. Learning is a time-consuming process and requires time to observe others' innovation activity, absorb the knowledge, and then convert it into a creative product. Thus, by slowly following, developers will be more likely to create an innovative app with good quality. However, innovation search would lead to a race between the developers. Developers are racing to release a new app to compete for a market position. Slow following would bring a risk for other developers to innovate faster, which would make your app a duplicated product. The later entrants will also face massive competition and pressure as other faster followers may already have many products in the market. *Google* wants developers

to follow each other while learning from one another, thereby securing the positive effects of selection mechanisms. Following will boost innovation speed via learning culture.

However, *Google* does not want to see following done too quickly. This is because following too quickly could challenge the prior innovative developer's leading time. The shorter leading time an app has, the less return a developer would receive for his investment in innovation. This would discourage developer's motivation on innovation, thereby reducing the average expectation of return on investment from innovation. *Google* also does not want to see following done too slowly. The switch cost is low for apps in *Google Play Store*, and users would like to see new apps. If the following is too slow, users have to wait for a long time before playing with a new app. Therefore, an optimal following speed is the quality that *Google* wants to select and then broadcast the desire to the whole developer community.

9.1.6 Who To Follow?

Following speed is important to platform selection, and how to follow is even more critical for the success as it will influence market performance. Following ordinal developers would be risky as they are more likely to make the wrong decision. Ordinal developers are more interested on short term benefit, so their low-quality apps may damage to the user experience. Also, it may cause a disorder in the market with behaviour like copying or duplication. The number of ordinal developers is huge in a crowded market, and they will bring much competition pressure via following. This is extremely accurate, especially when a developer is selected by *Google*; other ordinal developers may follow him to have many similar apps which will significantly damage his market performance when it should not.

However, by following a good developer, you get the opposite result. Good developers generally have strong research ability, so their innovation activities are typically driven by high expectation on market potential. With such strong research ability, good developers can also respond quicker to the rapid market change and technology development. Good developers have more interests in business with a large market size and long term competitive advantage. Thus, following good developers would have less risk on market failure. The disadvantage to follow good developer is to face stiff competition.

Nevertheless, good developers have a strong competitive advantage due to their market establishment. Some good developers are even monopolistic and hold a large market share already. The good news is that the total number of good developers is not huge in the app economy. However, good developers are more interested in the large market section. Therefore, the developers still have a lot of chances to win on niche market section.

Developers can learn more from good developers on innovation. The innovation knowledge gained by following good developers will also help developers to grow from niche market

quickly and robustly. The definite advantage of following good developers does not change much after *Google* selects a developer. When a developer is selected, he has already grown into an active player in the competition. Other good developers' innovation activities will save him research costs on innovation. Even if he is the first entrant in the market, other developers following him will bring more users and increase the product variety. This will help him to expand the market size and claim the market collectively. The competition between good developers will speed up the innovation. *Google* desires developers to follow good developers so that *Google* can use good developers to signal its' desire to other developers. In that case, it would save *Google*'s work by focusing more on innovation management on good developers.

9.2 The Answer To Research Question

This thesis discusses the model to explain how developers signal for platform selection in the app economy in theory and then empirically explores evidence to support the theory. I have qualitatively studied developers' privacy policy and business model to frame above explanation. I also scraped *Google Play Store* and other online source to have information for 93% of totals apps in *Google Play Store* and their developers. The statistical results show the evidence that uniqueness (how unique is a app to other apps) will increase the chances for selection, but too much uniqueness would reduce the chances for selection. Developers should balance the uniqueness nature of their apps. The statistical results also demonstrate that diversification strategy would negatively impact the chances to platform selection unless developers have diversification ability. The third evidence from the statistical results is that developers should have an optimistic following speed to be selected. Following ordinal developers would negatively impact market performance, especially when a developer becomes selected. Developers should always follow good developers in order to achieve better market performance.

Via this study, the thesis also aims at understanding how *Google* uses selection mechanism to easily and efficiently manage the innovation in such crowded virtual two-sided app market. *Google*, as a market operator, uses selection mechanism to influence developers to innovate by learning from other good developers. Developers should engage other developers to compete and collaboratively develop the *Android* platform into a healthy ecosystem. This signaling effect enables *Google* to focus more on influencing good developers and then spreading the sustainable competition culture among developers' community to enhance the competitiveness of another rival market. The app economy is becoming popular and it is forecasted that in the coming years more economies of this nature shall emerge. This study aims at making an early systematic study on the growing app economy that will be valuable for future researches.

Based on above discussion, now is the time to answer the research question framed in Section 2.3.2.

9.2.1 what costly actions generate signals in app economy?

The literatures review process tells that: 1) developers can differentiate the product with new features, 2) developers can diversify into different product domains. To achieve these goals, developers can 1) explore and exploit to search for innovation, or 2) learn from the rivals, or 3) change their business models.

The qualitative study finds that developer can use user data confidently. Developers have to plan the costs and resources to store and process a large amount of user data collecting from their apps. It would be risk if using user data improperly. Developers should demonstrate to use the minimum number of user data in their privacy policy by adjusting to their business model.

The quantitative study finds that developers needs to balance how unique their apps would be comparing to other competing apps. Creating a unique app would consume a lot of resources to solve technical difficulty and market uncertainty. Unlike traditional business, it finds that developers should build their specialist in app economy. The last action is that developers should follow other "good" developers to innovate, although the competition can be hard.

9.2.2 who can interpret signals in app economy?

Three parties want to interpret the signals in app economy. The first party is the platform owner who has the responsibility of vetting the quality of the markets. The second party is the final users who want to identify the "good" quality apps in their purchase. The third party are the developers who want to identify who to learn and then follow for innovation.

However, only two parties can interpret the signals. The interpretation of signals needs technical skills. The platform owner and some "good" developers have these technical skills. Some "lead" users may also have the technical skills, but the benefit of their recognition is small to the developers. Therefore, I argue the interpreters are the platform owner and "good" developers.

9.2.3 what feedback would encourage signaling in the app economy?

Signaling is costly, and the feedback should benefit more than the costs for developers to generate the signals. Two kinds of feedback are studied in the thesis. One is the platform selection that the platform owner would feedback to give an award, like a featuring status. This

award will make the public to see the clear distinguish of platform selection. So the platform selection would bring lots of customer accessing.

The second kind of feedback is the peer recognition. If a developer's work or ability is recognized by other "good" developers, his confidence will increase. This recognition confirms the positive estimation of the direction a developer pick. The involvement of "good" developers will also brings lots of users and the market can be expanding quickly. The developers can enjoy the benefits of the increasing on user base. The peer recognition also helps a developer to learn from "good" developers on innovation. In this sense, peer recognition is a long term benefit.

9.3 Contributions

The thesis makes many contributions, and I will discuss some most important ones.

9.3.1 To App Economy

The first contribution is to expand the knowledge of the app economy. The app economy is a new term and is rarely studied in a business academic context. However, it is expected to see an increase of popularity in the near future. The thesis has summarized the common characteristics and significant problems encountered in apps economy as well as it has done an academic study on the *Android* fragmentation. Fragmentation is an interesting concept, as it comes as a sequence of open strategy but have not been widely studied. The thesis has observed the strategy adopted by *Google* and evaluated its efficiency.

This research has been the largest study regarding the app economy topic so far. The *Google Play Store* was firstly studied by computer science scholars. Their purpose was to do a study on privacy protection and user experience and their sample size was much smaller. In contrast, the current study collected 93% of apps in *Google Play Store* with the purpose of exploring business management theory. The research method has been demonstrated regarding how to collect data from such large app economy and the analysis method used will also contribute to future analysis.

9.3.2 To Platform Strategy

It also contributes to expand platform strategies in a duopoly context. Existing platform strategies were designed to establish and increase the growth of the markets. The duopoly is recently formed, and is unlikely to turn to monopoly as a single-homing environment. [Eisenmann et al. \(2006\)](#) summarized the three conditions in order to have a single-homing

environment: 1) Multi-homing costs are high for at least one user side. 2) Network effects are positive and strong, at least for the users on the side of the network with high multi-homing costs. 3) Neither side's users have a strong preference for special features. Therefore, the study of duopoly in this thesis will be valuable.

The signaling selection model is supplement to solve the limitation of existing platform strategy. It differentiates the complements based on the innovation incentives, rather than criteria. Developers can not change their strategic behaviour, as it is binding to costs and ability. It also emphasis on the importance of new entrants who are potentially good, for example, user entrepreneurs, open source developers, etc. This kind of complementors is less studied by existing strategies.

The thesis has extended the platform management literature by outlining platform selection strategy. This provides a strategy to use for the market operator's innovation management strategy. Rather than governing on everyone, the platform owner should use a selection mechanism for guidance on platform development. The thesis has confirmed the efficiency of this selection signal.

Cusumano (2010) argued to call for more empirical research on platform strategies. Most existing research on platform strategies are based on theoretical economy research, and did not applied into empirical study. Influential early work by economists mostly took the form of theory and models with few detailed case studies. This is still a relatively new topic and there are very few studies with large samples. This study fills this gap by provide a large scale empirical study.

9.3.3 To Strategic Behaviour

There are many literatures study on developer's strategy behaviours in traditional software innovation. However, these studies are fragmented. The thesis systematically reviews the strategic behaviours discussed in the literatures and innovatively treat them as signals. It also contributes to update with latest strategic behaviours in app economy.

The prior researches look at the strategic behaviour in a macro scope. They assume that developers should response same to the platform strategy. I argue that this assumption is not always safe, especially in the duopoly context of app economy. Competence developers have to behave unique from less-competence developers in order to communicate their competence. This argue shift the focus of platform strategy and promote the micro study on strategic behaviour.

Existing literature has attempted at explaining diversification. This study has aimed to expand knowledge in order to suggest that diversification is not the first logical choice in an extremely competitive environment. Instead, professionals and specialists are more welcomed

by the market operator and users. Developers can learn from diversification, but they should focus on one area to firmly hold their market positions. This will enable developers to be selected and diversification will be more successful and efficient after they have been selected.

9.3.4 To Signaling Theory

One of the major contributions made in the thesis is to successfully apply signaling theory to solve the communication problem in the app economy. Signaling theory has already been applied in many areas, but this is presumably to be the first time to be applied in the app economy. The thesis studies the signals adopted by developers to disclose their innovation ability. In addition to this, the efficiency of signaling to the market operator and consumers is also tested. Crowdedness can expand the gap of information asymmetry, and signaling theory could attract more attention in the future for other large two-sided markets.

Prior research studies the crowding out effect. In addition to this, the thesis promotes the shift of competition into cooperation. The crowdedness generates the opportunity on learning and encourage the developers to develop the markets in collaborative efforts. The thesis proofs that developers should use *following* strategy to collaboratively develop and claim the market. However, in order to drive this culture, the market operator should take the responsibility to protect the investment return on innovation and provide a “fair” competition.

9.3.5 To Others

The thesis also contributes on research methodology. It solves a common challenge in the two-sided market research, which is to understand the historical changes in a two-sided market. The thesis scraped multiple online data sources, and developed a software algorithm to integrate the data sources to recap the *Google*’s decision making history in platform selection and also the trajectory of developers’ app creation activities. This research method should be inspiring for future research to understand the historical changes of an online market.

9.4 Summary

When a duopoly is established, complementors are attracted into its two-sided platform based markets. The competition between complementors can be extremely high, and innovative complements become the key assets for platforms (Lin et al., 2011). How to differentiate the complements on innovativeness becomes a difficult challenge.

The existing platform strategies focus on the motivation of attractions for innovative complements (Rysman, 2009) and setting up barriers to increase the opportunity cost for non-

innovative complements (Pehrsson, 2009). These strategies play important roles to enhance the network effects and increase the growth of the market (Gawer and Cusumano, 2013). They aim to identify the quality of complements based on complementers' current market establishment and status (Evans and Schmalensee, 2010), but lack the ability to protect the "weak" complementers like new entrants who are potentially innovative. The duopoly of app markets has recently formed by *Android* and *iOS*. Many innovative apps are created by "new" developers and "accidental" users (Shah and Tripsas, 2007). At the same time, open strategies have been widely adopted by current popular platforms, but they generate "fragmentation" issue and weakens the platform's governance power. Therefore, it is necessary to develop an efficient platform strategy to differentiate the complements based on their intrinsic innovativeness in such duopoly context.

The thesis develops a signaling selection model based on signaling theory (Spence, 1973) to solve the problem in app economy. No matter complementors are established or newly join in the market, there are patterns of their strategic behaviour when they do innovative activities. For example, they can establish or maintain the competitive advantage via product (like differentiation and diversification), knowledge (like learning) and process (like business model). These strategic behaviour are costly to make and only innovative complementers are likely to make as they have lower opportunity cost comparing to non-innovative complementors (Connelly et al., 2011). This distinction enables the platform to observe those costly actions as a signal of innovativeness and select some of them to feedback with great benefits (Stiglitz, 2002). The benefits from the feedback are greater than the opportunity cost that complementors paid to engage in the innovative activities, so innovative complementors will be motivated to continue on signaling. Other non-selected complementors are motivated to learn from these selected complementors to increase innovativeness in order to generate signals. This learning culture enables the platform to select a small number of innovative complementors to influence other complementors.

The signaling selection model is observed from the featuring selection mechanism in *Google Play Store*. *Google Play Store* is a two-sided app market for *Android*, and also one of the largest mobile app economies in the world. *Android*'s open strategy attracts a large number of developers to create massive number of apps, which has made *Google Play Store* over crowded. Developers have to signal their real quality on technology competence in order to be selected by *Google Play Store* as featured developers. I collect a panel data set composed 93% of all apps and their developers in *Google Play Store*, and the thesis empirically studies the strategic behaviour of all featured developers in the data set with both qualitative and quantitative methods.

In addition to the strategic behaviour identified from the literatures, I identify three specific costly actions for signals in app economy, including uniqueness (how unique is a app comparing to rivals), diversification and following. The findings indicate that uniqueness of products will significantly increase the chance of platform selection, but too much uniqueness will make negative impact. Developers are encouraged to specialize in what they are good at rather than doing diversification. Specialism will increase the chance of platform selection, and diversification will only positively contribute when developer is strong enough to confidently distribute their resource on multiple products. A proper following speed is also positive associate to the likelihood of platform selection. The statistical result suggests that developers should follow good developers to innovate on product and claim the market. In the platform management aspect, the signaling channel enables *Google* to efficiently manage the innovation via selection mechanism. *Google* can influence the whole developer community by focusing on selecting a small number of good developers. Other developers will follow these selected good developers to learn how to innovate in order to compete for selection opportunities.

This study promotes the study on "weak" complementers in platform strategies, and develops a signaling selection model by applying the signaling theory into app economy for the first time. It has expanded the platform knowledge with an update of the current and future coming duopoly economy context. Most existing platform research are based on theoretical modeling or small scale of qualitative study (Cusumano, 2010). The quantitative study on large scale of dataset in this thesis has provided a guidance to future empirical study (see Figure A.38).

9.5 Limitations

The first limitation is the availability of data. Market performance in this study does not form the real count of app installation. I calculated market performance in this study based on the assumption that there is an average portion of users who will review to the apps. However, this assumption is not true for every developer, and therefore, such calculation would have been biased. For example, popular apps should receive more user review, and also the distribution of reviewers is also not equal for each app. Another issue was the inability to track the market performance for each app when other apps were released. The collected market performance data represented the 2014 snapshot of *Google Play Store*, but it failed to provide the data when other apps were released in the market. The 2014 snapshot can reflect the market performance difference between apps and developers, but the sample fails to reflect the real difference in each event. Also, it prevents me to study the other direction of signalling, such as consumer-centric and self-signalling (*proving oneself*). Because, in order to make the analysis valid, I need to have *consumer interaction* measurement as independent variables, and then developers'

signalling behaviours as dependent variable. I can measure the signalling behaviours which is addressed in this thesis, however, I do not have "consumer" data at the time of producing each signalling behaviours. The "consumer feedback" and market performance data I have is only a 2014 snapshot. Therefore, I can not link the signalling behaviours with historical changes of "consumer feedback". Further problems were created by the listing of the most similar 16 apps to each target app by *Google*. Therefore, the data has not included every app that is similar to the target app, which creates a bias. *Google* also did not disclose the exact criteria to select the similar apps; therefore, there was insufficient scope for carrying out a robust test.

The second limitation during the course of study was related to app permission. There are some user data that are not relying on app permission to collect and analyse. The most commonly used permission is to access the Internet. Developers can transmit user data directly through Internet or use other third party library. This external use is not captured in my data set. Although classification of app permission is done under three user data groups, it still fails to include all the user data demanded by the developers in the app. Meanwhile, there is no standard way to describe the privacy policy and each developers drafts the privacy policy statement in his own ways. Therefore, the information attracted from privacy policy is not equally represented and it results in a biased qualitative analysis. Due to the inability to track the use of third party service in each app, the level of innovativeness cannot be tracked. This increases the difficulty to explore the business models at a very accurate and detailed level.

The third limitation is related to developers' strategy. The current study mainly focuses on the competition within a single market. However, many developers are developing apps for multiple platforms. This cross-platform strategy largely influences their behaviour on a single platform. For example, developers may treat competitors, quality of apps and competitive advantage differently. Due to this data limitation, it is difficult to distinguish these developers from those who are only creating apps for *Android*. It has to be assumed that all developers are equally to only compete within *Google Play Store*, and this in return may cause bias.

The fourth limitation is on confirmation. While studying *Google's* selection strategy, it become evident that *Google* never disclosed to the public how it makes selection decision and its related criteria. This shortcoming makes it difficult to test the findings.

9.6 Future Works

This research can be extended in several directions (see Figure A.38). The first direction is to expand the concept of uniqueness. The current angle of uniqueness is based on permission and user data. However, uniqueness can also be based on functionality and the third party libraries in use. Having this information would improve the accuracy in measuring uniqueness.

The improved uniqueness measurement potentially generates more opportunity to analyse the technology dependency and learning among developers.

The second direction involves understanding the pattern of diversification. Even though *Google* want developers to specialise in what they are doing, diversification is still a popular strategy that has been adopted by lots of developers. It is interesting to know when a developer shifts to a new app category he never knew, and when to move back to what he was originally good at. One motivation for developers to diversify is to experiment with several app categories in order to find the best category they can work on. I am interested to know whether a developer learns in order to diversify or diversifies in order to learn. I am not able to answer this question in my current dataset, but it can be done by selecting a sample of developers and conducting interviews in order to recap their history in app development. Meanwhile, whether *Google* can distinguish developers' diversification experiments or not is an interesting question regarding to the platform strategy. I already found the U-shaped relationship between diversification and selection, but it is interesting to study how *Google*, as a receiver, interprets diversification signals. This study provides more strategic guidance to balance between diversification and specialisation. Meanwhile there is a stream of research all focusing on unrelated diversification (Ng, 2007). The data from the *Google Play Store* can be tested on the adoption and the efficiency of unrelated diversification.

So far, I have quantitatively studied three factors that constitute the signal. There are other factors also believed to be important. Those factors are not only necessary for innovation. For example, it is interesting to know a developer's educational background, funding history, social capital, etc. It is observed that the founder of some start-ups may also have a strong relationship to other start-ups, all of which are live in *Google Play Store*. It would be interesting to know whether this factor will lead to similarities between start-ups. Another observation is that a company can have several names in the *Google Play Store*. For example, the developer of the famous game "Angry Birds" has two names in the *Google Play Store*. One is "Rovio Entertainment Ltd"¹ and the other is "Rovio Stars Ltd."². Are these two names competing or collaborating with each other? It would be interesting to study more.

My focus in this study is on good developers who are selected by *Google*. I had studied how they behave to be selected. It would also be interesting to study average developers, who are not selected by *Google*, to find out why they are not featured. This future work can benchmark findings from this thesis. This thesis studied and suggested that developers should always follow good developers who are selected. However, some good developers are on the way to being selected. They are like 'rising stars' who focus more on niche markets and are innovative.

¹<https://play.google.com/store/apps/dev?id=9133452689932095671>

²<https://play.google.com/store/apps/developer?id=Rovio+Stars+Ltd.>

Following them will also be good, and could provide direct suggestions to the business you are working on. The *Google Play Store* has a list of popular apps but their developers are not featured as “Top Developer”. Monitoring the changes on this list can provide insights on the strategy to follow “rising stars”.

I have studied how users interpret signals to make purchasing decisions and then feedback with their satisfaction information. It would also be interesting to see whether this satisfaction will change the developer’s competition strategy. There is a stream of scholars ([Hossain and Quaddus, 2012](#)) studying the users motivation of continually using information products after they accept a technology. The model they built is called “Expectation Confirmation Theory (ECT)”. ECT is to explain post-purchase or post-adoption satisfaction as a function of expectations, perceived performance and dis-confirmation of beliefs. I am interested to explore the dynamic impact of ECT. If previous apps achieve good results with high user satisfaction, it would take a market share and your later apps could be less competitive. However, the user satisfaction of previous apps would increase the expectation of your following apps, which would actually encourage users to try and buy your apps. This dynamic impact is interesting because developers are not solely competing with each other, but collectively working to satisfy users to encourage users for continued active behaviour.

One more future direction is to study the spatial difference of strategic behaviour. In this study, I treat developers from all countries the same. This assumption ignored the facts that the numbers of featured developers from different countries are not equally distributed. It will be interesting to know what makes some countries have higher number of featured developers? Is it because of their education and political policy? Or is it because a particular third party app market becomes popular and so attracts less developers to release their apps in the *Google Play Store*? In this context, the difference of global and local platform strategies to a dominant technology platform can be studied and benchmarked.

References

- Adams, R., Bessant, J., and Phelps, R. (2006). Innovation management measurement: A review. *International Journal of Management Reviews*, 8(1):21–47.
- Aghion, P. and Griffith, R. (2008). Competition and growth: reconciling theory and evidence.
- Ahlers, G. K., Cumming, D. J., Günther, C., and Schweizer, D. (2015). Signaling in Equity Crowdfunding. *Entrepreneurship Theory and Practice*, 39:955—980.
- Aitamurto, T. and Lewis, S. C. (2013). Open innovation in digital journalism: Examining the impact of Open APIs at four news organizations. *New Media & Society*, 15:324—331.
- Akerlof, G. a. (1970). The Market for "Lemons": Quality Uncertainty and the Market Mechanism. *The Quarterly Journal of Economics*, 84(3):488–500.
- Amelio, A. and Jullien, B. (2012). Tying and freebies in two-sided markets. *International Journal of Industrial Organization*, 30(5):436–446.
- Anderson, C. (2009). *Free: The Future of a Radical Price*. Random House.
- Anderson, I., Maitland, J., Sherwood, S., Barkhuus, L., Chalmers, M., Hall, M., Brown, B., and Muller, H. (2007). Shakra: Tracking and Sharing Daily Activity Levels with Unaugmented Mobile Phones. *Mobile Networks and Applications*, 12(2-3):185–199.
- Anderson, P. and Tushman, M. L. (1990). Technological Discontinuities and Dominant Designs: A Cyclical Model of Technological Change. *Administrative Science Quarterly*, 35(4):604–633.
- Anderson, R. M., Ellison, G., and Fudenberg, D. (2010). Location choice in two-sided markets with indivisible agents. *Games and Economic Behavior*, 69(1):2–23.
- Anderson, S. P. and Gabszewicz, J. J. (2006). The media and advertising: a tale of two-sided markets. *Handbook of the Economics of Art and Culture*, 1:567–614.
- Andrew, J. P., Sirkin, H. L., and Butman, J. (2013). *Payback: reaping the rewards of innovation*. Harvard Business Press.
- Ansoff, I. (1957). Strategies for diversification. *Havard Business Review*, 35(5):113–124.
- Argyris, C. and Schön, D. A. (1978). Organizational Learning: A Theory of Action Perspective. *The Journal of Applied Behavioral Science*, 15(4):542–548.

- Armstrong, M. (2006). Competition in two sided markets. *The RAND Journal of Economics*, 37:668–691.
- Armstrong, M. and Wright, J. (2007). Two-sided markets, competitive bottlenecks and exclusive contracts. *Economic Theory*, 32(2):353–380.
- Baker, H. K. and Kiymaz, H. (2011). *The Art of Capital Restructuring: Creating Shareholder Value through Mergers and Acquisitions*. Wiley.
- Baldwin, C., Hienert, C., and von Hippel, E. (2006). How user innovations become commercial products: A theoretical investigation and case study. *Research Policy*, 35(9):1291–1313.
- Baldwin, C. Y. and Clark, K. B. (2006). The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model? *Management Science*, 52(7):1116–1127.
- Baldwin, C. Y. and Woodard, C. J. (2009). The Architecture of Platforms: A Unified View. In Gawer, A., editor, *Platforms, Markets and Innovation*. Edward Elgar Publishing, Cheltenham, U.K.
- Barnett, W. P. (1997). The Dynamics of Competitive Intensity. *Administrative Science Quarterly*, 42(1):128–160.
- Barney, J. (1991). Firm Resources and Sustained Competitive Advantage. *Journal of Management*, 17(1):99–120.
- Barney, J. (1995). Looking inside for competitive advantage. *Academy of Management Perspectives*, 9(4):49–61.
- Barrera, D., Clark, J., and McCartney, D. (2012). Understanding and improving app installation security mechanisms through empirical analysis of android. In *ACM workshop on Security and privacy in smartphones and mobile devices*, pages 81–92.
- Basole, R. C. and Karla, J. (2011). On the evolution of mobile platform ecosystem structure and strategy. *Business and Information Systems Engineering*, 3(5):313–322.
- Basole, R. C. and Karla, J. (2012). Value Transformation in the Mobile Service Ecosystem: A Study of App Store Emergence and Growth. *Service Science*, 4(1):24–41.
- Bauer, J. M. (2014). Platforms, systems competition, and innovation: Reassessing the foundations of communications policy. *Telecommunications Policy*, 38:662–673.
- Bebchuk, L. A. and Fried, J. M. (2005). Pay Without Performance: Overview of the Issues. *Journal of Applied Corporate Finance*, 17(4):8–23.
- Bedard, K. (2001). Human Capital versus Signaling Models: University Access and High School Dropouts. *Journal of Political Economy*, 109(4):749–775.
- Belleflamme, P. and Peitz, M. (2010). Platform competition and seller investment incentives. *European Economic Review*, 54(8):1059–1076.
- Bessen, J. (2006). Open Source Software: Free Provision of Complex Public Goods. In *The Economics of Open Source Software Development*, pages 57–81.

- Bhattacharya, S., Krishnan, V., and Mahajan, V. (1998). Managing New Product Definition in Highly Dynamic Environments. *Management Science*, 44(11):50–64.
- Bliege Bird, R. and Smith, E. A. (2005). Signaling Theory, Strategic Interaction, and Symbolic Capital. *Current Anthropology*, 46(2):221–248.
- Blundell, R., Griffith, R., and Van Reenen, J. (1999). Market share, market value and innovation in a panel of British manufacturing firms. *The Review of Economic Studies*, 66(3):529–554.
- Bogers, M., Afuah, A., and Bastian, B. (2010). Users as Innovators: A Review, Critique, and Future Research Directions. *Journal of Management*, 36(4):857–875.
- Bonner, J. M. and Walker, O. C. (2004). Selecting influential business-to-business customers in new product development: Relational embeddedness and knowledge heterogeneity considerations. *Journal of Product Innovation Management*, 21(3):155–169.
- Boudreau, K. (2008). Does opening a platform stimulate innovation? effects on modular and systemic innovation. *Organization*, (West 2003):1–33.
- Boudreau, K. (2010). Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. *Management Science*, 56(10):1849–1872.
- Boudreau, K. (2012). Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation. *Organization Science*, 23(5):1409–1427.
- Boudreau, K. and Lakhani, K. (2009). How to Manage Outside Innovation. *MIT Sloan Management Review*, 50(4):68–77.
- Boudreau, K. and Lakhani, K. (2013). Using the crowd as an innovation partner. *Harvard business review*, 91(4).
- Brockhoff, K. (2003). Customers’ perspectives of involvement in new product development. *International Journal of Technology Management*, 26(5/6):464.
- Brockmann, T., Stieglitz, S., and Cvetkovic, A. (2015). Prevalent Business Models for the Apple App Store.
- Brousseau, E. and Penard, T. (2007). The Economics of Digital Business Models: A Framework for Analyzing the Economics of Platforms. *Review of Network Economics*, 6(2):81–114.
- Buur, J. and Matthews, B. (2008). Participatory innovation. *International Journal of Innovation Management*, 12(03):255–273.
- Caillaud, B. and Jullien, B. (2003). Chicken & Egg: Competition among Intermediation Service Providers. *The RAND Journal of Economics*, 34(2):309–328.
- Calantone, R. J., Cavusgil, S. T., and Zhao, Y. (2002). Learning orientation, firm innovation capability, and firm performance. *Industrial Marketing Management*, 31(6):515–524.
- Campbell-Kelly, M., Garcia-Swartz, D., Lam, R., and Yang, Y. (2014). Economic and business perspectives on smartphones as multi-sided platforms. *Telecommunications Policy*.

- Carlin, W., Fries, S., Schaffer, M. E., and Seabright, P. (2001). Competition and enterprise performance in transition economies: Evidence from a cross-country survey.
- Caves, R. E. (2003). Contracts Between Art and Commerce. *Journal of Economic Perspectives*, 17(2):73–83.
- Cecere, G., Corrocher, N., and Battaglia, R. D. (2015). Innovation and competition in the smartphone industry: Is there a dominant design? *Telecommunications Policy*, 39(3):162–175.
- Cennamo, C. and Santalo, J. (2013). Platform competition: Strategic trade-offs in platform markets. *Strategic Management Journal*, 34(11):1331–1350.
- Certo, S. T., Daily, C. M., and Dalton, D. R. (2001). Signaling Firm Value Through Board Structure: An Investigation of Initial Public Offerings. *Entrepreneurship Theory and Practice*, 26:33–50.
- Charland, A., Leroux, B., Charland, D., and Leroux, B. (2011). Mobile Application Development: Web vs. Native. *Communications of the ACM*, 54(5):0–5.
- Chen, M.-J. and Miller, D. (2012). Competitive Dynamics: Themes, Trends, and a Prospective Research Platform. *The Academy of Management Annals*, 6(1):135–210.
- Chesbrough, H. (2007a). Business model innovation: it's not just about technology anymore. *Strategy & Leadership*, 35(6):12–17.
- Chesbrough, H. (2010). Business Model Innovation: Opportunities and Barriers. *Long Range Planning*, 43(2-3):354–363.
- Chesbrough, H. and Crowther, A. K. (2006). Beyond high tech: early adopters of open innovation in other industries. *R&D Management*, 36(3):229–236.
- Chesbrough, H. W. (2003a). *Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press.
- Chesbrough, H. W. (2003b). The new business logic of open innovation. *Harvard Business Review*, pages 3–5.
- Chesbrough, H. W. (2007b). Why companies should have open business models. *MITSloan Management Review*, 48(2):22+.
- Chia, P., Yamamoto, Y., and Asokan, N. (2012). Is this app safe? a large scale study on application permissions and risk signals. In *Proceedings of the 21st International Conference on World Wide Web*, pages 311–320, Lyon, France. ACM.
- Chin, E., Felt, A. P. A., Greenwood, K., and Wagner, D. (2011). Analyzing inter-application communication in Android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services - MobiSys '11*, pages 239–252, New York, New York, USA. ACM Press.
- Choi, J. P. (2010). Tying in two-sided markets with multi-homing. *Journal of Industrial Economics*, 58(3):607–626.

- Christensen, C. M., Baumann, H., Ruggles, R., and Sadtler, T. M. (2006). Disruptive innovation for social change. *Harvard Business Review*, 84(12).
- Clemons, E. K. (2009). Business Models for Monetizing Internet Applications and Web Sites: Experience, Theory, and Predictions. *Journal of Management Information Systems*, 26(2):15–41.
- Cohen, B. D. and Dean, T. J. (2005). Information asymmetry and investor valuation of IPOs: Top management team legitimacy as a capital market signal. *Strategic Management Journal*, 26(7):683–690.
- Connelly, B. L., Certo, S. T., Ireland, R. D., and Reutzel, C. R. (2011). Signaling Theory: A Review and Assessment. *Journal of Management*, 37(1):39–67.
- Cusumano, M. (2010). Technology strategy and management - The evolution of platform thinking. *Communications of the ACM*, 53(1):32.
- Dahlander, L. and Gann, D. M. (2010). How open is innovation? *Research Policy*, 39(6):699–709.
- Dahlander, L. and Magnusson, M. (2008). How do Firms Make Use of Open Source Communities? *Long Range Planning*, 41(6):629–649.
- Damiano, E., Hao, L., Damiano, E., and Forthcoming, H. L. (2008). Competing Matchmaking. *Journal of the European Economic Association*, 6(4):789–818.
- Deb, P. (2013). Signaling Type and Post-IPO Performance. *European Management Review*, (10):99–116.
- Deci, E. L. and Ryan, R. M. (2010). The "What" and "Why" of Goal Pursuits: Human Needs and the Self-Determination of Behavior. *Psychological Inquiry*, 11(4):227–268.
- Dixon, N. M. (1992). Organizational learning: A review of the literature with implications for HRD professionals. *Human Resource Development Quarterly*, 3(1):29–49.
- Durlauf, S. N. and Blume, L. E. (2008). The new Palgrave dictionary of economics.
- Economides, N. and Katsamakos, E. (2006). Two-Sided Competition of Proprietary vs. Open Source Technology Platforms and the Implications for the Software Industry. *Management Science*, 52(7):1057–1071.
- Egelman, S. (2013). My profile is my password, verify me!: the privacy/convenience tradeoff of facebook connect. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2369–2378.
- Eisenmann, T., Parker, G., and Van Alstyne, M. (2008). Opening Platforms: How, When and Why? *Working Papers – Harvard Business School Division of Research*, pages 1–27.
- Eisenmann, T., Parker, G., and Van Alstyne, M. (2011). Platform envelopment. *Strategic Management Journal*, 1285(12):1270–1285.
- Eisenmann, T., Parker, G., and Van Alstyne, M. W. (2006). Strategies for Two-Sided Markets. *Harvard Business Review*, 84(10):12.

- Enkel, E. and Gassmann, O. (2010). Creative imitation: Exploring the case of cross-industry innovation. *R&D Management*, 40(3):256–270.
- Enkel, E., Gassmann, O., and Chesbrough, H. (2009). Open R&D and open innovation: Exploring the phenomenon. *R and D Management*, 39(4):311–316.
- Enkel, E., Kausch, C., and Gassmann, O. (2005). Managing the risk of customer integration. *European Management Journal*, 23(2):203–213.
- Evans, D. S. (2003). The Antitrust Economics of Multi-Sided Platform Markets. *Yale Journal on Regulation*, 20:325.
- Evans, D. S. (2009). How catalysts ignite: The economics of platform-based start-ups. *Platforms, markets and innovation*, pages 99–128.
- Evans, D. S., Hagiu, A., and Schmalensee, R. (2006). *Invisible engines: how software platforms drive innovation and transform industries*. MIT Press Cambridge.
- Evans, D. S. and Schmalensee, R. (2007). The Industrial Organization of Markets with Two-Sided Platforms. *Competition Policy International*, 3(1):150—179.
- Evans, D. S. and Schmalensee, R. (2010). Failure to Launch: Critical Mass in Platform Businesses. *Review of Network Economics*, 9(4).
- Farrell, J. and Klemperer, P. (2007). Coordination and lock-in: Competition with switching costs and network effects. *Handbook of industrial organization*, 3:1967–2072.
- Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. (2011a). Android permissions demystified. *Proceedings of the 18th ACM conference on Computer and communications security - CCS '11*, page 627.
- Felt, A. P., Finifter, M., and Chin, E. (2011b). A survey of mobile malware in the wild. In *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices. ACM, 2011.*, pages 3–14.
- Felt, A. P., Greenwood, K., and Wagner, D. (2011c). The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*. USENIX Association.
- Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. (2012). Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security - SOUPS '12*, page 1, New York, New York, USA. ACM Press.
- Fichman, R. G. and Kemerer, C. F. (1997). The Assimilation of Software Process Innovations: An Organizational Learning Perspective. *Management Science*, 43(10):1345–1363.
- Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine.
- Filatotchev, I. and Bishop, K. (2002). Board composition, share ownership, and underpricing' of U.K. IPO firms. *Strategic Management Journal*, 23(10):941–955.

- Fitzgerald, B. (2006). The Transformation of Open Source Software. *MIS Quarterly*, 30(3):587.
- Frakes, W. and Kang, K. (2005). Software reuse research: status and future. *IEEE Transactions on Software Engineering*, 31(7):529–536.
- Franke, N. and Shah, S. (2003). How communities support innovative activities: an exploration of assistance and sharing among end-users. *Research Policy*, 32(1):157–178.
- Franz, K. (2006). Tinkering: Consumers Reinvent the Early Automobile. *The Business History Review*, 80(2):370–373.
- Fu, B., Lin, J., Li, L., and Faloutsos, C. (2013). Why people hate your app: making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 9, Chicago, Illinois, USA. ACM.
- Gassmann, O. (2006). Opening up the innovation process: towards an agenda. *R&D Management*, 36(3):223–228.
- Gat, I. and Succi, G. (2013). A Survey of the API Economy. *Cutter Consortium*.
- Gawer, A. (2011). *Platforms, markets and innovation*. Edward Elgar Publishing.
- Gawer, A. and Cusumano, M. (2002). Platform leadership: How Intel, Microsoft and Cisco drive industry innovation. *Harvard Business School Press*.
- Gawer, A. and Cusumano, M. A. (2008). How Companies Become Platform Leaders. *Management Review*, 49(2):28–35.
- Gawer, A. and Cusumano, M. A. (2013). Industry Platforms and Ecosystem Innovation.
- Gawer, A. and Henderson, R. (2007). Platform owner entry and innovation in complementary markets: Evidence from Intel. *Journal of Economics and Management Strategy*, 16(1):1–34.
- Genakos, C. and Valletti, T. (2012). Regulating prices in two-sided markets: The waterbed experience in mobile telephony.
- Ghazawneh, A. and Henfridsson, O. (2012). Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, 22(2):1–29.
- Ghose, A. and Han, S. P. (2014). Estimating Demand for Mobile Applications in the New Economy. *Management Science*, 60(6):1470–1488.
- Gilbert, R. (2006). Looking for Mr. Schumpeter: Where are we in the competition-innovation debate? *Innovation Policy and the Economy*, 6(Griliches 1992):159–215.
- Gonçalves, V. and Ballon, P. (2011). Adding value to the network: Mobile operators’ experiments with Software-as-a-Service and Platform-as-a-Service models. *Telematics and Informatics*, 28(1):12–21.
- Goode, S. (2005). Something for nothing: Management rejection of open source software in Australia’s top firms. *Information and Management*, 42(5):669–681.

- Goranova, M., Alessandri, T. M., Brandes, P., and Dharwadkar, R. (2007). Managerial ownership and corporate diversification: A longitudinal view. *Strategic Management Journal*, 28(3):211–225.
- Gordon, M. E. (2013). The History of App Pricing, And Why Most Apps Are Free. Technical report.
- Greene, W. H. (2003). *Econometric analysis*. Pearson Education India.
- Gruber, M., MacMillan, I. C., and Thompson, J. D. (2008). Look Before You Leap: Market Opportunity Identification in Emerging Technology Firms.
- Hagiu, A. (2006). Pricing and commitment by two-sided platforms. *The RAND Journal of Economics*, 37(3):720–737.
- Hagiu, A. (2009a). Quantity vs. quality and exclusion by two-sided platforms. *Harvard Business School Strategy Unit Working Paper*, pages 1–26.
- Hagiu, A. (2009b). Two-sided platforms: Product variety and pricing structures. *Journal of Economics and Management Strategy*, 18(4):1011–1043.
- Hagiu, A. and Lee, R. S. (2011). Exclusivity and control. *Journal of Economics and Management Strategy*, 20(3):679–708.
- Ham, H. K. and Park, Y. B. (2011). Mobile application compatibility test system design for Android fragmentation. In *Communications in Computer and Information Science*, pages 314–320.
- Han, D., Zhang, C., Fan, X., Hindle, A., Wong, K., and Stroulia, E. (2012a). Understanding android fragmentation with topic analysis of vendor-specific bugs. In *Proceedings - Working Conference on Reverse Engineering, WCRE*, pages 83–92.
- Han, K., Oh, W., Im, K. S., Chang, R. M., Oh, H., Pinsonneault, A., and Chang, R. M. (2012b). Value Cocreation and Welath Spillover in Open Innovation Alliances. *MIS Quarterly*, 36(1):291–315.
- Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., and Song, D. (2013). Juxtapp: A scalable system for detecting code reuse among android applications. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7591 LNCS, pages 62–81.
- Harhoff, D., Henkel, J., and Von Hippel, E. (2003). Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations. *Research Policy*, 32(10):1753–1769.
- Harison, E. and Koski, H. (2010). Applying open innovation in business strategies: Evidence from Finnish software firms. *Research Policy*, 39(3):351–359.
- Hart, S. J. G. and D, O. (1983). An Analysis of the Principal-Agent Problem. *Econometrica*, 51(1):7–45.

- Hauge, Ø., Ayala, C., and Conradi, R. (2010). Adoption of open source software in software-intensive organizations - A systematic literature review. *Information and Software Technology*, 52(11):1133–1154.
- Healy, P. M. and Palepu, K. G. (2001). Information asymmetry, corporate disclosure, and the capital markets: A review of the empirical disclosure literature. *Journal of Accounting and Economics*, 31(1-3):405–440.
- Heeringa, W. (2004). *Measuring dialect pronunciation differences using Levenshtein distance*. PhD thesis, University of Groningen, Groningen.
- Hendry, J. (2002). The principal's other problems: Honest incompetence and the specification of objectives. *Academy of Management Review*, 27(1):98–113.
- Hertel, G., Niedner, S., and Herrmann, S. (2003). Motivation of software developers in open source projects: An Internet-based survey of contributors to the Linux kernel. *Research Policy*, 32(7):1159–1177.
- Himanen, P. (2001). *The Hacker Ethic and the Spirit of the Information Age*.
- Hoffmann, W. H. (2007). Strategies for managing a portfolio of alliances. *Strategic Management Journal*, 28(8):827–856.
- Holmstrom, B. (1979). Moral Hazard and Observability. *The Bell Journal of Economics*, 10(1):74–91.
- Holzer, A. and Ondrus, J. (2011). Mobile application market: A developer's perspective. *Telematics and Informatics*, 28(1):22–31.
- Hossain, M. A. and Quaddus, M. (2012). Expectation–confirmation theory in information system research: A review and analysis. In *Information systems theory*, pages 441–469. Springer.
- Huizingh, E. K. (2011). Open innovation: State of the art and future perspectives. *Technovation*, 31(1):2–9.
- Hurley, R. F. and Hult, G. T. M. (1998). Innovation, Market Orientation, and Organizational Learning: An Integration and Empirical Examination. *Journal of Marketing*, 62(3):42–54.
- Huyett, W. and Viguerie, S. (2005). Extreme competition. *McKinsey Quarterly*, pages 64–66.
- Hyrynsalmi, S., Mäkilä, T., Järvi, A., Suominen, A., Seppänen, M., and Knuutila, T. (2012a). App store, marketplace, play! an analysis of multi-homing in mobile software ecosystems. In *CEUR Workshop Proceedings*, volume 879, pages 59–72.
- Hyrynsalmi, S., Seppänen, M., and Suominen, A. (2014). Sources of value in application ecosystems. *Journal of Systems and Software*, 96:61–72.
- Hyrynsalmi, S., Suominen, A., Mäkilä, T., Järvi, A., and Knuutila, T. (2012b). Revenue models of application developers in android market ecosystem. In *Lecture Notes in Business Information Processing*, pages 209–222.

- Ickin, S., Wac, K., Fiedler, M., Janowski, L., Hong, J.-H., and Dey, A. K. (2012). Factors Influencing Quality of Experience of Commonly Used Mobile Applications. *IEEE Communications Magazine*, 50(4):48–56.
- Jansen, S. and Bloemendal, E. (2013). Defining app stores: The role of curated marketplaces in software ecosystems. In *Lecture Notes in Business Information Processing*, pages 195–206.
- Jensen, M. C. and Meckling, W. H. (1976). Theory of the firm: Managerial behavior, agency costs and ownership structure. *Journal of Financial Economics*, 3(4):305–360.
- Jeppesen, L. B. and Frederiksen, L. (2006). Why Do Users Contribute to Firm-Hosted User Communities? The Case of Computer-Controlled Music Instruments. *Organization Science*, 17(1):45–63.
- Jiang, J. J., Klein, G., Hwang, H.-G., Huang, J., and Hung, S.-Y. (2004). An exploration of the relationship between software development process maturity and project performance. *Information and Management*, 41(3):279–288.
- Johnson, W. R. (1978). Overinsurance and Public Provision of Insurance: The Roles of Moral Hazard and Adverse Selection. *Quarterly Journal of Economics*, 92(4):693–696.
- Jung, J., Chan-Olmsted, S., and Kim, Y. (2013). From Access to Utilization: Factors Affecting Smartphone Application Use and Its Impacts on Social and Human Capital Acquisition in South Korea. *Journalism & Mass Communication Quarterly*, 90(4):715–735.
- Kaiser, U. and Wright, J. (2006). Price structure in two-sided markets: Evidence from the magazine industry. *International Journal of Industrial Organization*, 24(1):1–28.
- Kalapesi, C. (2013). Unlocking the value of personal data: From collection to usage. *World Economic Forum, Tech. Rep.*
- Karhu, K., Tang, T., and Hämäläinen, M. (2014). Analyzing competitive and collaborative differences among mobile ecosystems using abstracted strategy networks. *Telematics and Informatics*, 31(2):319–333.
- Katila, R. and Ahuja, G. (2002). Something old, something new: A longitudinal study of search behavior and new product introduction. *Academy of Management Journal*, 45(6):1183–1194.
- Katila, R. and Chen, E. L. (2008). Effects of Search Timing on Innovation: The Value of Not Being in Sync with Rivals. *Administrative Science Quarterly*, 53(4):593–625.
- Katila, R., Rosenberger, J. D., and Eisenhardt, K. M. (2008). Swimming with Sharks: Technology Ventures, Defense Mechanisms and Corporate Relationships. *Administrative Science Quarterly*, 53(2):295–332.
- Kenney, M. and Pon, B. (2011). Structuring the smartphone industry: Is the mobile Internet OS platform the key? *Journal of Industry, Competition and Trade*, 11(3):239–261.
- Keupp, M. M. and Gassmann, O. (2009). Determinants and archetype users of open innovation. *R&D Management*, 39(4):331–341.
- Kiczales, G. (1996). Beyond the black box: open implementation. *IEEE Software*, 13(1):8–10.

- Kim, K. and Viswanathan, S. (2013). The Signals in the Noise: The Role of Reputable Investors in a Crowdfunding Market. In *TPRC 41: The 41st Research Conference on Communication, Information and Internet Policy*, pages 1–31.
- Kirmani, A. and Rao, A. R. (2000). No Pain, No Gain: A Critical Review of the Literature on Signaling Unobservable Product Quality. *Journal of Marketing*, 64(2):66–79.
- Kirzner, I. M. and Discovery, E. (1997). Entrepreneurial Discovery and the Competitive Market Process : An Austrian Approach. *Journal of Economic Literature*, (March):60–85.
- Koch, S. and Kerschbaum, M. (2014). Joining a smartphone ecosystem: Application developers' motivations and decision criteria. *Information and Software Technology*, 56(11):1423–1435.
- Kodama, F. (1993). Technology fusion and the new R&D. *Long Range Planning*, 26(1):154.
- Kotler, P. (2000). Marketing Management , Millenium Edition. *Marketing Management*, 23(6):188–193.
- Kouris, I. and Kleer, R. (2012). Business models in two-sided markets: an assessment of strategies for app platforms. In *2012 International Conference on Mobile Business*, page 22.
- Krishnan, V., Eppinger, S. D., and Whitney, D. E. (1997). A Model-Based Framework to Overlap Product Development Activities. *Management Science*, 43(4):437–451.
- Kuk, G. (2006). Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. *Management Science*, 52(7):1031–1042.
- Lakhani, K. R. and Von Hippel, E. (2003). How open source software works: "free" user-to-user assistance. *Research Policy*, 32(6):923–943.
- Lakomaa, E. and Kallberg, J. (2013). Open Data as a Foundation for Innovation: The Enabling Effect of Free Public Sector Information for Entrepreneurs. *Access, IEEE*, 1:558–563.
- Lang, K. and Kropp, D. (1986). Human capital versus sorting: the effects of compulsory attendance laws. *The Quarterly Journal of Economics*, 101(3):609–624.
- Langlois, R. N. and Robertson, P. L. (1992). Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries. *Research Policy*, 21(4):297–313.
- Laursen, K. and Salter, A. (2006). Open for innovation: The role of openness in explaining innovation performance among U.K. manufacturing firms. *Strategic Management Journal*, 27(2):131–150.
- Lavie, D. and Rosenkopf, L. (2006). Balancing exploration and exploitation in alliance formation. *Academy of Management Journal*, 49(4):797–818.
- Lee, G. and Raghu, T. (2011). Product portfolio and mobile apps success: evidence from App Store market. In *Americas Conference on Information Systems*, page 444.
- Lee, R. S. and Wu, T. (2009). Subsidizing Creativity through Network Design: Zero-Pricing and Net Neutrality. *Journal of Economic Perspectives*, 23(3):61–76.

- Lee, S., Moisa, N., and Weiss, M. (2003). Open source as a signalling device-an economic analysis. Technical report, Working Paper Series: Finance & Accounting, Johann Wolfgang Goethe-Universität Frankfurt a. M.
- Lee, T. and BradLow, E. (2011). Automated Marketing Research Using Online Customer Reviews. *Journal of Marketing Research*, 48:881–894.
- Leesig, L. (2014). *Remix making art and commerce thrive in the hybrid economy*. Epic Publishing.
- Lella, A. and Lipsman, A. (2014). The U.S. Mobile App Report. Technical report, comScore Inc.
- Lerner, J. and Tirole, J. (2002). Some Simple Economics of Open Source. *The Journal of Industrial Economics*, L:197–234.
- Lerner, J. and Tirole, J. (2005). The Economics of Technology Sharing: Open Source and Beyond. *Journal of Economic Perspectives*, 19(2):99–120.
- Lettl, C., Herstatt, C., and Gemuenden, H. G. (2006). Users' contributions to radical innovation: Evidence from four cases in the field of medical equipment technology. *R&D Management*, 36(3):251–272.
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.
- Li, M., Goh, K.-Y., and Cavusoglu, H. (2014). Investigating Developers' Entry To Mobile App Platforms: A Network Externality View. In *Proceedings of the 22nd European Conference on Information Systems*,.
- Lightfoot, G. and Wisniewski, T. P. (2014). Information Asymmetry and Power in a Surveillance Society. *Information and Organization*, 24:214–235.
- Lilien, G. L., Morrison, P. D., Searls, K., Sonnack, M., and von Hippel, E. (2002). Performance Assessment of the Lead User Idea-Generation Process for New Product Development. *Management Science*, 48(8):1042–1059.
- Lim, S. L. and Bentley, P. J. (2012a). App epidemics: Modelling the effects of publicity in a mobile app ecosystem. In *Artificial Life*, volume 13, pages 202–209.
- Lim, S. L. and Bentley, P. J. (2012b). How to be a successful app developer: lessons from the simulation of an app ecosystem. *ACM SIGEVOlution*, 6(1):2–15.
- Lim, S. L., Bentley, P. J., Kanakam, N., Ishikawa, F., and Honiden, S. (2015). Investigating country differences in mobile app user behavior and challenges for software engineering. *Software Engineering, IEEE Transactions on*, 41(1):40–64.
- Lin, J., Sadeh, N., Amini, S., and Lindqvist, J. (2012). Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 501–510.
- Lin, M., Li, S., and Whinston, A. B. (2011). Innovation and Price Competition in a Two-Sided Market. *Journal of Management Information Systems*, 28(2):171–202.

- Linares-Vásquez, M., Bavota, G., Bernal-Cárdenas, C., Di Penta, M., Oliveto, R., and Poshyvanyk, D. (2013). API change and fault proneness: a threat to the success of Android apps. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2013*, page 477.
- Linberg, K. R. (1999). Software developer perceptions about software project failure: A case study. *Journal of Systems and Software*, 49(2):177–192.
- Lins, K. V., Servaes, H., and Tufano, P. (2010). What drives corporate liquidity? An international survey of cash holdings and lines of credit. *Journal of Financial Economics*, 98(1):160–176.
- Liu, C. Z., Au, Y. A., and Choi, H. S. (2014). Effects of Freemium Strategy in the Mobile App Market: An Empirical Study of Google Play. *Journal of Management Information Systems*, 31(3):325–354.
- Lüthje, C. (2004). Characteristics of innovating users in a consumer goods field. *Technovation*, 24(9):683–695.
- MacCormack, A., Rusnak, J., and Baldwin, C. Y. (2006). Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code. *Management Science*, 52(7):1015–1030.
- MacCormack, A. and Verganti, R. (2003). Managing the sources of uncertainty: Matching process and context in software development. *Journal of Product Innovation Management*, 20(3):217–232.
- MacCormack, A., Verganti, R., and Iansiti, M. (2001). Developing Products on "Internet Time": The Anatomy of a Flexible Development Process. *Management Science*, 47(1):133–150.
- Majumdar, S. N. and Nechaev, S. (2005). Exact asymptotic results for the Bernoulli matching model of sequence alignment. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 72(2).
- Mankiw, N. G. (2006). *Principles of Microeconomics*.
- Mantena, R., Sankaranarayanan, R., and Viswanathan, S. (2010). Platform-based information goods: The economics of exclusivity. *Decision Support Systems*, 50(1):79–92.
- March, J. G. (1991). Exploration and Exploitation in Organizational Learning. *Organization Science*, 2(1):71–87.
- Markides, C. and Geroski, P. (2004). *Fast second: How smart companies bypass radical innovation to enter and dominate new markets*. Financial Times.
- Matsusaka, J. G. (2001). Corporate Diversification, Value Maximization, and Organizational Capabilities. *The Journal of Business*, 74(3):409–431.
- Maurer, C. and Tiwana, A. (2012). Control in App Platforms: The Integration-Differentiation Paradox. *ICIS2012*.

- McDonnell, T., Ray, B., and Kim, M. (2013). An empirical study of API stability and adoption in the android ecosystem. In *IEEE International Conference on Software Maintenance, ICSM*, pages 70–79.
- Miller, M. H. and Rock, K. (1985). Dividend Policy under Asymmetric Information. *The Journal of Finance*, 40(4):1031–1051.
- Minelli, R. and Lanza, M. (2013). Software Analytics for Mobile Applications—Insights&Lessons Learned. In *2013 17th European Conference on Software Maintenance and Reengineering*, pages 144–153.
- Miotti, L. and Sachwald, F. (2003). Co-operative R&D: Why and with whom? An integrated framework of analysis. *Research Policy*, 32(8):1481–1499.
- Mody, C. (2006). Corporations, Universities, and Instrumental Communities: Commercializing Probe Microscopy, 1981-1996. *Technology and Culture*, 47(1):56–80.
- Mojica, I. J., Adams, B., Nagappan, M., Dienst, S., Berger, T., and Hassan, A. E. (2014). A large-scale empirical study on software reuse in mobile apps. *IEEE Software*, 31(2):78–86.
- Mojica, I. J., Nagappan, M., Adams, B., and Hassan, A. E. (2012). Understanding Reuse in the Android Market. In *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, pages 113 – 122.
- Möller, A., Michahelles, F., Diewald, S., Roalter, L., and Kranz, M. (2012). Update Behavior in App Markets and Security Implications : A Case Study in Google Play. In *Proceedings of the 3rd International Workshop on Research in the Large. Held in Conjunction with Mobile HCI*, pages 3–6.
- Moorman, C. and Miner, A. S. (1998). Organizational improvisation and organizational memory. *Academy of Management Review*, 23(4):698–723.
- Morris, M., Schindehutte, M., and Allen, J. (2005). The entrepreneur’s business model: Toward a unified perspective. *Journal of Business Research*, 58(6):726–735.
- Nelson, R. R. and Winter, S. G. (2009). *An evolutionary theory of economic change*.
- Ng, D. W. (2007). A modern resource based approach to unrelated diversification. *Journal of Management Studies*, 44(8):1481–1502.
- Nickell, S. J. (1996). Competition and Corporate Performance. *Journal of Political Economy*, 104(4):724.
- Nonaka, I. (1994). A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, 5:14–37.
- Normann, R. and Ramírez, R. (1993). From value chain to value constellation: designing interactive strategy. *Harvard Business Review*, 71(4):65–77.
- O’Donnell, B., Reith, R., and Shirer, M. (2013). Worldwide Smart Connected Device Market Crossed 1 Billion Shipments in 2012, Apple Pulls Near Samsung in Fourth Quarter, According to IDC. Technical report.

- Ogawa, S. (1998). Does sticky information affect the locus of innovation? Evidence from the Japanese convenience-store industry. *Research Policy*, 26(7-8):777–790.
- O'Mahony, S. (2003). Guarding the commons: How community managed software projects protect their work. *Research Policy*, 32(7):1179–1198.
- Pappas, A. (2014). UK App Economy 2014: Sizing the App Economy, Jobs and Opportunities for Developers in Britain. Technical report, VisionMobile Inc.
- Parker, G., Alstyne, M. V., and Van Alstyne, M. (2010). Innovation, openness & platform control. In *Proceedings of the 11th ACM conference on Electronic commerce - EC '10*, volume 4684-08, page 95.
- Parker, G. and Van Alstyne, M. (2000). Information complements, substitutes, and strategic product design. *Proceedings of the twenty first international conference on Information systems*, pages 13–15.
- Parker, G. G. and Van Alstyne, M. W. (2005). Two-sided network effects: A theory of information product design. *Management Science*, 51(10):1494–1504.
- Parrish, E. D., Cassill, N. L., and Oxenham, W. (2006). Niche market strategy for a mature marketplace. *Marketing Intelligence & Planning*, 24(7):694–707.
- Pearce, P., Felt, A. P., Nunez, G., and Wagner, D. (2012). AdDroid: Privilege separation for applications and advertisers in Android. In *Proceedings of AsiaCCS*.
- Pehrsson, A. (2004). Strategy competence: a successful approach to international market entry. *Management Decision*, 42(6):758–768.
- Pehrsson, A. (2009). Barriers to entry and market strategy: a literature review and a proposed model. *European Business Review*, 21(1):64–77.
- Peppard, J. and Rylander, A. (2006). From value chain to value network: Insights for mobile operators. *European Management Journal*, 24(2-3):128–141.
- Perrons, R. K. (2009). The open kimono: How Intel balances trust and power to maintain platform leadership. *Research Policy*, 38(8):1300–1312.
- Petsas, T., Papadogiannakis, A., Polychronakis, M., Markatos, E. P., and Karagiannis, T. (2013). Rise of the planet of the apps: A systematic study of the mobile app ecosystem. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 277–290. ACM.
- Piller, F. T. and Walcher, D. (2006). Toolkits for idea competitions: A novel method to integrate users in new product development. *R&D Management*, 36(3):307–318.
- Pon, B., Seppälä, T., and Kenney, M. (2014). Android and the demise of operating system-based power: Firm strategy and platform control in the post-PC world. *Telecommunications Policy*, 38(11):979–991.

- Poot, T., Faems, D., and Vanhaverbeke, W. (2009). Toward a Dynamic Perspective on Open Innovation: A longitudinal Assessment of the Adoption of Internal and External Innovation Strategies in the Netherlands. *International Journal of Innovation Management*, 13(02):177–200.
- Porter, M. E. (1985). Competitive Advantage. *Management Information Systems*, 19(4):487–505.
- Porter, M. E. (2008). The five competitive forces that shape strategy. *Harvard business review*, 86(1):78–93, 137.
- Ramu, S. (2012). Mobile Malware Evolution , Detection and Defense. (April):1–15.
- Rasch, A. and Wenzel, T. (2013). Piracy in a two-sided software market.
- Raymond, E. S. (2001). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*.
- Remneland-Wikhamn, B., Ljungberg, J., Bergquist, M., and Kuschel, J. (2011). Open Innovation, Generativity and the Supplier As Peer: the Case of Iphone and Android. *International Journal of Innovation Management*, 15(01):205–230.
- Rese, A. and Baier, D. (2011). Success factors for innovation management in networks of small and medium enterprises. *R&D Management*, 41(2):138–155.
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. a. Y., Silverman, B., and Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, 52:60–67.
- Roberts, J. A., Hann, I.-H., and Slaughter, S. A. (2006). Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects. *Management Science*, 52(7):984–999.
- Rochet, J.-C. and Tirole, J. (2003). Platform Competition in Two-Sided Markets. *Journal of the European Economic Association*, 1(4):990–1029.
- Rochet, J.-C. and Tirole, J. (2006). Two-sided markets: a progress report. *RAND Journal of Economics*, 37(3):645–667.
- Ross, S. A. (1977). The Determination of Financial Structure: The Incentive-Signalling Approach. *The Bell Journal of Economics*, 8(1):23–40.
- Rossi, M. A. (2006). Decoding the Free/Open Source Software Puzzle: A Survey of Theoretical and Empirical Contributions. *The Economics of Open Source Software Development*, pages 15–55.
- Rothschild, M. and Stiglitz, J. (1976). Equilibrium in Competitive Insurance Markets: An Essay on the Economics of Imperfect Information. *The Quarterly Journal of Economics*, 90(4):629–649.
- Rysman, M. (2009). The economics of two-sided markets. *The Journal of Economic Perspectives*, pages 125–143.

- Santos, F. and Eisenhardt, K. (2009). Constructing markets and shaping boundaries: Entrepreneurial power in nascent fields. *Academy of Management Journal*, 52(4):643–671.
- Sarma, B., Li, N., Gates, C., Bhaskar, S., Ninghui, L., Chris, G., Rahul, P., and Nita-Rotaru, C. (2012). Android permissions: a perspective combining risks and benefits. In *Symposium on Access control Models and Technologies*, pages 13–22.
- Sawhney, M. S. (1998). Leveraged High-Variety Strategies: From Portfolio Thinking to Platform Thinking. *Journal of the Academy of Marketing Science*, 26(1):54–61.
- Saxton, G. D. and Anker, A. E. (2013). The aggregate effects of decentralized knowledge production: Financial bloggers and information asymmetries in the stock market. *Journal of Communication*, 63(6):1054–1069.
- Schiff, A. (2003). Open and closed systems of two-sided networks. *Information Economics and Policy*, 15(4):425–442.
- Schultz, N., Wulf, J., Zarnekow, R., and Nguyen, Q. T. (2011). The new role of developers in the mobile ecosystem: An Apple and Google case study. In *2011 15th International Conference on Intelligence in Next Generation Networks, ICIN 2011*, pages 103–108.
- Schwab, K., Marcus, A., Oyola, J., Hoffman, W., and Luzi, M. (2011). Personal data: The emergence of a new asset class. Technical report.
- Seinen, I. and Schram, A. (2006). Social status and group norms: Indirect reciprocity in a repeated helping experiment. *European Economic Review*, 50(3):581–602.
- Shah, S. (2000). Sources and Patterns of Innovation in a Consumer Products Field: Innovations in Sporting Equipment. *Management*, 18:1–27.
- Shah, S. (2003). *Community-based innovation & product development: finding from open source software and consumer sporting goods*. PhD thesis, Massachusetts Institute of Technology.
- Shah, S. and Tripsas, M. (2007). The accidental entrepreneur: The emergent and collective process of user entrepreneurship. *Strategic Entrepreneurship Journal*, 140(November):123–140.
- Shane, S. and Venkataraman, S. (2000a). The promise of entrepreneurship as a field of research. *Academy of Management Review*, 25(1):217–226.
- Shane, S. and Venkataraman, S. (2000b). the Promise of Entrepreneurship As a Field of Research. *Academy of Management Review*, 25(1):217–226.
- Sharp, B. and Dawes, J. (2001). What is Differentiation and How Does it Work? *Journal of Marketing Management*, 17(7-8):739–759.
- Sheffi, Y. (2005). *The Resilient Enterprise: Overcoming Vulnerability for Competitive Advantage*.
- Simon, H. A. (1955). A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics*, 69:99–118.

- Singer, J. D. and Willett, J. B. (2003). *Applied Longitudinal Data Analysis: Modeling Change and Event Occurrence*. Oxford University Press, New York.
- Sinkula, J. M., Baker, W. E., and Noordewier, T. (1997). A Framework for Market-Based Organizational Learning: Linking Values, Knowledge, and Behavior. *Journal of the Academy of Marketing Science*, 25(4):305–318.
- Speier, C., Valacich, J. S., and Vessey, I. (1999). The Influence of Task Interruption on Individual Decision Making: An Information Overload Perspective. *Decision Sciences*, 30(2):337–360.
- Spence, M. (1973). Job Market Signaling. *The Quarterly Journal of Economics*, 87(3):355–374.
- Spence, M. (2002). Signaling in retrospect and the informational structure of markets. *American Economic Review*, 92(3):434–459.
- Spital, F. C. (1979). An analysis of the role of users in the total R&D portfolios of scientific instrument firms. *Research Policy*, 8(3):284–296.
- Srinivasan, A. and Venkatraman, N. (2010). Indirect network effects and platform dominance in the video game industry: A network perspective. *IEEE Transactions on Engineering Management*, 57(4):661–673.
- Stiglitz, J. E. (1975). The Theory the of "Screening" Education, and the Distribution of Income. *The American Economic Review*, 65(3):283–300.
- Stiglitz, J. E. (2002). Information and the change in the paradigm in economics. *American Economic Review*, 92(3):460–501.
- Suarez, F. F. (2004). Battles for technological dominance: An integrative framework. *Research Policy*, 33(2):271–286.
- Sun, M. and Tse, E. (2009). The Resource-Based View of Competitive Advantage in Two-Sided Markets. *Journal of Management Studies*, 46(1):45–64.
- Swann, G. (2002). Bandwagon Effects in High Technology Industries. *Information Economics and Policy*, 14(3):431–433.
- Syer, M. D., Adams, B., Zou, Y., and Hassan, A. E. (2011). Exploring the development of micro-apps: A case study on the blackberry and android platforms. In *Proceedings - 11th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2011*, pages 55–64.
- Taudes, A., Feurstein, M., and Mild, A. (2000). Options Analysis of Software Platform Decisions: A Case Study. *MIS Quarterly*, 24(2):227–243.
- Teece, D. J. (2010). Business models, business strategy and innovation. *Long Range Planning*, 43(2-3):172–194.
- Tripsas, M. (2008). Customer preference discontinuities: A trigger for radical technological change. *Managerial and Decision Economics*, 29(2-3):79–97.

- Turber, S., Vom Brocke, J., Gassmann, O., and Fleisch, E. (2014). Designing business models in the era of Internet of Things: Towards a reference framework. In *Lecture Notes in Computer Science*, pages 17–31.
- Ulaga, W. and Eggert, A. (2006). Value-Based Differentiation in Business Relationships: Gaining and Sustaining Key Supplier Status. *Journal of Marketing*, 70(1):119–136.
- Utterback, J. M. (2004). The Dynamics Of Innovation. *Educuse Review*, (February):43–51.
- Utterback, J. M. and Suárez, F. F. (1993). Innovation, competition, and industry structure.
- van de Vrande, V., de Jong, J. P. J., Vanhaverbeke, W., and de Rochemont, M. (2009). Open innovation in SMEs: Trends, motives and management challenges. *Technovation*, 29(6-7):423–437.
- Vanhaverbeke, W. W. and Van De Vrande, V. (2008). Understanding the advantages of open innovation practices in corporate venturing in terms of real options. In *2008 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2008*, pages 954–958.
- Von Ehrlich, M. and Greiner, T. (2013). The role of online platforms for media markets - Two-dimensional spatial competition in a two-sided market. *International Journal of Industrial Organization*, 31(6):723–737.
- von Hippel, E. (1986). Lead Users: A Source of Novel Product Concepts. *Management Science*, 32(7):791–805.
- von Hippel, E. (1988). *The Sources of Innovation*. Oxford University Press, New York.
- von Hippel, E. (1994). "Sticky Information" and the Locus of Problem Solving: Implications for Innovation. *Management Science*, 40(4):429–439.
- von Hippel, E. (2005). *Democratizing innovation*. MIT Press.
- von Hippel, E. and Katz, R. (2002). Shifting Innovation to Users via Toolkits. *Management Science*, 48(7):821–833.
- von Hippel, E. and Tyre, M. J. (1995). How learning by doing is done: problem identification in novel process equipment. *Research Policy*, 24(1):1–12.
- von Hippel, E. and von Krogh, G. (2003). Open Source Software and the "Private-Collective" Innovation Model: Issues for Organization Science. *Organization Science*, 14(2):209–223.
- Von Krogh, G., Spaeth, S., and Lakhani, K. R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, 32(7):1217–1241.
- von Krogh, G. and von Hippel, E. (2006). The Promise of Research on Open Source Software. *Management Science*, 52(7):975–983.
- Voskoglou, C., Veritsis, A., and Constantinou, A. (2014). Developer Segmentation Q3 2014: The 8 Types of App Developers, What Motivates Them, and How To Engage Them. Technical report, VisionMobile Inc.

- Voss, C. a. (1985). Determinants of Success in the Development of Applications Software. *Journal of Product Innovation Management*, 2(2):122–129.
- Weiss, A. (1995). Human Capital vs. Signalling Explanations of Wages. *Journal of Economic Perspectives*, 9(4):133–154.
- West, J. (2003). How open is open enough? Melding proprietary and open source platform strategies. *Research Policy*, 32(7):1259–1285.
- West, J. and Gallagher, S. (2006). Challenges of open innovation: The paradox of firm investment in open-source software. *R&D Management*, 36(3):319–331.
- West, J. and Lakhani, K. R. (2008). Getting Clear About Communities in Open Innovation. *Industry & Innovation*, 15(2):223–231.
- West, J. and Mace, M. (2010). Browsing as the killer app: Explaining the rapid success of Apple's iPhone. *Telecommunications Policy*, 34(5-6):270–286.
- Wiersema, M. F. and Bowen, H. P. (2008). Corporate diversification: The impact of foreign competition, industry globalization, and product diversification. *Strategic Management Journal*, 29(2):115–132.
- Wilcox, M. and Voskoglou, C. (2015). Developer Economics Q1 2015: State of the Developer Nation. Technical report, VisonMobile Inc.
- Wittenberg-Moerman, R. (2008). The role of information asymmetry and financial reporting quality in debt trading: Evidence from the secondary loan market. *Journal of Accounting and Economics*, 46(2-3):240–260.
- Wooldridge, J. M. (2010). *Econometric Analysis of Cross Section and Panel Data*.
- Xing, L., Pan, X., Wang, R., Yuan, K., and Wang, X. (2014). Upgrading Your Android, Elevating My Malware: Privilege Escalation Through Mobile OS Updating. In *IEEE Symposium on Security and Privacy*.
- Yadong Luo, Jinyun Sun, and Lu Wang, S. (2011). Emerging Economy Copycats: Capability, Environment, and Strategy. *Academy of Management Perspectives*, 25(2):37–56.
- Yamakami, T. (2011). A Three-Dimension Analysis of Driving Factors for Mobile Application Stores: Implications of Open Mobile Business Engineering. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 885–889. IEEE.
- Yoffie, D. B. and Cusumano, M. A. (1998). *Competing on internet time: Lessons from netscape and its battle with microsoft*. Simon and Schuster.
- Yoo, B. and Choudhary, V. (2003). A Model of Neutral B2B Intermediaries. *Journal of Management Information Systems*, 19(3):43–68.
- Yoo, Y., Henfridsson, O., and Lyytinen, K. (2010). The new organizing logic of digital innovation: An agenda for information systems research. *Information Systems Research*, 21(4):724–735.

- Yoon, I. (2014). *Platform policy and its effect on diffusion: the case study of Android and iOS*. PhD thesis, Massachusetts Institute of Technology.
- Yu, J., Benatallah, B., Casati, F., and Daniel, F. (2008). Understanding mashup development. *IEEE Internet Computing*, 12(5):44–52.
- Zhou, W., Zhou, Y., Jiang, X., and Ning, P. (2012a). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy - CODASKY '12*, pages 317–326.
- Zhou, Y. and Jiang, X. (2012). Dissecting Android malware: Characterization and evolution. In *Proceedings - IEEE Symposium on Security and Privacy*, pages 95–109.
- Zhou, Y., Wang, Z., Zhou, W., and Jiang, X. (2012b). Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets. In *Proceedings of the 19th Network and Distributed System Security Symposium*, number 2.
- Zhu, F. and Iansiti, M. (2012). Entry into platform-based markets. *Strategic Management Journal*, 33(1):88–106.
- Zott, C., Amit, R., and Massa, L. (2011). The Business Model: Recent Developments and Future Research. *Journal of Management*, 37(4):1019–1042.

Appendix A

A.1 Panel Data Setting

```
.
. use data_signal, clear
(all apps by featured developers)

.
. sort developer_id_num within_app_time_order
. xtset developer_id_num within_app_time_order
    panel variable:  developer_id_num (unbalanced)
    time variable:  within_app_~r, 1 to 129
                delta:  1 unit

. xtdescribe
developer_id_num:  1, 2, ..., 365                n =          365
within_app_time_order:  1, 2, ..., 129          T =          129
                Delta(within_app_time_order) = 1 unit
                Span(within_app_time_order)  = 129 periods
                (developer_id_num*within_app_time_order uniquely identifies each observation)

Distribution of T_i:  min      5%      25%      50%      75%      95%      max
                    1        1        1        3        9       39      129

    Freq.  Percent   Cum. | Pattern
-----|-----
    94     25.75   25.75 | 1.....
> .....
    54     14.79   40.55 | 11.....
> .....
    48     13.15   53.70 | 111.....
> .....
    26      7.12   60.82 | 1111.....
> .....
    18      4.93   65.75 | 11111.....
> .....
    11      3.01   68.77 | 11111111.....
> .....
     9      2.47   71.23 | 111111111111.....
> .....
     8      2.19   73.42 | 11111.....
> .....
```

```

      8      2.19   75.62 | 11111111.....
> .....
      89      24.38 100.00 | (other patterns)
-----
      365     100.00 | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
> XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
.

```

A.2 Python Code to Determine Developer's Featuring Time

```

# The purpose is to narrow down the confirmed range of dates for
    developers to transit from non-
    featured into featured status.

# The method will be based on date to sort the snapshot of the web
    containing the developer's
    featuring status. It will then
    check whether each snapshot has
    developed to becoming featured to
    determine the narrowest date range
    of featuring status change.

def get_featured_date_range(developers):
    featured_date_range = {}
    for developer_id in developers:
        featured_date_min = 0
        featured_date_max = 0
        apps = developers[developer_id]
        for app_id in sort(apps): # sort app by release date
            app = apps[app_id]
            wbm_dates = GET_WAYBACK_MACHINE_ARCHIVES(app_id) # the archive
                                                                has been collected and processed
                                                                with other python code.

            for wbm_date in sort(wbm_dates): # sort by wbm_date first
                wbm_status = wbm_dates[wbm_date]
                if wbm_status == 'not featured':
                    if featured_date_min < wbm_date:
                        featured_date_min = wbm_date
                if wbm_status == 'featured':
                    if featured_date_max > wbm_date:
                        featured_date_max = wbm_date

    # worth to mention, featured_date_min can be 0, because
        developers are featured when they
        released their first app.

```

```

    featured_date_range[developer_id] = {
        'min': featured_date_min,
        'max': featured_date_min
    }
    return featured_date_range

# By using the date range confirmed from above method, it check
whether an app is definitely
released before its status changed
or after that. Otherwise, it needs
to be manually checked whether it
happened before or after by
searching heavily online.

def confirm_featured_type(developers, featured_date_range):
    for developer_id in developers:
        featured_date_min = featured_date_range[developer_id]['min']
        featured_date_max = featured_date_range[developer_id]['min']
        apps = developers[developer_id]
        app_featured_range_count = 0
        for app_id in sort(apps): # sort app by release date
            app = apps[app_id]
            app_featured_type = 0
            app_featured_range_count = 0
            app_release_data = app['app_release_date']
            if app_release_data <= featured_date_min:
                app_featured_type = 1 # app released before developer was
featured
            if app_release_data >= featured_date_max:
                app_featured_type = 3 # app released after developer was
featured
            if app_release_data < featured_date_max and app_release_data >
                featured_date_min:
                app_featured_range_count += 1 # so may need to manually check
.
            app['app_featured_type'] = app_featured_type
            developers[app_id]['app_featured_range_count'] =
                app_featured_range_count
    return developers

# main method.
def determine_app_featured_type(developers):
    featured_date_range = get_featured_date_range(developers)
    developers = confirm_featured_type(developers, featured_date_range)

```

```
# then manual check the app_featured_type for each app by each
# developer ...
# the rest is done in excel.
```

A.3 Statistics Output

Table A.1 Variable descriptions

Variable	Unit of analysis	Description
<i>dependent variable</i>		
status	developer-app	The transition from non-featured to featured status (0 = non-featured, 1 = featured).
performance	app	Multiplication between overall average review score by total number of reviewers treated as proxy for market performance.
<i>control variables</i>		
coverage	developer	Whether developers have apps only in game range, other range or mixed with both ranges. (0 = other-only, 1 = game-only, 2 = mixed)
price	app	The price strategy used by target app, 0 = completely free, 1 = in-app purchase 2 = paid.
sequence	developer-app	The rank position of the app within all apps by a featured developer, starting from 1.
immediate	developer	The developer was featured immediately after releasing his first app.
productivity	developer	The count of apps by a featured developer.
<i>unique</i>		
unique_dvlp	developer-app	The uniqueness of permission use from the developer's prior app.
unique_cat	category-app	The uniqueness of permission use from the prior featured app within subject.
unique_funct	functionality-app	The uniqueness of permission use from the prior app in the same functionality group.

<i>follow</i>		
days_lap_dvlp	developer-app	Number of days lapsed from the developer's prior app.
days_lap_cat	category-app	Number of days lapsed from the prior featured app within subject.
days_lap_fun	functionality-app	Number of days lapsed from the prior app in the same functionality group.

<i>diverse</i>		
scope	developer	Number of distinct categories that the developer tried before being featured.

<i>recognise</i>		
apps_featured	functionality-app	Number of apps by featured developers prior to the release of the target app within the same functionality group.
apps_nonfeatured	functionality-app	Number of apps by non-featured developers prior to the release of the target app within the same functionality group.

Table A.2 Descriptive Statistics

	mean	sd	min	max
Status	.1793855	.3837699	0	1
Performance **	331151.1	2388899	0	6.73e+07
coverage	.9801784	.7024312	0	2
price	1.073835	.7330969	0	2
sequence	14.90634	22.18869	1	126
immediate	.079782	.2710225	0	1
unique_dvlp	.2197182	.2940823	0	1
unique_cat	.3348031	.2941767	0	1
unique_funct	.3007582	.3003618	0	1
days_lapsed_dvlp	52.54115	117.8853	0	1274.059
days_lapsed_cat	16.61095	37.39198	0	493.4131
days_lapsed_funct	79.86444	118.6545	0	1330.272
scope	4.460852	2.60519	1	10
apps_featured	1.646184	2.231004	0	16
apps_nonfeatured	3.408821	3.450458	0	16
productivity	35.27106	38.98307	1	129
<i>N</i>	2018			

* All apps are released by featured developers.

** *performance* has a large standard variance than mean, so peer recognition statistic (see Section 8.3.4) adopted negative binomial regression model.

Table A.3 Correlation matrix for Signalling Model on platform

	(1)											
	status	coverage	price	sequence	immediate	unique_dvlp	unique_cat	unique_funct	days_lapsed_dvlp	days_lapsed_cat	days_lapsed_funct	scope
status	1											
coverage	-0.215***	1										
price	-0.190***	0.00477	1									
sequence	-0.198***	-0.152***	0.142***	1								
immediate	0.630***	-0.281***	-0.237***	-0.185***	1							
unique_dvlp	-0.0600**	0.180***	0.0818***	-0.0568*	-0.220***	1						
unique_cat	0.130***	0.00514	-0.0752***	-0.190***	0.0781***	0.307***	1					
unique_funct	0.101***	-0.00260	-0.109***	-0.0798***	0.113***	0.179***	0.181***	1				
days_lapsed_dvlp	0.129***	-0.0187	-0.0606**	-0.154***	-0.131***	0.158***	0.0960***	0.0461*	1			
days_lapsed_cat	0.0968***	-0.112***	-0.141***	-0.0937***	0.120***	-0.0148	0.0717**	0.0394	0.106***	1		
days_lapsed_funct	-0.0131	0.0194	-0.00607	-0.0151	0.00997	-0.0222	0.00374	0.101***	0.0328	0.0362	1	
scope	-0.417***	0.468***	0.0533*	0.275***	-0.391***	0.0981***	-0.0408	-0.0777***	-0.120***	-0.101***	0.0268	1

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

Table A.4 Signalling Model on platform selection

status	(1)		(2)		(3)	
game-only	-.610**	(-2.78)	.281	(1.26)	.226	(.96)
mixed	-.800**	(-3.22)	.468+	(1.80)	.477+	(1.79)
in-app purchase	-.164	(-.78)	.0341	(.16)	-.127	(-.56)
paid	-.367+	(-1.68)	-.416+	(-1.83)	-.400+	(-1.74)
sequence	-.0150++	(-1.47)	.000582	(.11)	.00798++	(1.46)
immediately-featured	22.57	(.01)	31.25	(.00)	32.19	(.00)
unique_dvlp			.613*	(2.29)	1.492***	(3.75)
unique_cat			.643*	(2.25)	1.016**	(3.04)
unique_funct			-.130	(-.49)	-.241	(-.85)
days_lapsed_dvlp			.00316***	(6.11)	.00552***	(5.35)
days_lapsed_cat			-.00128	(-.54)	-.00214	(-.87)
days_lapsed_funct			-.00101++	(-1.29)	-.00122++	(-1.51)
scope			-.370***	(-7.75)	-.425***	(-9.15)
unique_dvlp_2					-2.877*	(-2.51)
unique_cat_2					-1.687++	(-1.39)
days_lapsed_dvlp_2					-.00000484**	(-3.25)
scope_2					.0421**	(2.89)
Constant	-1.112***	(-5.45)	-1.315***	(-4.41)	-1.343***	(-4.13)
Constant(lnsig2u)	-1.742+	(-1.91)	-9.414	(-.43)	-13.42	(-.75)
Observations	2018		2018		2018	
N_g	365		365		365	
aic	1242.9		1103.9		1072.0	
bic	1287.7		1188.1		1178.6	
ll	-613.4		-537.0		-517.0	
df_m	6		13		17	
p	.00246		7.22e-27		7.06e-33	

t statistics in parentheses

Model 1: control variables only.

Model 2: independent variables + control variables.

Model 3: additional variables + independent variables + control variables

++ $p < .20$, + $p < .10$, * $p < .05$, ** $p < .01$, *** $p < .001$

Table A.5 Correlation matrix for Signalling Model on peer recognition

	(1)								
	performance	coverage	price	sequence	immediate	status	productivity	apps_featured	apps_nonfeatured
performance	1								
coverage	-0.0274	1							
price	-0.0622***	-0.00825	1						
sequence	-0.0533**	-0.0259	0.122***	1					
immediate	0.134***	-0.0787***	-0.175***	-0.121***	1				
status	0.0618***	0.0683***	-0.142***	0.131***	0.446***	1			
productivity	-0.0654***	-0.0340*	0.178***	0.761***	-0.159***	-0.0446**	1		
apps_featured	0.0206	0.0532**	0.0174	0.373***	0.0510**	0.214***	0.187***	1	
apps_nonfeatured	-0.0362*	-0.0381*	-0.110***	-0.000437	0.0728***	0.192***	-0.268***	-0.0420*	1

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

Table A.6 Signalling Model on peer recognition

performance	(1)		(2)		(3)	
game-only	.388***	(7.26)	.309***	(5.69)	.300***	(5.52)
mixed	.265***	(4.78)	.209***	(3.73)	.194***	(3.45)
in-app purchase	.487***	(10.53)	.515***	(11.12)	.512***	(11.02)
paid	-.303***	(-6.08)	-.311***	(-6.25)	-.306***	(-6.13)
sequence	-.00456***	(-4.08)	-.00536***	(-4.34)	-.00569***	(-4.49)
immediately-featured	.0550	(.95)	.0479	(.83)	.0425	(.73)
featured	-.0571++	(-1.39)	-.0460	(-1.09)	.104+	(1.69)
productivity	.00129+	(1.66)	-.000191	(-.23)	.0000271	(.03)
apps_featured			.0402***	(5.87)	0	(.)
apps_nonfeatured			-.0246***	(-5.08)	0	(.)
apps_featured_2					-.00322*	(-2.02)
apps_nonfeatured_2					-.00103	(-.93)
featured X apps_featured					-.0105	(-.79)
featured X apps_nonfeatured					-.0348***	(-3.56)
Constant	-1.276***	(-20.58)	-1.139***	(-16.98)	-1.249***	(-17.16)
Constant(ln_r)	-.658***	(-7.77)	-.629***	(-7.45)	-.626***	(-7.42)
Constant(ln_s)	10.79***	(61.58)	10.80***	(62.50)	10.80***	(62.53)
Observations	3409		3409		3409	
N_g	365		365		365	
aic	80183.3		80122.1		80106.2	
bic	80250.8		80201.8		80210.5	
ll	-40080.6		-40048.1		-40036.1	
df_m	8		10		14	
p	8.06e-103		1.25e-115		2.77e-117	

t statistics in parentheses

Model 1: control variables only.

Model 2: independent variables + control variables.

Model 3: additional variables + independent variables + control variables

++ $p < .20$, + $p < .10$, * $p < .05$, ** $p < .01$, *** $p < .001$

A.4 Other Tables

Table A.7 Key literatures: Platform Strategy

Platform Thinking
Gawer and Cusumano (2013); Gawer (2011); Cusumano (2010); Damiano et al. (2008); Evans et al. (2006); Eisenmann et al. (2006); Rochet and Tirole (2006); Parker and Van Alstyne (2005); Caillaud and Jullien (2003); Parker and Van Alstyne (2000); Sawhney (1998)
Platform Growth
Evans and Schmalensee (2010); Rysman (2009); Gawer and Cusumano (2008); Baldwin and Woodard (2009); Gawer and Henderson (2007); Evans and Schmalensee (2007); Suarez (2004); Gawer and Cusumano (2002)
Platform Competition
Cennamo and Santalo (2013); Chen and Miller (2012); Eisenmann et al. (2011); Srinivasan and Venkatraman (2010); Gawer and Henderson (2007); Armstrong (2006); Rochet and Tirole (2003); Yoffie and Cusumano (1998)
Platform Open
Boudreau (2010); West and Mace (2010); Eisenmann et al. (2008); Boudreau (2008); Economides and Katsamakas (2006); von Hippel (2005); Schiff (2003); West (2003); West and Lakhani (2008)
Attraction for Complements
Price: Lin et al. (2011); Hagiu (2009b); Hagiu (2006)
Tying: Amelio and Jullien (2012); Choi (2010); Lee and Wu (2009)
Boundary Resource: Ghazawneh and Henfridsson (2012)
Differentiation of Complements
Barriers to Entry: Pehrsson (2009); Farrell and Klemperer (2007)
Exclusive Contracting: Hagiu and Lee (2011); Mantena et al. (2010); Hagiu (2009a); Armstrong and Wright (2007)
Trust: Perrons (2009); Evans (2003); Rasch and Wenzel (2013)
Regulation: Bauer (2014); Genakos and Valletti (2012)

Table A.8 Key literatures: App Economy

Platform Focuses
Brockmann et al. (2015); Cecere et al. (2015); Campbell-Kelly et al. (2014); Karhu et al. (2014); Hyrynsalmi et al. (2014); Pon et al. (2014); Kouris and Kleer (2012); Basole and Karla (2011); Remneland-Wikhamn et al. (2011)
Developer Focuses
Lim et al. (2015); Koch and Kerschbaum (2014); Xing et al. (2014); Li et al. (2014); Liu et al. (2014); Fu et al. (2013); Hyrynsalmi et al. (2012a); Möller et al. (2012); Boudreau (2012); Zhou et al. (2012a); Ickin et al. (2012); Lim and Bentley (2012b); Schultz et al. (2011); Lee and Raghu (2011); Charland et al. (2011); Chin et al. (2011); Holzer and Ondrus (2011)
App Permission
Jung et al. (2013); Felt et al. (2012); Barrera et al. (2012); Zhou and Jiang (2012); Chia et al. (2012); Sarma et al. (2012); Felt et al. (2011b); Felt et al. (2011a); Felt et al. (2011c)
Other
Ghose and Han (2014); Maurer and Tiwana (2012); Basole and Karla (2012); Jansen and Bloemendal (2013); Petsas et al. (2013); Yamakami (2011); Han et al. (2012a); Ham and Park (2011)

Table A.9 Key literatures: Signaling Theory

Review
Connelly et al. (2011); Bliege Bird and Smith (2005); Stiglitz (2002); Spence (2002); Kirmani and Rao (2000)
Application: IPO
Deb (2013); Cohen and Dean (2005); Filatotchev and Bishop (2002); Certo et al. (2001)
Application: Human Capital
Bedard (2001); Weiss (1995); Spence (1973)
Application: Others
Ahlers et al. (2015); Keupp and Gassmann (2009); Wittenberg-Moerman (2008); Healy and Palepu (2001); Ross (1977); Akerlof (1970); Rothschild and Stiglitz (1976); Akerlof (1970)

Table A.10 Key literatures: Strategic Behaviour

Differentiation
Anderson et al. (2010); Parrish et al. (2006); Pehrsson (2004); Santos and Eisenhardt (2009); Katila et al. (2008)
Diversification
Wiersema and Bowen (2008); Goranova et al. (2007); Lavie and Rosenkopf (2006); Ulaga and Eggert (2006); Matsusaka (2001)
Learning
Yadong Luo et al. (2011); Enkel and Gassmann (2010); Katila and Chen (2008); Baldwin and Clark (2006); Markides and Geroski (2004); Hertel et al. (2003); Katila and Ahuja (2002); Calantone et al. (2002); Hurley and Hult (1998); Sinkula et al. (1997); March (1991)
Business Model
Hyrnsalmi et al. (2012b); Zott et al. (2011); Rese and Baier (2011); Teece (2010); Clemons (2009); Boudreau and Lakhani (2009); Peppard and Rylander (2006); Morris et al. (2005); Jiang et al. (2004); Lakhani and Von Hippel (2003); Normann and Ramírez (1993); Linberg (1999)
Competitive Advantage
Zhu and Iansiti (2012); Evans (2009); Porter (2008); Sheffi (2005); MacCormack and Verganti (2003); Caves (2003); Barney (1995)
Reuse
Mojica et al. (2014); McDonnell et al. (2013); Mojica et al. (2012); Frakes and Kang (2005)
Unique Resource
Kalapesi (2013); Schwab et al. (2011); Belleflamme and Peitz (2010); Seinen and Schram (2006); Harhoff et al. (2003)
Open Source Software
Dahlander and Magnusson (2008); von Krogh and von Hippel (2006); Lerner and Tirole (2005); Roberts et al. (2006); MacCormack et al. (2006); Kuk (2006); Von Krogh et al. (2003); von Hippel and von Krogh (2003); Raymond (2001); von Hippel (1994)
Open Innovation
Aitamurto and Lewis (2013); Huizingh (2011); Dahlander and Gann (2010); Harison and Koski (2010); Enkel et al. (2009); van de Vrande et al. (2009); Chesbrough (2007b); Chesbrough and Crowther (2006); Gassmann (2006); Chesbrough (2003a); Chesbrough (2003b)
Wisdom of Crowds
Boudreau and Lakhani (2013); Bogers et al. (2010); Shah and Tripsas (2007); Jeppesen and Frederiksen (2006); Baldwin et al. (2006); Franke and Shah (2003); von Hippel and Katz (2002); Lerner and Tirole (2002); Shane and Venkataraman (2000a); von Hippel (1986)

Table A.11 The history of Android and Google Play Store

2003	Android Inc. founded by three developers: Andy Rubin, Rich Miner, Nick Sears and Chris White
2004	Apple started confidential “Project Purple” with a team of 1000 employees, later become iPhone and iOS
07-2005	Google acquired Android Inc. and kept major developers (Rubin, Miner, Sears) to continue develop Android in Google
09-01-2007	Apple unveiled iPhone and iOS project to public at Macworld Confernce
29-06-2007	Apple released first iPhone device on sale
17-10-2007	Apple announced that iOS SDK was under development
05-11-2007	“Open Handset Alliance” (OHA) was found which is led by Google with 34 members companies, and included mobile handset makers, application developers, mobile carriers and chip makers. OHA unveiled Android based on open source license to compete against other mobile platforms.
12-11-2007	released first Android SDK m3-rc20a to public and its open strategy, no hardware device available to public at this time
16-11-2007	released Android SDK m3-rc22a
29-11-2007	highlighted a few projects that fork on Android SDK
14-12-2007	released Android SDK m3-rc37a
03-01-2008	anonounced first “Android Developer Challenge” contest for open innovation
18-01-2008	anounnced Android Discussion Groups to encourage online discussion and help
13-02-2008	released Android SDK m5-rc14
03-03-2008	released Android SDK m5-rc15
04-03-2008	announced “Apps for Android” to tutorial on app development
06-03-2008	Apple released first iOS SDK to developers
17-04-2008	closed first “Android Developer Challenge”, and developers from 70 countries submitted 1788 entries to the contest
10-07-2008	Apple released iOS App Store
11-07-2008	Apple released second generation iPhone device (iPhone 3G) in twenty-two countries
18-08-2008	released Android SDK 0.9, and also first Android physical device (G1 or HTC Dream)
28-08-2008	unveiled Android Market to developers, presented the winners of the first “Android Developer Challenge” judged by specialists

* Appendix A.6 has a detailed explanation of some technical terms used here.

Table A.11 The history of Android and Google Play Store - continued

23-09-2008	• released Android SDK 1.0-r1
21-10-2008	• announced Android System open sourced, and can be free to download and develop
22-10-2008	• released Android Market to users, and the first Android device (G1) is available to purchase on the day
07-11-2008	• started a series online talk about Android apps development, the first two developers talked how they build music-related apps
05-12-2008	• added resource for developers and announced program for developers to purchase devices that enable developers to test and debug applications more easily
09-02-2009	• released Android SDK 1.1
13-02-2009	• enabled paid app strategy in Android Market
09-03-2009	• encouraged software update for “Android Developer Phone” as first Android system was unstable
27-04-2009	• released Android SDK 1.5
27-05-2009	• announced second “Android Developer Challenge” contest by submitting entries to Android Market
15-09-2009	• released Android SDK 1.6 by support CDMA and more screen size devices
24-09-2009	• started first round judging of “Android Developer Challenge” contest by users on Android market for two weeks
06-10-2009	• closed first round judging of “Android Developer Challenge” contest by users on Android market, over 26000 users reviewed submitted apps
27-10-2009	• released Android 2.0 with sync features
30-11-2009	• announced the winners of second “Android Developer Challenge”, showcased top 200 submission in the gallery
11-12-2009	• announced “Virtual Office Hours” by allowing developers talking to Google staff in IRC during those sessions each week
17-12-2009	• added Resources tabs in developer.android.com
18-12-2009	• integrated Stack Overflow to support Android app development by fitting to developers’ habit
11-01-2010	• released Android SDK 2.1 and announced first device made by Google in its “Nexus series”

* Appendix A.6 has a detailed explanation of some technical terms used here.

Table A.11 The history of Android and Google Play Store - continued

22-01-2010	announced “Android Developer Labs World Tour” to provide hand-on tutorial by coming to developers’ cities
20-05-2010	announced Android SDK 2.2 in Google I/O by adding enterprise capabilities and free Android Market Error Report service
23-06-2010	announced the service by Google to detect and remove malicious apps in Android Market
27-07-2010	added tutorial on how to better protect apps against unauthorized use, announced free licensing service by Google
04-08-2010	added tutorial on how to handle user data in Android apps
30-09-2010	expanded Android Market into 20 more countries to coverage more users and developers
21-10-2010	discussed the importance of app quality to the Android ecosystem
24-12-2010	enabled content rating for apps by users in Android Market
06-12-2010	released Android SDK 2.3 by improving on mutlimedia, announced second “Nexus series” phone device by Google
22-02-2011	released Android 3.0 targeting on tablet
29-03-2011	enabled in-app billing in Android Market which gives developers more ways to monetize apps with try-and-buy, virtual goods, upgrades and other billing models
10-05-2011	released Android SDK 3.1 in Google I/O conference
11-05-2011	announced featuring mechanism for apps and developers, to easy users search from a ocean of apps
12-07-2011	released redesign Android Market, to better showcase featured apps
18-10-2011	released Android SDK 4.0 with multitasking and other features
15-12-2011	announced “Android Training” which is a collection of free classes on how to build better Android apps
16-12-2011	released Android SDK 4.0.3 with new API
02-02-2012	discussed the phenomena and solution of security issue in Android app development
06-03-2012	released Google Play Store for Google center control of market place. It integrated Android Market, Google Music, Google eBookstore in a central place
24-05-2012	enabled in-app subscription in Google Play for media contents

* Appendix A.6 has a detailed explanation of some technical terms used here.

Table A.11 The history of Android and Google Play Store - continued

21-06-2012	enabled featured developers to reply to user review in Google Play. Featured developers can discuss the new features and potential improvement of the app with users directly and make collective effort
27-06-2012	released Android SDK 4.1 at Google I/O conference, by enabling automatic app updating on devices
14-09-2012	discussed on the benefits & importance of compatibility and announced free “Compatiability Test Suite” by Google to developers
26-09-2012	announced Google Play Services API 1.0, including Google+ APIs and OAuth 2.0
13-11-2012	released Android 4.2 to global users
06-12-2012	released Google Play Services API 2.0, adding Maps and Photo Sphere APIs
06-03-2013	celebrated first birthday of Google Play App to give free apps as gift
09-04-2013	released redesigned Google Play Store for easier search
13-05-2013	enable all developers to reply to user review in Google Play
15-05-2013	announced “Android Studio” which is a first dedicated IDE (Integrated Development Environment) for Android development. Previously Android development was based on third-party IDE. In additional, announced Google Play game services enable cross platform game development with iOS
16-05-2013	released Google Play Services API 3.1 by adding Google play Games services API, Location API, Google Cloud Messaging API, Cross-Platform Single Sign On API
23-07-2013	released Android SDK 4.3 to support tablet devices
20-08-2013	released Google Play Services API 3.2 by adding InstantBuy API
31-10-2013	released Android SDK 4.4 and Project Svelte by reducing memory to run Android on broader range of devices, also released Google Play Services API 4.0 by adding Google Mobile Ads API
09-01-2014	released Google Play Services API 4.1 by adding Google Drive API
03-02-2014	released Google Play Services API 4.2 by adding Google Cast API and Google Client API
17-03-2014	released Google Play Services API 4.3 by adding Google Analytics API, Google Tag Manager API and Address API
07-05-2014	released Google Play Services API 4.4 by adding Activity Recognition API and Wallet API

* Appendix A.6 has a detailed explanation of some technical terms used here.

Table A.11 The history of Android and Google Play Store - continued

06-06-2014	hosted Google I/O developers conference to encourage gloabl app business
02-07-2014	released Google Play Services API 5.0 by adding Wearable Service API, Dynamic Security Provider API, and App Indexing API
16-09-2014	released Google Play Services API 6.1 by adding Google Fit API
17-10-2014	released Android SDK 5.0
17-11-2014	released Google Play Services API 6.5 by adding Granular Dependency Managment mechanism
08-12-2014	released Android Studio IDE 1.0
02-03-2015	released Google Play Services API 7.0
10-03-2015	released Android SKD 5.1
...	...
...	...
...	...

official blog sources:

<http://android-developers.blogspot.co.uk/> <http://officialandroid.blogspot.co.uk/> <http://googleblog.blogspot.co.uk/> <http://googleresearch.blogspot.co.uk/> <http://googledevelopers.blogspot.co.uk/> <http://googlemobile.blogspot.co.uk/>

Table A.12 All non-game categories in Google Play Store

<i>Book & Reference</i>	<i>Business</i>	<i>Music & Audio</i>
<i>Communication</i>	<i>Education</i>	<i>New & Magazines</i>
<i>Entertainment</i>	<i>Finance</i>	<i>Personalisation</i>
<i>Heath & Fitness</i>	<i>Lifestyle</i>	<i>Photography</i>
<i>Libraries & Demo</i>	<i>Medical</i>	<i>Productivity</i>
<i>Live Wallpaper</i>	<i>Comics</i>	<i>Shopping</i>
<i>Media & Video</i>	<i>Social</i>	<i>Sports</i>
<i>Transport</i>	<i>Tools</i>	

Table A.13 All game categories in Google Play Store

<i>Adventure</i>	<i>Action</i>	<i>Arcade</i>
<i>Board</i>	<i>Card</i>	<i>Casino</i>
<i>Casual</i>	<i>Family</i>	<i>Educational</i>
<i>Music</i>	<i>Puzzle</i>	<i>Racing</i>
<i>Role Playing</i>	<i>Sports</i>	<i>Simulation</i>
<i>Strategy</i>	<i>Trivia</i>	<i>World</i>

Table A.14 Official app download information

<i>Index</i>	<i>app_install</i>
<i>0</i>	<i>0 - 10</i>
<i>1</i>	<i>10 - 50</i>
<i>2</i>	<i>50 - 100</i>
<i>3</i>	<i>100 - 500</i>
<i>4</i>	<i>500 - 1,000</i>
<i>5</i>	<i>1,000 - 5,000</i>
<i>6</i>	<i>5,000 - 10,000</i>
<i>7</i>	<i>10,000 - 50,000</i>
<i>8</i>	<i>50,000 - 100,000</i>
<i>9</i>	<i>100,000 - 500,000</i>
<i>10</i>	<i>500,000 - 1,000,000</i>
<i>11</i>	<i>1,000,000 - 5,000,000</i>
<i>12</i>	<i>5,000,000 - 10,000,000</i>
<i>13</i>	<i>10,000,000 - 50,000,000</i>
<i>14</i>	<i>50,000,000 - 100,000,000</i>
<i>15</i>	<i>100,000,000 - 500,000,000</i>
<i>16</i>	<i>500,000,000 - 1,000,000,000</i>

Table A.15 List of permission groups defined by Android platform

	<i>perm_group</i>	permission
<i>system</i>	<i>Bluetooth connection info</i>	An app can control bluetooth on your devices, which includes broadcasting to or getting information about nearby Bluetooth devices.
	<i>Cellular data settings</i>	An app can control your mobile data connection and potentially the data you receive.
	<i>Device & app history</i>	e.g. Read sensitive log data, Retrieve system internal state, Read web bookmarks and history, Retrieve running apps.
	<i>Device ID & call info</i>	An app can access your device ID(s), phone number, whether you are on the phone and the number connected by call.
	<i>Wearable sensors/activity</i>	Allows the app to access data from wearable sensors, such as heart rate monitors. Can receive periodic updates on physical activity levels.
	<i>WIFI connection info</i>	An app can access your device's WIFI information, like if WIFI is turned on and the name(s) of connected devices.
<i>person</i>	<i>Identify</i>	An app can use your account and/or profile information on your devices. e.g. Find accounts on the device, Reading your own contact card, Modify your own contact card, Add or remove accounts.
	<i>Location</i>	e.g. Approximate location, Precise location, Access extra location provider comments, GPS access.
	<i>Camera</i>	e.g. Take pictures and videos, Record video.
	<i>Microphone</i>	An app can use your device's microphone.
	<i>Photo/Media/Files</i>	An app can use files or data stored on your device.
	<i>Others</i>	e.g. Read your social stream (on some social networks), Write to your social stream (on same social networks), Access subscribed feeds.
<i>social</i>	<i>Calendar</i>	e.g. Read calendar events plus confidential information, Add or modify calendar events and send email to guests without owner's knowledge.
	<i>Contacts</i>	An app can use your device's contacts, it may include the ability to read and modify your contacts.
	<i>Phone</i>	An app can use your phone and/or its call history.
	<i>SMS</i>	An app can use your device's text messaging (SMS) or (MMS). This group may include the ability to use text, picture, or video messages.

* Google has defined more than 100 permissions for all kinds of app development. Each permission can fit into permission group defined by Google later. I then group permission groups into three user data types.

Table A.16 Distribution of category counting of featured developers

cate count	developer type				total developers counts		
	no game-only		game-only				
	N	(%)	N	(%)	N	(%)	Cum.
1	125	(78.62%)	34	(21.38%)	159	(43.56%)	43.56
2	37	(57.81%)	27	(42.19%)	64	(17.53%)	61.10
3	22	(53.66%)	19	(46.34%)	41	(11.23%)	72.33
4	14	(42.42%)	19	(57.58%)	33	(9.04%)	81.37
5	5	(33.33%)	10	(66.67%)	15	(4.11%)	85.48
6	6	(50.00%)	6	(50.00%)	12	(3.29%)	88.77
7	3	(23.08%)	10	(76.92%)	13	(3.56%)	92.33
8	3	(42.86%)	4	(57.14%)	7	(1.92%)	94.25
9	4	(57.14%)	3	(42.86%)	7	(1.92%)	96.16
10	1	(50.00%)	1	(50.00%)	2	(0.55%)	96.71
11	4	(80.00%)	1	(20.00%)	5	(1.37%)	98.08
12	3	(100.00%)	0	(0.00%)	3	(0.82%)	98.73
13	3	(100.00%)	0	(0.00%)	3	(0.82%)	99.73
15	1	(100.00%)	0	(0.00%)	1	(0.27%)	100.0

descriptive summary of category counting of featured developer:

Obs = 365, Mean = 2.9, Stand Error = 0.1409, 95% Confidence Interval = [2.63, 3.19]

* The distribution is heavily power law shape. Game developers would be different from non-featured developer on the unique category.

Table A.17 Study on featured developers' business model on user data

Developer ID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
NFL	US	1920	3		1			1	1	1		1	1		1	
NBC	US	1926	3		1			1	1	1		1	1		1	
PGA	US	1929	2		1	1		1	1	1		1	1		1	
Walmart	US	1962	2		1			1	1	1	1		1		1	
HRS	DE	1972	2					1	1			1	1		1	
Home+Box+Office	US	1972	3													
Guardian	UK	1982	3	1		1	1	1	1	1		1	1		1	
The+Weather	US	1982	2	1				1	1	1	1		1		1	
Earth+Networks	US	1983	3	1				1	1	1	1	1	1	1	1	1
Showtime	US	1983	2					1	1	1			1		1	
A+E TV	US	1984	4		1	1		1	1	1		1	1		1	
MeteoGroup	DE	1986	4			1	1	1	1	1		1	1		1	1
Eurosport	FR	1989	3		1	1		1	1	1			1		1	
Team+17	UK	1990	3			1				1		1	1		1	
Ancestry.com	US	1990	2			1		1	1				1	1		1
D3Publisher	JP	1992	2			1							1			
DIRECTV	US	1994	3		1			1	1	1		1	1		1	
Netflix	US	1997	2		1			1	1	1	1		1		1	
Immobilien+Scout	DE	1998	4		1			1	1	1	1	1	1		1	
PayPal	US	1998	3					1		1	1	1	1		1	
Scripps+Networks	US	1999	2		1			1	1	1			1		1	
NBA+Digital	US	1999	3		1			1	1	1		1	1		1	
ProSiebenSat 1	DE	2000	3			1		1		1		1	1		1	
Fandango	US	2000	3		1		1	1		1			1		1	
GAMEVIL	US	2000	4			1				1		1	1		1	
Tomorrow+Focus	DE	2001	2	1				1	1	1	1		1		1	
Large+Animal	US	2001	3	1		1				1		1	1		1	
Rhapsody	US	2001	3					1	1	1		1	1		1	
Nekki.	RU	2002	3		1							1	1		1	
Shazam	UK	2002	2	1	1		1	1	1	1			1		1	
LinkedIn	US	2002	2	1	1			1	1	1	1		1	1	1	1
StumbleUpon	US	2002	2		1			1	1	1	1		1		1	
inXile	US	2002	4			1				1		1	1		1	
Rovio+Stars	FI	2003	3			1			1	1		1	1		1	
Skype	LU	2003	3					1		1	1	1	1		1	
TOTAL COUNT:100	19	28	264	21	33	47	18	58	53	84	31	50	90	18	80	22

* The definition of each column is defined in the end of table.

Table A.17 Study on featured developers' business model on user data - continue

Developer ID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
LogMeIn,+Inc.	US	2003	2				1	1	1	1			1			
HyperDevbox	JP	2005	4				1			1		1	1		1	
Trulia	US	2005	2		1			1	1	1	1		1		1	1
Automattic	US	2005	2	1				1	1		1		1	1		1
Box	US	2005	2				1	1		1	1			1	1	1
Deezer	FR	2006	2			1		1	1	1	1	1	1		1	
Apalon+Apps	IE	2006	4			1				1		1	1		1	
TripIt,+Inc.	US	2006	2	1	1			1	1	1	1		1		1	1
Twitter,+Inc.	US	2006	2	1	1			1		1	1		1	1	1	1
textPlus	CA	2007	2			1			1	1			1	1		1
KITERETSU	JP	2007	3													
JetRadar	TH	2007	2					1	1	1	1		1		1	
Taptu	UK	2007	2					1	1	1			1		1	
comiXology	US	2007	2		1	1		1	1	1	1	1				
Dropbox,+Inc.	US	2007	3			1	1	1		1	1	1	1	1		1
Bandsintown	US	2007	2	1	1	1		1			1		1	1	1	1
LivingSocial	US	2007	2		1			1	1	1			1		1	
Mobiata	US	2007	2						1	1			1		1	
MobilesRepublic	FR	2008	4		1	1		1		1		1	1		1	
socialpoint	SP	2008	3			1		1		1		1	1		1	
Spotify+Ltd.	UK	2008	2	1	1	1		1	1	1	1		1		1	
SwiftKey	UK	2008	3		1	1			1	1		1	1		1	
Hulu	US	2008	2	1	1			1	1	1			1		1	
Everte	US	2008	3			1	1	1		1	1	1	1	1		1
Groupon	US	2008	2		1			1		1	1		1		1	
DJiT	FR	2009	4			1				1		1	1		1	
Wooga	DE	2009	2							1	1		1		1	
Goodgame	DE	2009	2			1						1	1		1	
Escapist	UK	2009	2			1	1		1	1	1	1	1		1	
SoundHound	US	2009	3		1				1	1			1		1	
Foursquare	US	2009	3		1			1		1	1		1	1	1	1
cloud.tv	US	2009	4			1		1		1		1	1	1	1	1
Barnesandble	US	2009	2	1		1		1	1						1	
Budge+Studios	CA	2010	3			1				1		1	1			
Sarah+rthway	CA	2010	3													
TOTAL COUNT:100	19	28	264	21	33	47	18	58	53	84	31	50	90	18	80	22

* The definition of each column is defined in the end of table.

Table A.17 Study on featured developers' business model on user data - continue

Developer ID	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Supercell	FI	2010	3			1				1			1			
Tuokio+Oy	FI	2010	2	1					1	1		1	1		1	
LevelUp+Studio	FR	2010	3			1	1			1		1	1		1	
The+Game+Bakers	FR	2010	2			1			1	1			1		1	
Korrisoft	FR	2010	4			1				1		1	1		1	
6+Wunderkinder	DE	2010	2		1	1	1				1	1	1	1	1	1
Yooii+Studios	KR	2010	3			1				1		1	1		1	
BiCORE	KR	2010	3			1										
ticketmonster	KR	2010	2					1	1	1	1		1		1	
Max+MP	RU	2010	3													
Ironhide+Game	UY	2010	3													
Rdio	US	2010	2	1	1		1	1	1	1	1	1	1		1	
AllTrails	US	2010	2			1	1	1	1	1		1	1	1	1	1
HealthTap	US	2010	2					1		1	1		1	1	1	1
Skimble+Inc.	US	2010	3	1		1	1		1	1		1	1	1	1	1
Songza	US	2010	2					1	1	1			1		1	
Yodo1+Games	CN	2011	2							1			1			
IUMSOCIUS	KR	2011	2			1		1		1			1		1	
Dirtybit	NO	2011	4	1		1	1			1		1	1			
Rayark+Inc.	TW	2011	3			1				1		1	1		1	
Beamly	UK	2011	3	1	1			1	1	1	1		1	1	1	1
Hidden+Variable	US	2011	3			1	1			1		1	1		1	
Duolingo	US	2011	2			1		1	1	1		1	1		1	
Nexercise	US	2011	3				1		1	1		1	1		1	
theScore,+Inc.	CA	2012	2	1					1	1			1		1	
Tomas+Hubalek	CZ	2012	3													
Medisafe	IL	2012	2		1					1		1	1	1	1	1
Supersolid	UK	2012	3			1				1		1	1			
Team+Chaos	US	2012	2			1		1	1	1			1		1	1
TiKL+Inc	US	2012	3	1		1				1			1		1	
TOTAL COUNT:100	19	28	264	21	33	47	18	58	53	84	31	50	90	18	80	22

A = country domain code, B = company founded year, C = count of apps by the developer, D = does advertisement banner?, E = has content as AD?, F = does in-app billing?, G = does freemium, H = is complementary to other core business, I = has competitive resource?, J = has functionality expansion?, K = provides API?, L = app can offline to use?, M = has integrated social service?, N = has user data as a service in the app? O = Has user data integrated?, P = has user contributed data?

Those 100 developers are randomly selected from 365 featured developers.

Table A.18 Developers who adopt diversification but demand the same user data

Developer ID	Price	Primary Category	A	B	C	D	E
Ancestry.com	i(1) f(2)	Photography		2	1		
BarnesandNoble.com	f(2)	Books & Reference		2	1		1
Box	f(2)	Business	1	2	1		
Clear+Channel	f(2)	Music & Audio	1	2	1		1
comiXology	i(2) f(2)	Comics	1	2	1		
D3Publisher	i(2) f(2)	Puzzle		2	1		1
Deezer+Mobile	i(1) f(2)	Music & Audio	1	2	1		1
Duolingo	i(1) f(2)	Education	1	2	1		
Escapist+Games	i(1) p(1) f(1)	Education	1	2	1		
Goodgame+Studios	i(2) f(2)	Strategy	1	2	1		1
Groupon,+Inc.	f(2)	Shopping		2	1		1
HRS+Hotel	f(2)	Travel & Local	1	2	1		
Hulu	f(2)	Entertainment	1	2			1
JetRadar	f(2)	Travel & Local	1	2	1		
LivingSocial,+Inc.	f(2)	Shopping		2	1		
Medisafe+Project	f(2)	Medical		2	1	1	1
Netflix,+Inc.	f(2)	Entertainment	1	2	1		1
Postano	f(2)	Health & Fitness		2	1		1
RabugenTom	p(1) f(1)	Books & Reference	1	2			
Rdio	f(2)	Music & Audio	1	2	1		1
Scripps+Networks	f(2)	Lifestyle	1	2	1		1
Shazam	p(1) f(1)	Music & Audio	1	2	1		1
Songza	f(2)	Music & Audio	1	2	1		1
Spotify+Ltd.	i(2) f(2)	Music & Audio	1	2	1		1
Taptu	i(1) f(2)	News & Magazines	1	2	1		1
Team+Chaos	i(2) f(2)	Puzzle		2	1		1
textPlus	i(2) p(1) f(1)	Social	1	2	1	1	1
The+Game+Bakers	i(2) p(2)	Action		2	1		1
The+Weather	f(2)	Weather		2	1		1
theScore,+Inc.	f(2)	Sports	1	2	1	1	1
COUNT TOTAL: 94		30	40	509	90	9	78

* The definition of each column is defined in the end of table.

Table A.18 Developers who adopt diversification but demand the same user data - continue

Developer ID	Price	Primary Category	A	B	C	D	E
Tomorrow+Focus	f(2)	News & Magazines	1	2	1		
TripIt,+Inc.	p(1) f(1)	Travel & Local	1	2	1	1	1
Trulia	f(2)	Lifestyle	1	2	1		1
Tuokio+Oy	i(2) f(2)	Action		2	1		1
Twitter,+Inc.	f(2)	Social	1	2	1	1	1
Walmart	f(2)	Shopping		2	1		
Yodo1+Games	i(2) f(2)	Action		2	1		1
BiCORE	i(3) f(3)	Puzzle		3	1	1	1
Budge+Studios	i(3) f(3)	Family		3	1		1
DIRECTV,+LLC	f(3)	Entertainment	1	3	1		1
Dropbox,+Inc.	i(2) f(3)	Productivity		3	1	1	1
Evernote	i(1) f(3)	Lifestyle		3	1	1	1
Hidden+Variable	i(3) p(1) f(2)	Puzzle		3	1		1
Immobilien+Scout	f(3)	Lifestyle		3	1		
Ironhide+Game	i(2) p(2) f(1)	Arcade		3	1		1
Large+Animal	i(2) p(1) f(2)	Puzzle		3	1		1
Marvel+Comics	i(1) f(3)	Comics		3	1		1
Marvel+Games	i(2) p(2) f(1)	Action	1	3	1		1
Nekki.	i(3) f(3)	Casino		3	1		1
Rayark+Inc.	i(3) f(3)	Music		3	1		1
Rhapsody	f(3)	Music & Audio	1	3	1		1
Rovio+Stars	i(3) f(3)	Puzzle	1	3	1		1
Sarah+Northway	p(2) f(1)	Puzzle		3			
Skimble+Inc.	i(1) p(1) f(2)	Health & Fitness	1	3	1		1
socialpoint	i(3) f(3)	Simulation	1	3	1		1
SoundHound	p(1) f(2)	Music & Audio	1	3	1		1
Supersolid	i(3) f(3)	Adventure		3	1		1
SwiftKey	i(1) p(2) f(1)	Productivity	1	3	1	1	1
Apalon+Apps	i(2) p(2) f(2)	Weather		4	1		1
DJiT	i(4) p(2) f(2)	Music & Audio	1	4	1		1
COUNT TOTAL: 94		30	40	509	90	9	78

* The definition of each column is defined in the end of table.

Table A.18 Developers who adopt diversification but demand the same user data - continue

Developer ID	Price	Primary Category	A	B	C	D	E
HyperDevbox	i(1) p(3) f(1)	Arcade	1	4	1		1
Onefootball	f(4)	Sports		4	1		1
PIXOWL+INC.	i(4) f(4)	Arcade		4	1		1
Smule,+Inc	i(4) f(4)	Music & Audio	1	4	1		1
Vector+Unit	i(2) p(3) f(1)	Racing	1	4	1		
ESPN+Inc	f(5)	Sports	1	5	1		1
flaregames	i(5) f(5)	Action		5	1		1
JFDP+Labs	p(3) f(2)	Photography		5	1		
Limbic	i(5) p(3) f(2)	Strategy		5	1		1
MOB+IN+LIFE	i(5) f(5)	Strategy		5	1		1
SkyVu	i(5) p(1) f(4)	Arcade		5	1		1
TripAdvisor	f(5)	Travel & Local	1	5	1		
Halfbrick+Studios	i(6) p(2) f(4)	Action		6	1		1
TheAppsGames	i(5) p(1) f(5)	Role Playing		6	1		1
DotEmu	p(7)	Adventure		7	1		1
Spacetime	i(7) f(7)	Role Playing		7	1		1
Toca+Boca+AB	p(8)	Education	1	8	1		1
Backflip+Studios	i(5) f(9)	Arcade		9	1		1
Frima+Studio	i(4) p(4) f(5)	Educational		9	1		1
Kiwi,+Inc.	i(9) f(9)	Simulation		9	1		1
Minoraxis	i(9) p(3) f(6)	Arcade		9	1		1
Storm8+Games	i(10) f(10)	Arcade		10			1
tap4fun	i(10) f(10)	Strategy		10	1		1
Rockstar+Games	i(4) p(9) f(2)	Action		11	1		1
Deemedya+m.s	i(8) p(3) f(9)	Racing		12	1		1
ZeptoLab	i(12) p(6) f(6)	Puzzle	1	12	1		1
Hothead+Games	i(14) f(14)	Sports		14	1		1
PlayScape	i(6) f(14)	Arcade		14	1		1
Smartpix+Games	i(3) p(9) f(7)	Puzzle		16	1		1
Hexage+Ltd	i(9) p(8) f(9)	Puzzle		17	1		1
COUNT TOTAL: 94		30	40	509	90	9	78

* The definition of each column is defined in the end of table.

Table A.18 Developers who adopt diversification but demand the same user data - continue

Developer ID	Price	Primary Category	A	B	C	D	E
StoryToys	i(17) p(23) f(5)	Books & Reference		28	1		1
Nevosoft+Inc	i(18) p(13) f(18)	Casual		31	1		1
Miniclip.com	i(32) p(2) f(30)	Arcade		32	1		1
TeamLava	i(32) f(32)	Casual		32	1		1
COUNT TOTAL: 94		30	40	509	90	9	78

A = is the developer's apps all in same category?, B = count of the developer's app, C = use personal user data, D = use social user data, E = use system user data

Price: the letter present the price type for the developer's apps (i = in-app billing, f = free, p = paid), the number in bracket is the count of his apps in this price type. Remember, an app can use multiple price strategy.

Primary Category: it is the most used category by the developer's app.

There are 94 developers in the list. All developers in the list have more than 2 apps and every developer's apps use same user data. This list is to study why do developers diverse but using the same user data.

Table A.19 Developers who only have one app

Developer ID	Price	Primary Category	A	B	C	D	E	F
intellijoy	f(1)	Educational	1	1	o			
Pixelberry+Studios	i(1) f(1)	Simulation	1	1	o	1		1
HUUUGE+GAMES	i(1) f(1)	Strategy	1	1	o	1		1
Hit+Entertainment	p(1)	Casual	1	1	o			
PlayDate+Digital+Inc.	p(1)	Education	1	1	o			
CJ+Netmarble	f(1)	Strategy	1	1	p	1		1
Stick+Sports+Ltd	i(1) f(1)	Sports	1	1	p	1		1
maxelus.net	f(1)	Entertainment	1		t			
Bauer+Vertriebs	i(1) f(1)	Entertainment	1		t	1		1
shiftyjelly	p(1)	Weather	1		t	1		1
Cabin	f(1)	Lifestyle	1		u	1	1	
500px	f(1)	Photography	1		u	1		
Banjo+Inc	f(1)	Social	1		u	1	1	1
truecaller	i(1) f(1)	Communication	1		u	1	1	1
Avocado+Software	i(1) f(1)	Social	1		u	1	1	1
Frozen+Gun+Games	i(1) f(1)	Arcade		1	i			
Dejobaan+Games,+LLC	p(1)	Action		1	i			
I+Fight+Bears	p(1)	Arcade		1	i			
Vlambeer	p(1)	Arcade		1	i			1
Wadjet+Eye+Games	p(1)	Role Playing		1	i	1		1
HyperBees	i(1) f(1)	Racing		1	o	1		1
DIGITAL+LEGENDS	i(1) f(1)	Action		1	p	1		1
Frogmind	i(1) f(1)	Action		1	p	1		1
Playrise+Digital+Ltd	i(1) f(1)	Racing		1	p	1		1
Alim+Co.,+Ltd.	i(1) f(1)	Role Playing		1	p	1		1
Nordeus	i(1) f(1)	Sports		1	p	1	1	1
Space+Ape+Games	i(1) f(1)	Strategy		1	p	1		1
Originator+Inc.	i(1) f(1)	Education		1	t	1		1
Crackle	f(1)	Entertainment			c	1		1
Telly,+Inc.	f(1)	Media & Video			c	1	1	1
COUNT TOTAL: 94			15	20		86	32	68

* The definition of each column is in the end of table.

Table A.19 Developers who only have one app - continue

Developer ID	Price	Primary Category	A	B	C	D	E	F
Comedy+Central	f(1)	Media & Video			c	1		
MTV	f(1)	Music & Audio			c	1		1
The+Washington+Post	f(1)	News & Magazines			c	1		1
Gmarket	f(1)	Shopping			c	1	1	1
Thrillist	f(1)	Shopping			c	1		1
Bleacher+Report+Inc.	f(1)	Sports			c	1		1
hotels.com	f(1)	Travel & Local			c	1		1
Expedia	f(1)	Travel & Local			c	1		1
Hipmunk	f(1)	Travel & Local			c	1	1	1
HotelTonight	f(1)	Travel & Local			c	1	1	1
Travel+Channel,+LLC	f(1)	Travel & Local			c	1		1
orbitz.com	f(1)	Travel & Local			c	1		1
n-tv+Nachrichtenfernsehen	i(1) f(1)	News & Magazines			c	1		1
Flixster+Inc.	f(1)	Entertainment			t	1		1
SweetTracker	f(1)	Lifestyle			t	1	1	1
Zite,+Inc.	f(1)	News & Magazines			t	1		1
EyeEm+Mobile	f(1)	Photography			t	1		
Flayvr+Media	f(1)	Photography			t	1	1	1
Zappos.com	f(1)	Shopping			t	1		
Whisk	f(1)	Shopping			t	1		
Dash	f(1)	Travel & Local			t	1		1
Voxy,+Inc.	i(1) f(1)	Education			t	1		
Picadelic	i(1) f(1)	Entertainment			t	1		1
zoobe+message	i(1) f(1)	Entertainment			t	1		1
Bitspin	i(1) f(1)	Lifestyle			t	1		1
MAGIX+Audio	i(1) f(1)	Music & Audio			t	1		1
Magisto	i(1) f(1)	Photography			t	1		1
Sumoing	i(1) f(1)	Photography			t	1		1
Any.do	i(1) f(1)	Productivity			t	1	1	1
Athentech	p(1)	Photography			t	1		
COUNT TOTAL: 94			15	20		86	32	68

* The definition of each column is in the end of table.

Table A.19 Developers who only have one app - continue

Developer ID	Price	Primary Category	A	B	C	D	E	F
Frograms+Inc.	f(1)	Entertainment			u	1	1	1
trailerpop,+inc.	f(1)	Entertainment			u	1		1
IMDb	f(1)	Entertainment			u	1		
UNPA	f(1)	Health & Fitness			u	1		1
yogiyo.co.kr	f(1)	Lifestyle			u	1	1	1
Tinder	f(1)	Lifestyle			u	1		1
Houzz	f(1)	Lifestyle			u	1	1	1
livelovely.com	f(1)	Lifestyle			u	1	1	
TuneIn+Inc	f(1)	Music & Audio			u	1	1	1
Soundwave	f(1)	Music & Audio			u	1		
SoundCloud	f(1)	Music & Audio			u	1	1	1
Digg	f(1)	News & Magazines			u	1		
Moment.me	f(1)	Photography			u	1	1	1
Trello	f(1)	Productivity			u	1		
Slice	f(1)	Shopping			u	1		1
Delivery+Club	f(1)	Shopping			u	1		1
ShopSavvy,+Inc.	f(1)	Shopping			u	1		
Mercari,+Inc.	f(1)	Shopping			u	1	1	1
OpenSky	f(1)	Shopping			u	1		
Poshmark,+Inc	f(1)	Shopping			u	1	1	1
Wish+Inc.	f(1)	Shopping			u	1	1	
Zoomingo+Local+Sales	f(1)	Shopping			u	1	1	1
Nextdoor	f(1)	Social			u	1	1	1
Pinterest,+Inc.	f(1)	Social			u	1	1	
Meetup	f(1)	Social			u	1	1	
Tumblr,+Inc.	f(1)	Social			u	1	1	1
OpenTable	f(1)	Travel & Local			u	1		
WhatsApp+Inc.	i(1) f(1)	Communication			u	1	1	1
musiXmatch	i(1) f(1)	Music & Audio			u	1		1
Keepy+Inc.	i(1) f(1)	Photography			u	1	1	1
COUNT TOTAL: 94			15	20		86	32	68

* The definition of each column is in the end of table.

Table A.19 Developers who only have one app - continue

Developer ID	Price	Primary Category	A	B	C	D	E	F
Hootsuite	i(1) f(1)	Social			u	1		1
Path,+Inc.	i(1) f(1)	Social			u	1	1	1
Littleinc+Labs	i(1) f(1)	Social			u	1	1	1
Pheed	i(1) f(1)	Social			u	1	1	1
COUNT TOTAL: 94			15	20		86	32	68

A = is the app in early stage?, B = is a game app? C = what is the competitive advantage? D = use personal user data, E = use social user data, F = use system user data

Competitive Advantage: u = has a large use community, t = has advanced technology, c = has a established core business to complement, i = is a independent developer rather than company developer, p = has product in multiple platform and provides seamless services, o = other.

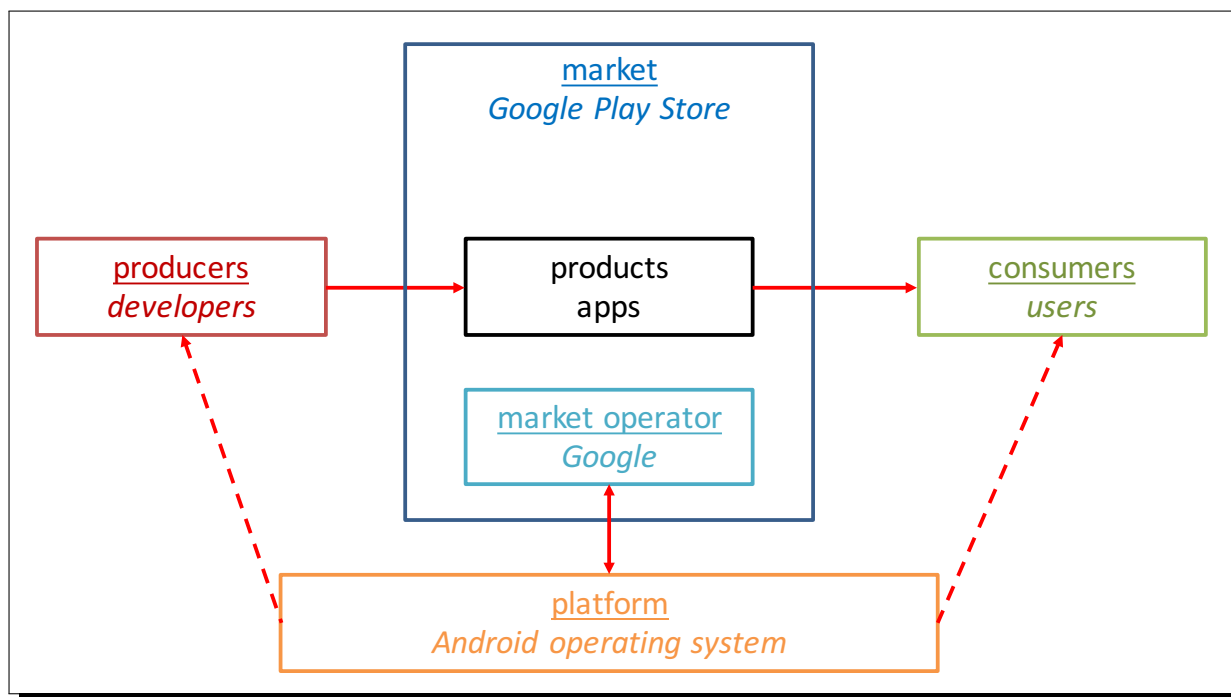
Price: the letter presents the price type for the developer's apps (i = in-app billing, f = free, p = paid), the number in bracket is the count of his apps in this price type. Remember, an app can use multiple price strategies.

Primary Category: it is the most used category by the developer's app.

There are 94 developers in the list. All developers have only one app and was featured by Google during my data collection period. This list is to study why developer do not diverse.

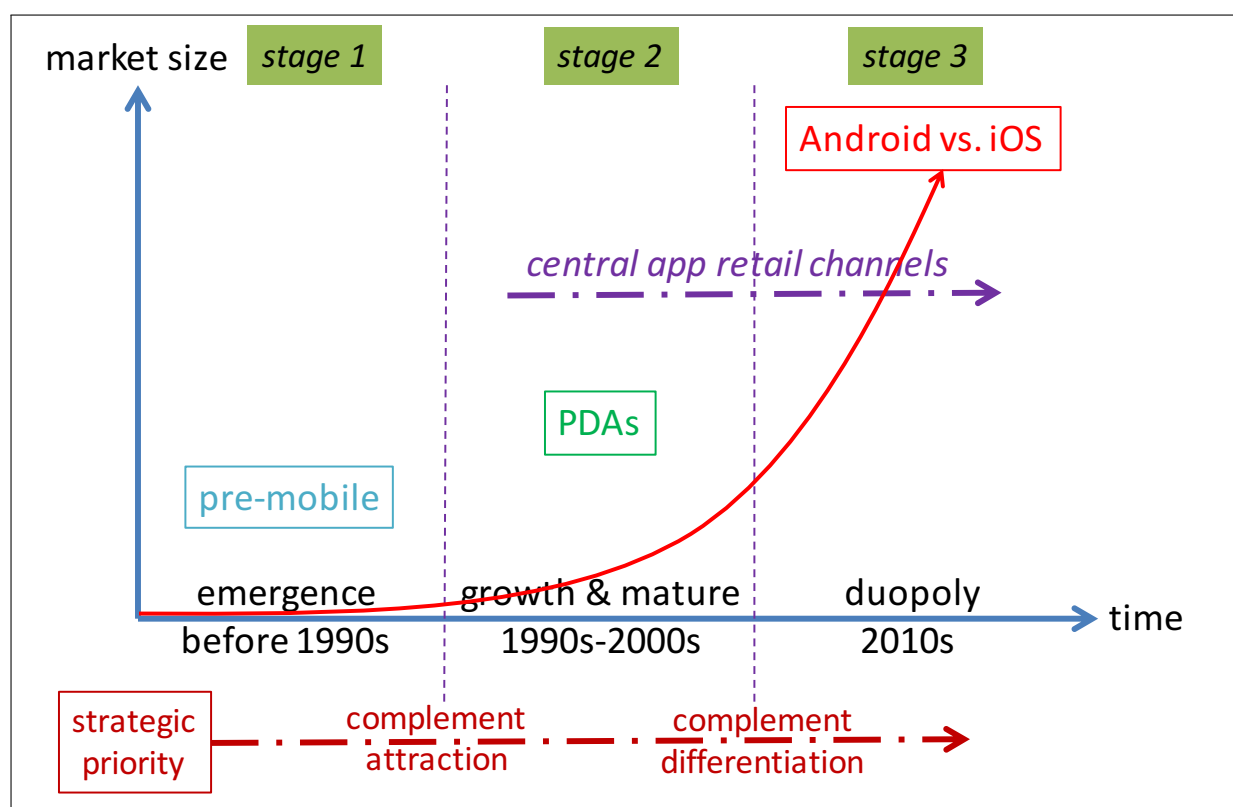
A.5 Other Figures

Fig. A.1 The *Android* app market

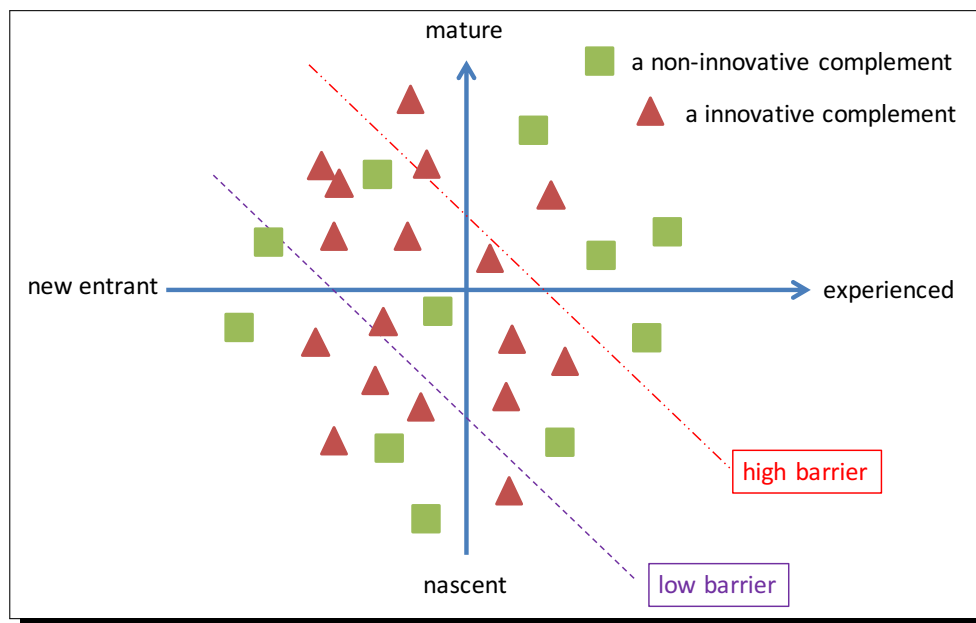
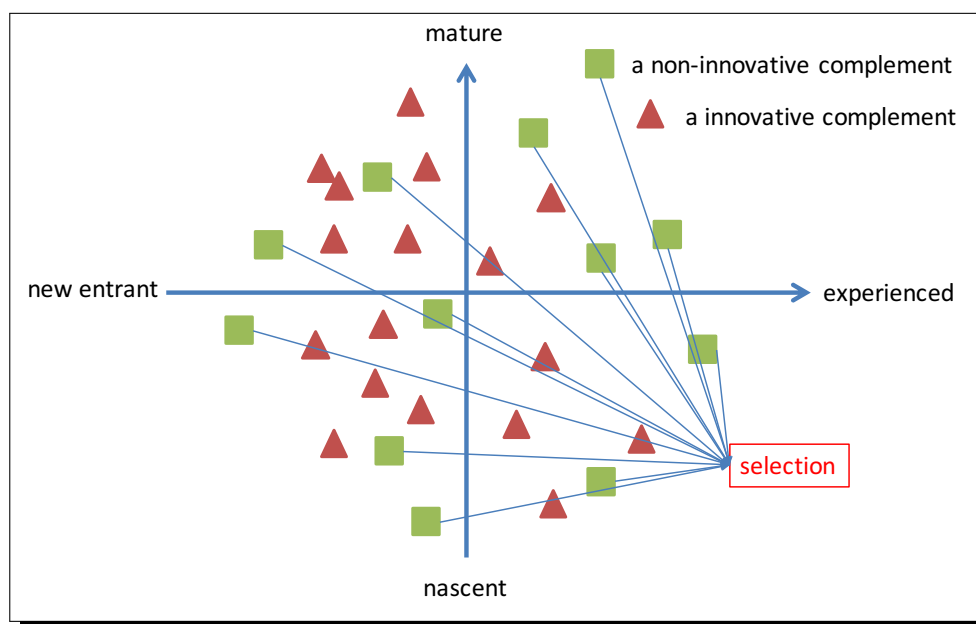


Use *Android* to illustrate a two-sided app market. Apps are the products to trade in the *Google Play Store* (market). Developers (producers) create apps to sell in the market and users (consumers) purchase apps in the market. Apps are built and running on top of *Android* technology (platform). *Google Play Store* is owned and managed by *Google* (market operator). *Google* is also the primary creator and maintainer of *Android* technology. In some other two-sided market, platform owner is not necessary to be the market operator.

Fig. A.2 Three stages of app economy

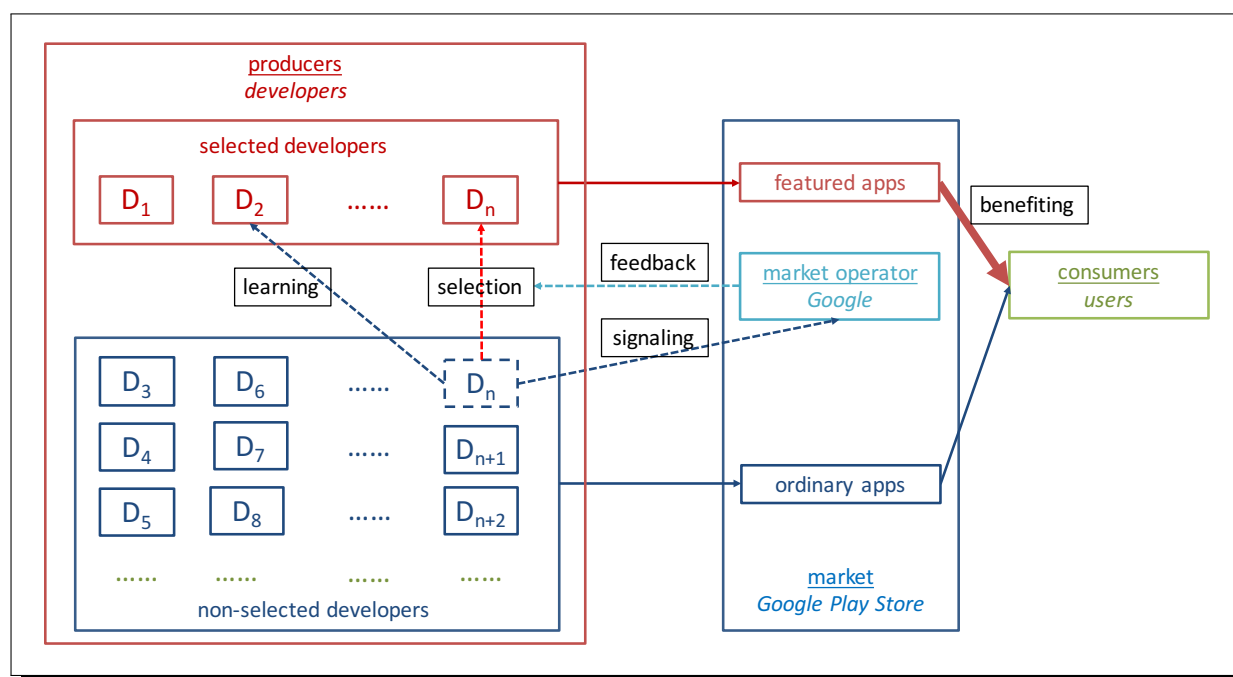


App economy has been categorized into three stages in the thesis according the scenario of market size and status. The popular of central app retail channels is to distinguish the "old" and "modern" app economy. The size and concentration of app market is to determine the second and third stages.

Fig. A.3 Differentiation of complements**(a)** Barrier-based**(b)** Selection-based

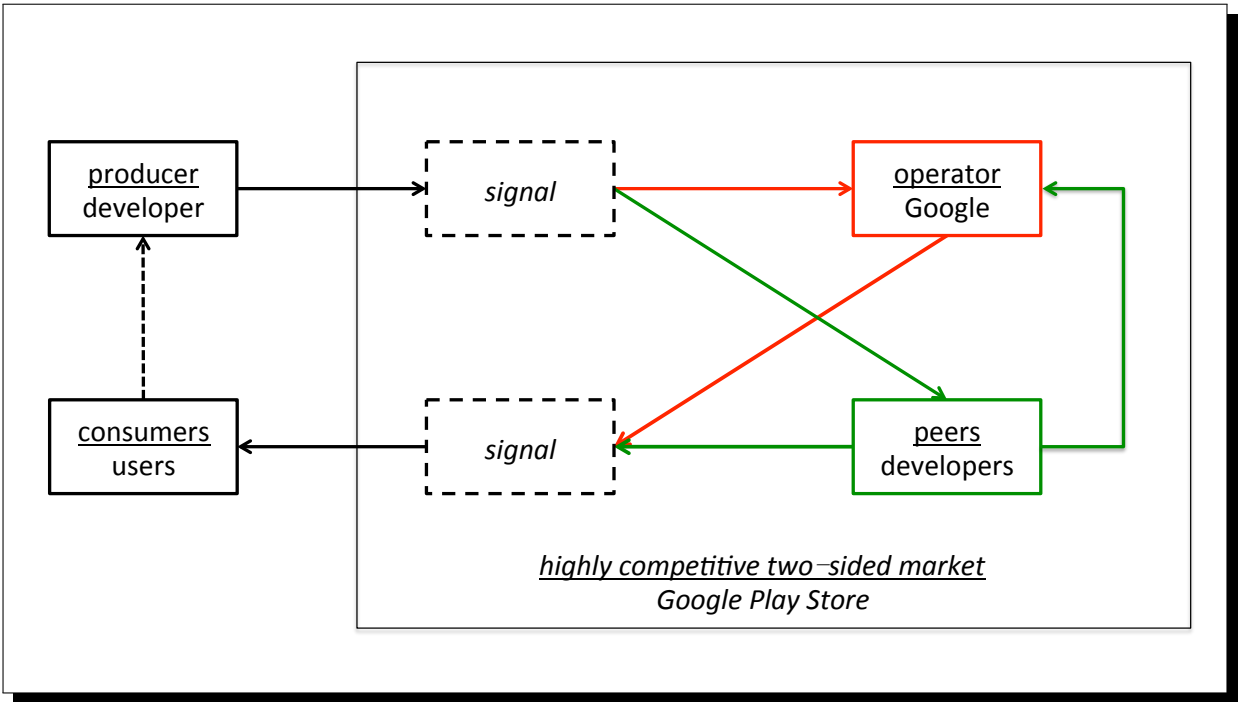
A two-sided platform based market is full of complementers who have different combinations of experience and establishment. Some of them may join the market easier and some of them may focus on mature business sectors. Each combination can have innovative and non-innovative complements. Existing platform strategies have emphasis on the barrier-based differentiation of complements in innovativeness, which would filter out a lot of non-innovative complements, but also increase the opportunity cost for new or "small" entrants who are "potentially" innovative. Meanwhile, some non-innovative complementers in the upright corner still can not be filtered out by barrier-based differentiation even by setting up high barrier. Even more, barrier-based will reduce the total number of participants in complements which will reduce the network effects to attract for more complements and market size.

Selection-based mechanism is proposed to solve the limitations of barrier-based mechanism in this thesis. It emphasizes to select complements based on their intrinsic innovativeness, no matter their current status or competitiveness which create the fairness of competition. It does not recommend the platform to increase the opportunity cost, so it will not impact on the total number of complements to weaken the network effect. It suggests platform to select innovative complementers and give them extra benefits to stimulate other non-selected complementers to learn or improve their innovativeness.

Fig. A.4 The signaling selection strategy in app economy

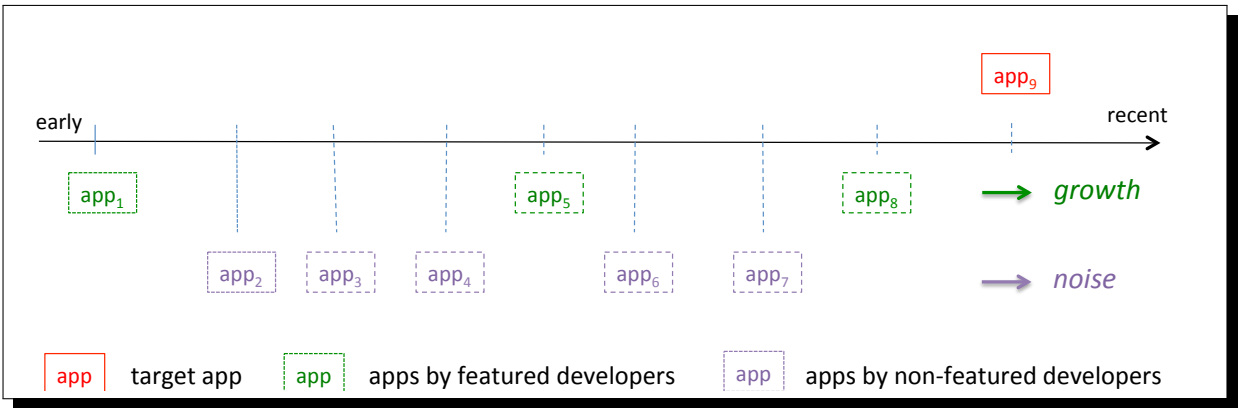
It is a strategy aiming to efficiently identify developers' ability and select desired developers to feedback with great benefits. Developers conduct certain costly actions to generate signals. Developers with high ability are more likely to conduct those actions as they have lower opportunity cost comparing to low ability developers. The market operator will observe those signals and select desired developers to feedback with great benefits. The great benefits will attract other non-featured developers to learn from selected developers. This enables the market operator to manage the market by focusing on selecting the desired developers, and the learning between developers will spread market operator's desire widely to the whole developer community.

Fig. A.5 Signaling Model in App Market



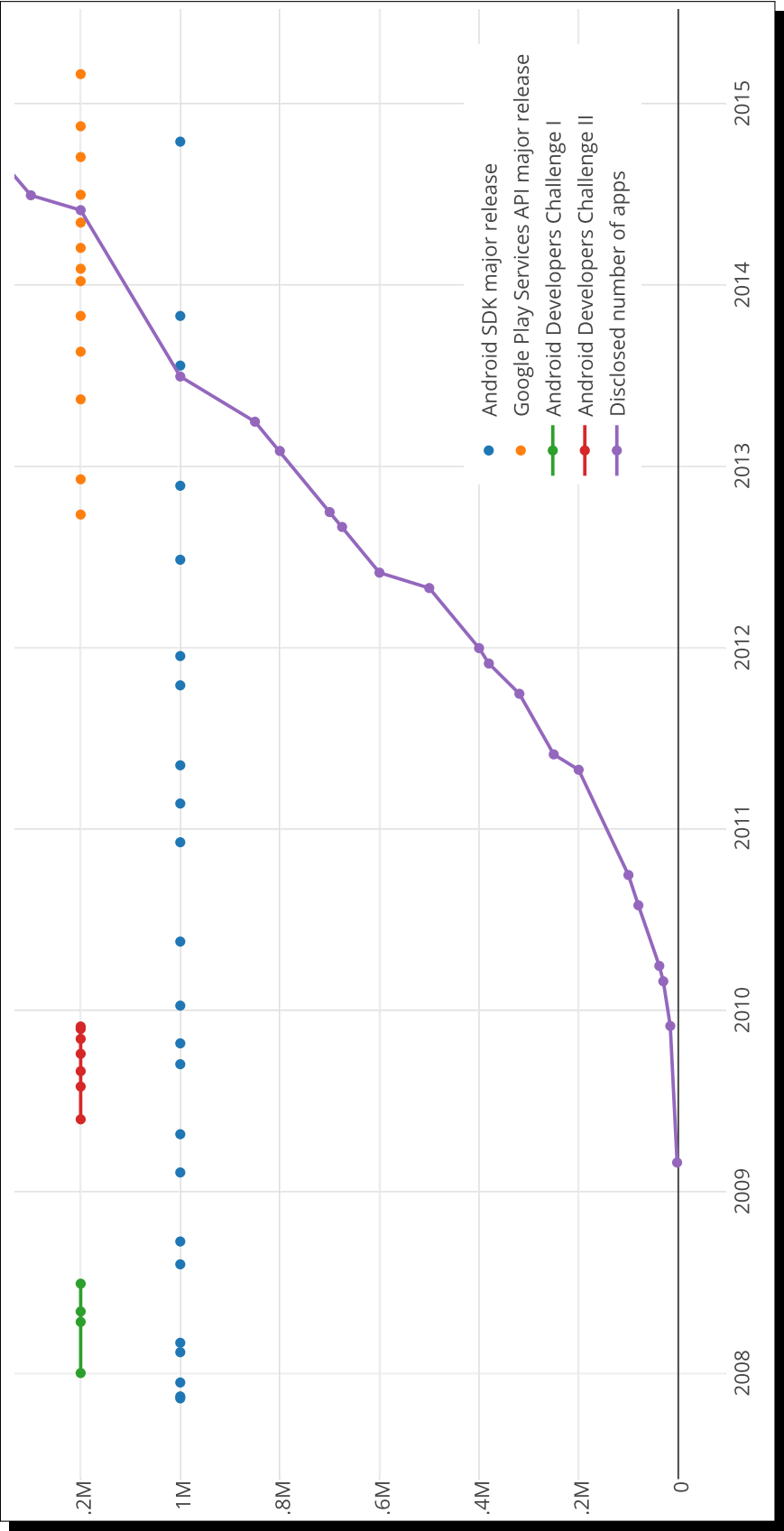
Users purchase apps through market. Due to the high competition and information asymmetry, developers have to influence user purchase via indirect reciprocity by platform. Therefore, developer signals to market operator for platform selection, which is to obtain a featuring status. Developer also signals to other developers for peer recognition, and this will also be complementary to platform selection.

Fig. A.6 Who to follow and signal?



Following good developers would attract more users to increase the market size. In opposite, apps by ordinal developers would add “noise” by damaging user experience, disordering competition and confusing users’ purchase selection.

Fig. A.7 *Android* key events



The figure includes two sources of information. The first source is in dot to show the key events in *Android* history. The other source is to benchmark the disclosed number of available apps in *Google Play Store*. More details about *Android* key events can be found in Table A.11.

Fig. A.8 Example of an app profile

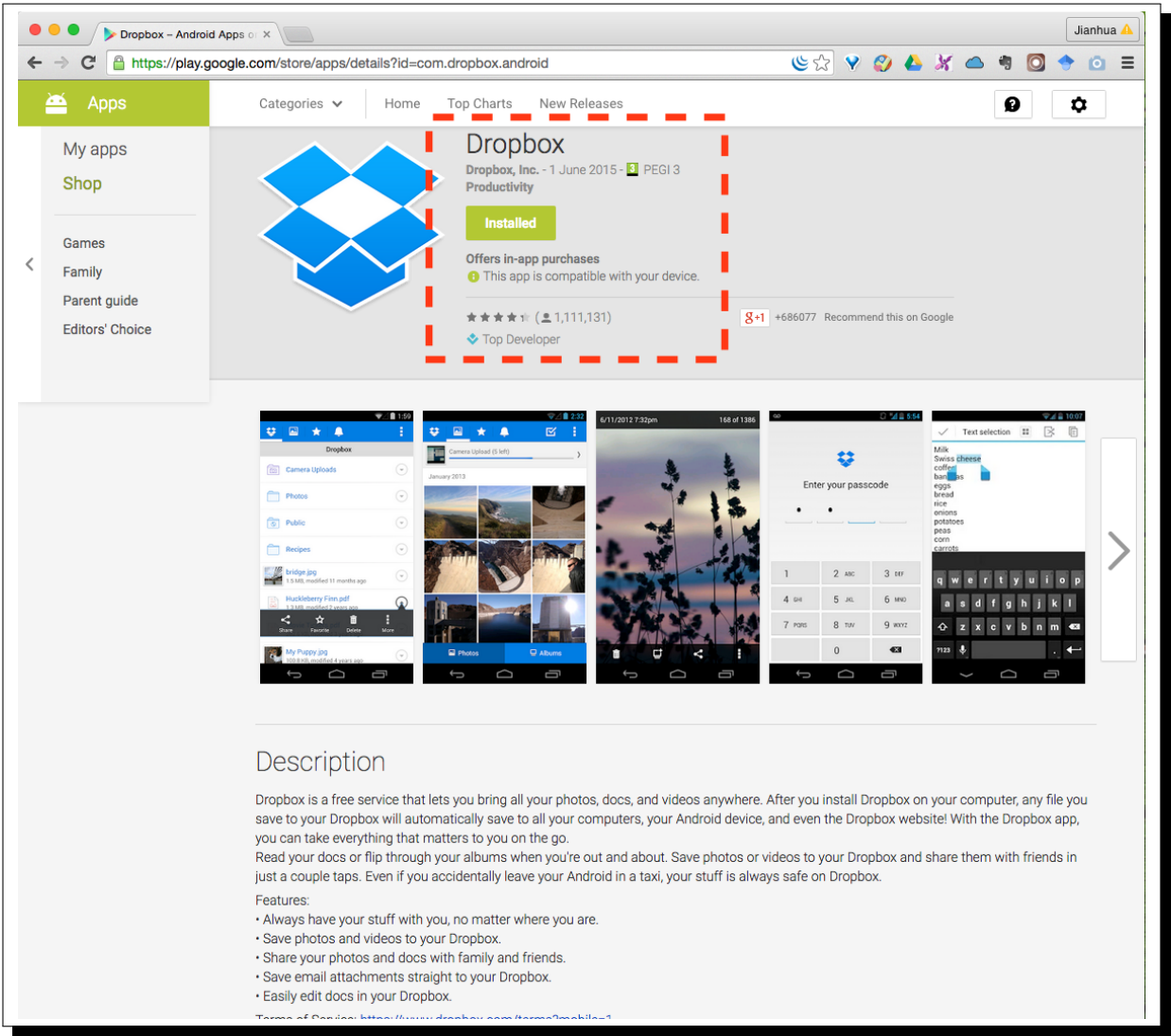


Fig. A.9 User review and other meta data in an app profile

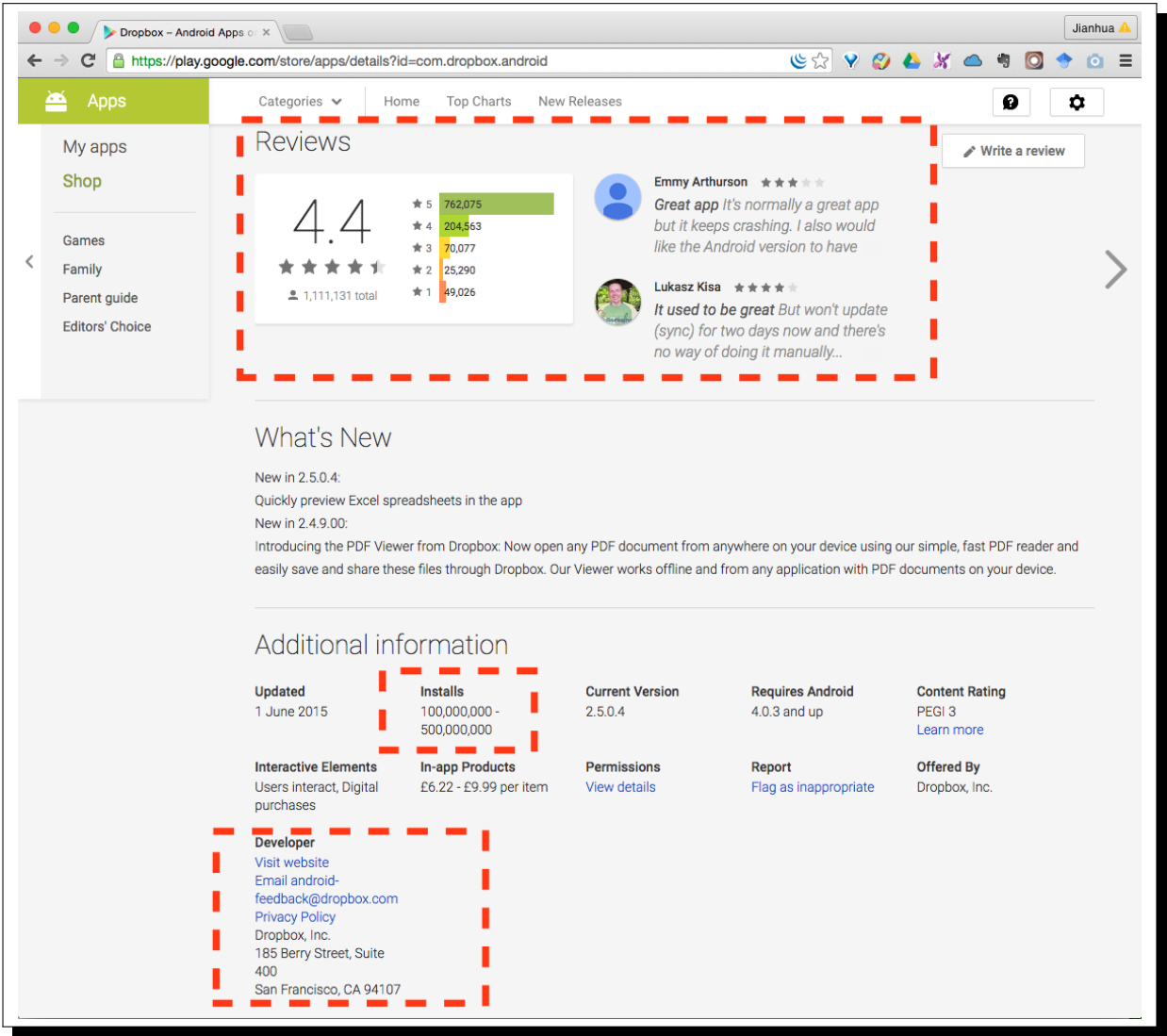


Fig. A.10 List of relevant apps in an app profile

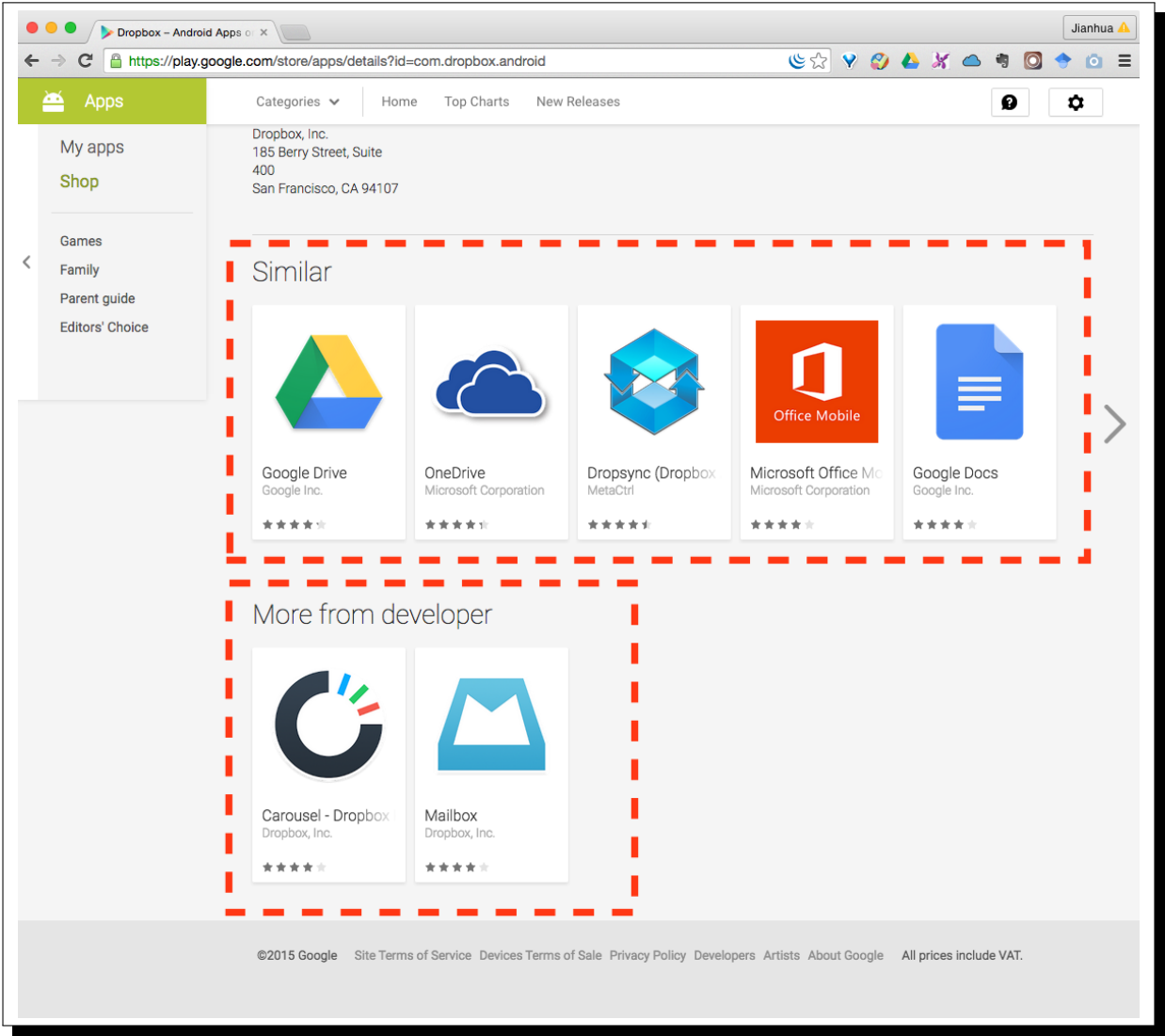


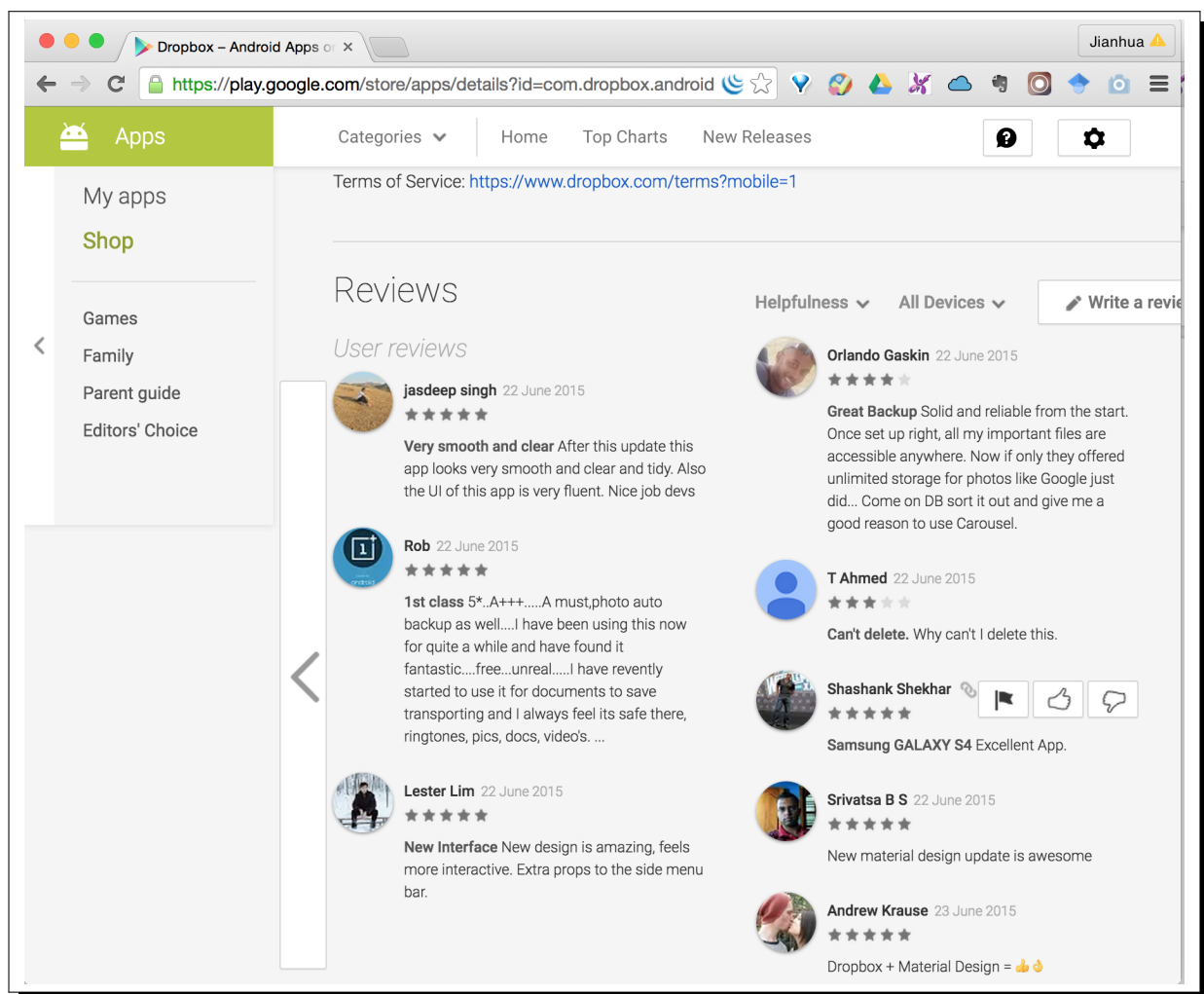
Fig. A.11 Examples of user review comment in an app profile

Fig. A.12 Top apps in an example category

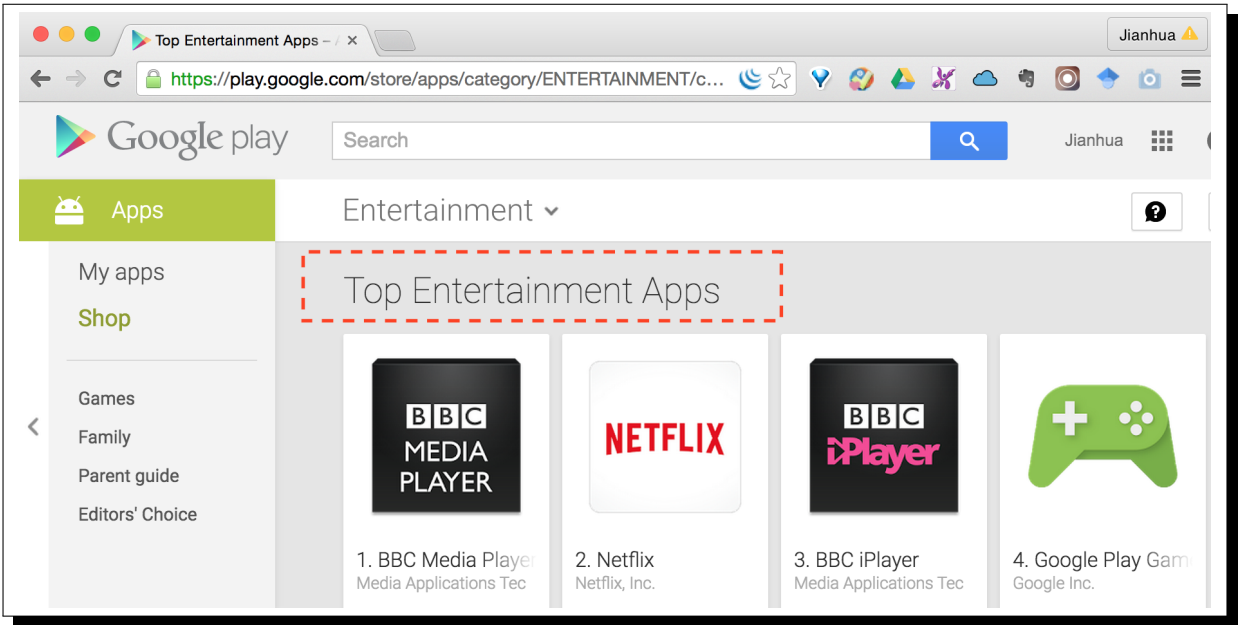


Fig. A.13 A screenshot of review meta data for an app

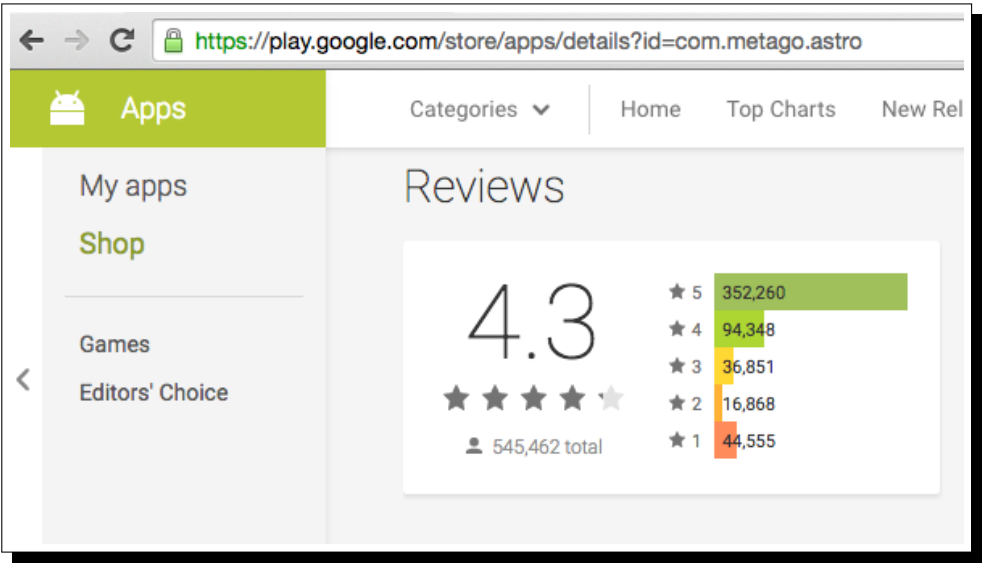


Fig. A.14 A developer who received two selection rewards

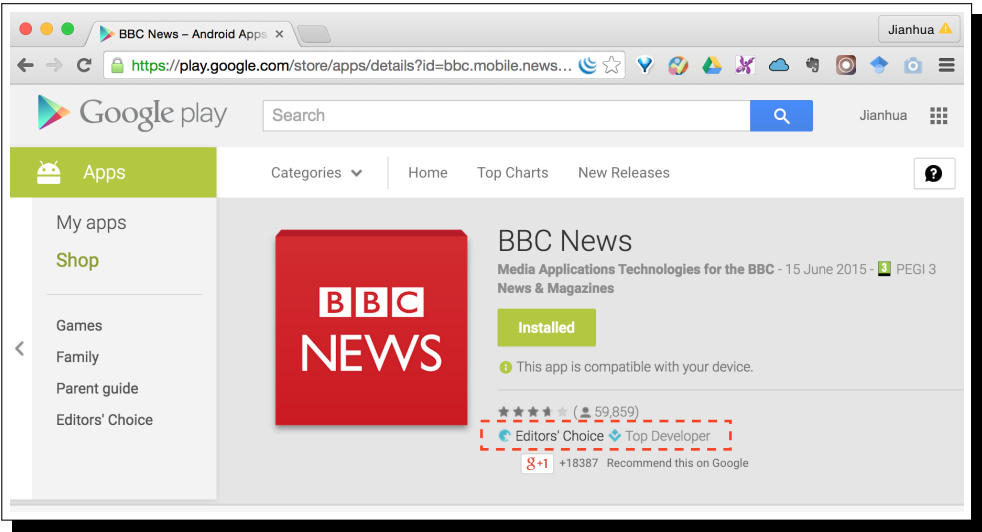


Fig. A.15 History of available Android apps in Google Play Store

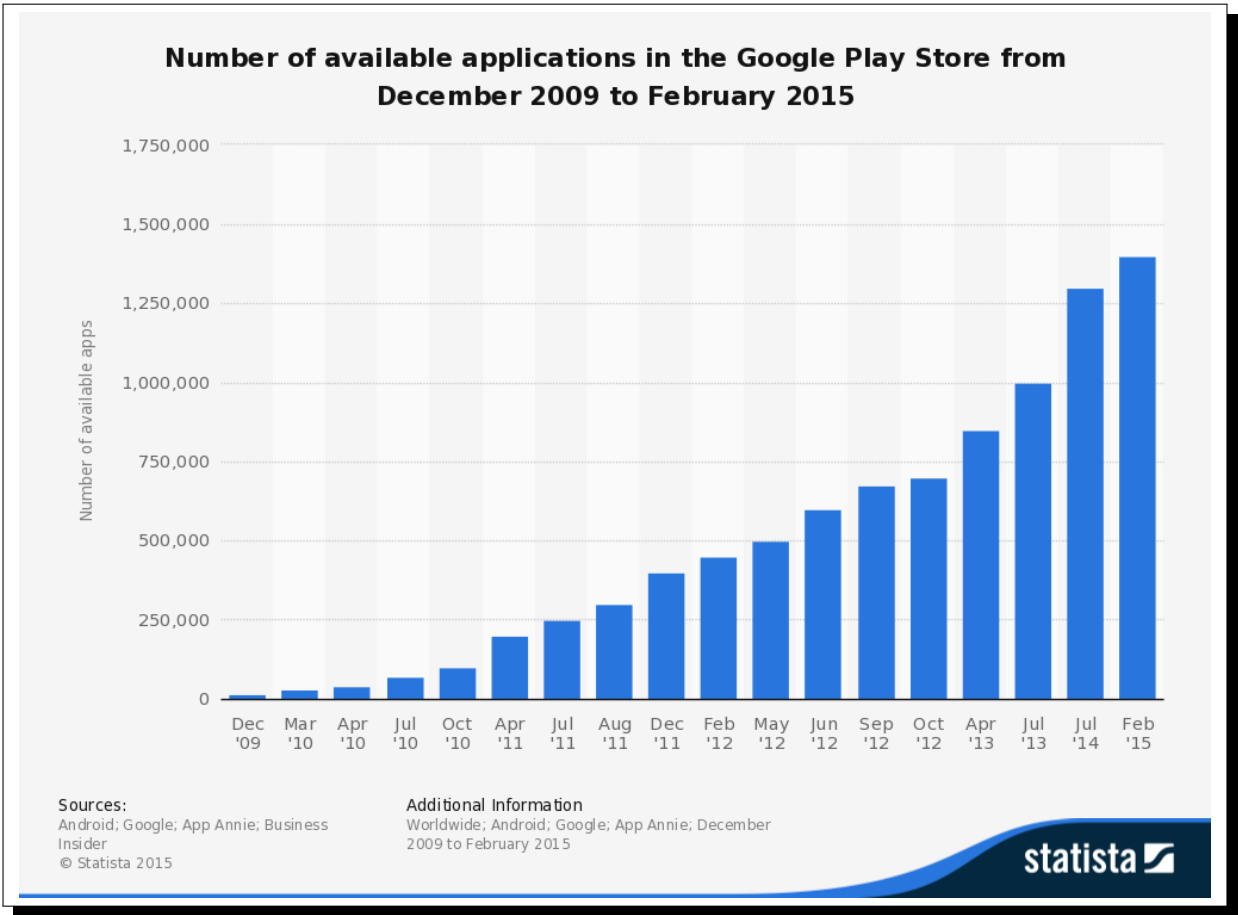


Fig. A.16 Google Play Store’s front page

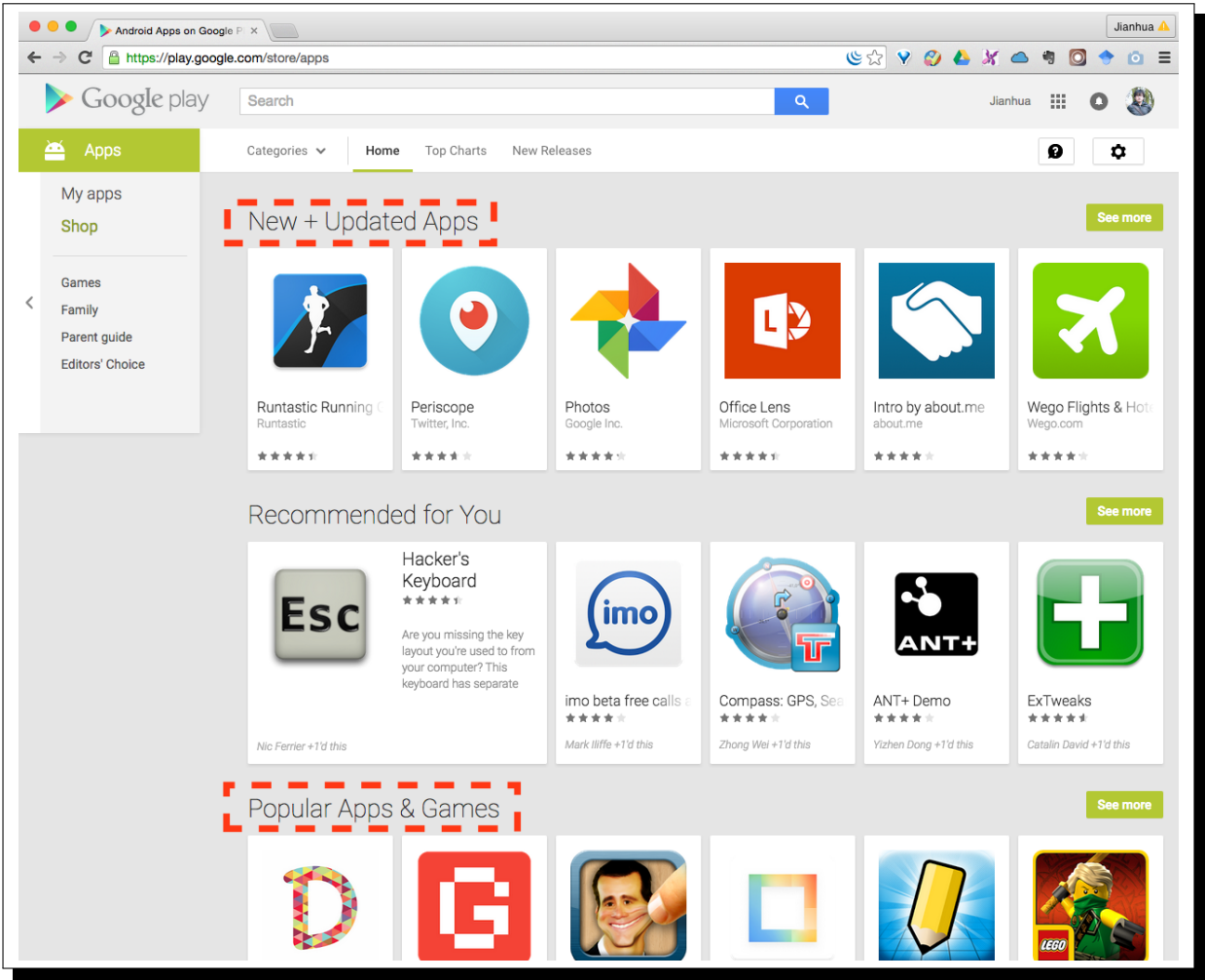


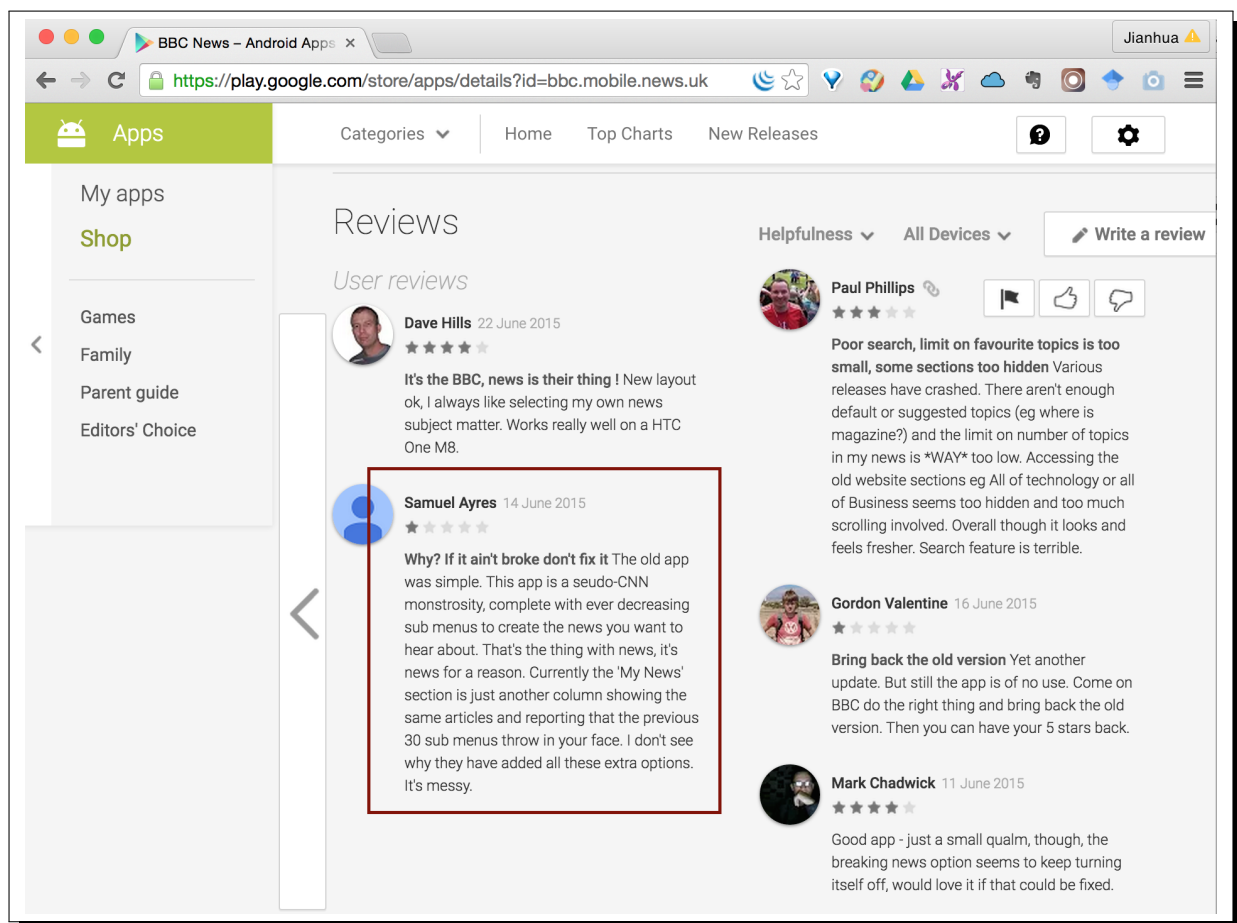
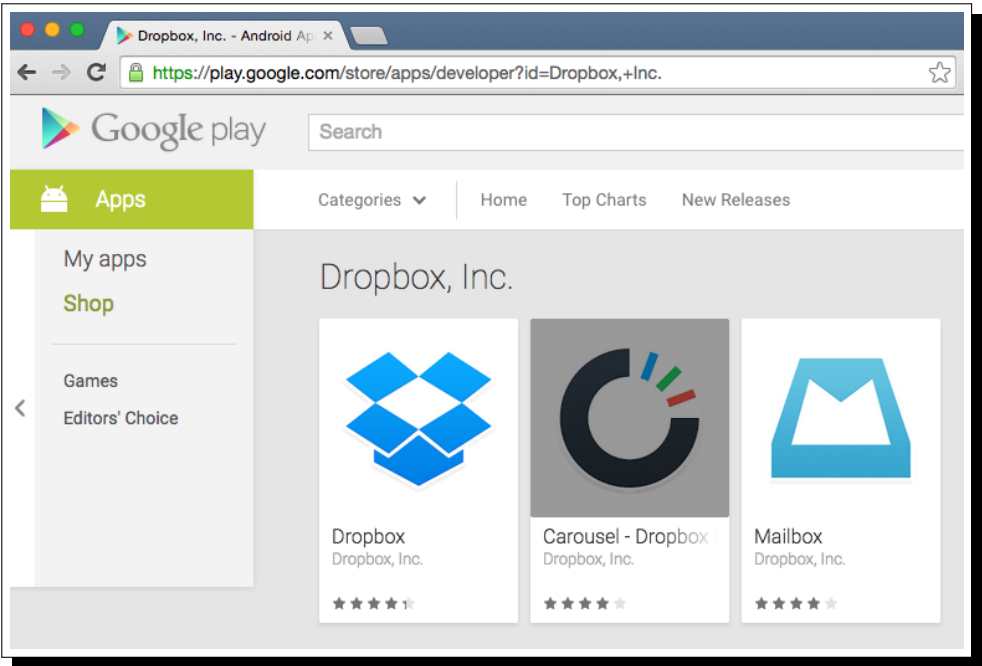
Fig. A.17 Example of lead user who tried to challenge the developer

Fig. A.18 Example of a developer profile



Google Play Store enables people to easily review every apps from a developer.

Fig. A.19 Permission requested by a particular app during installation

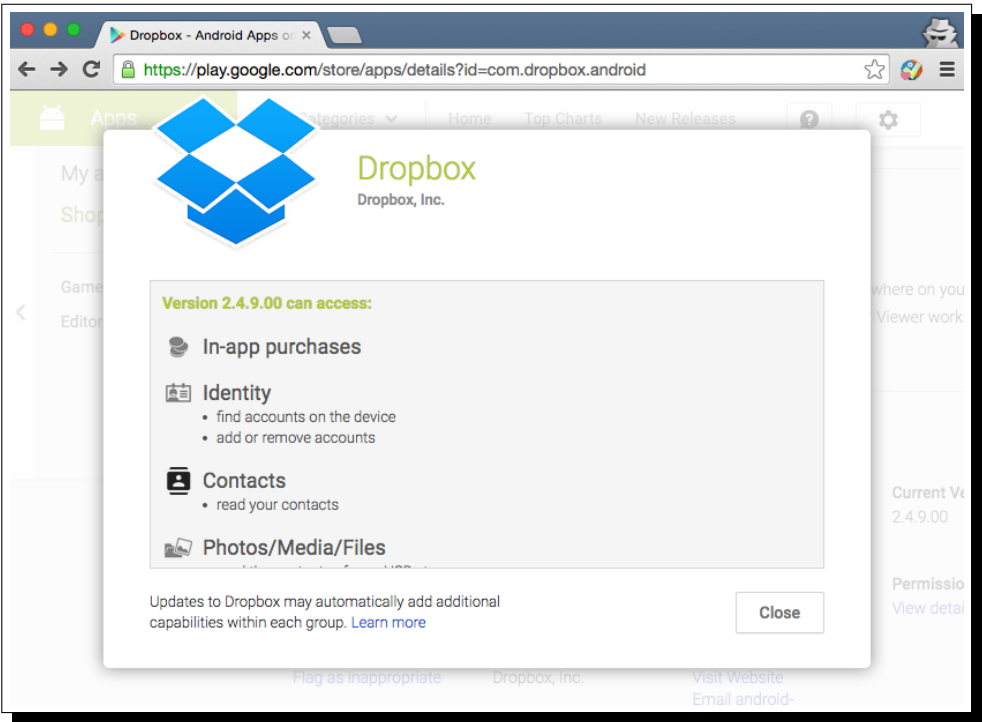


Fig. A.20 Web archive in Wayback Machine for a particular app

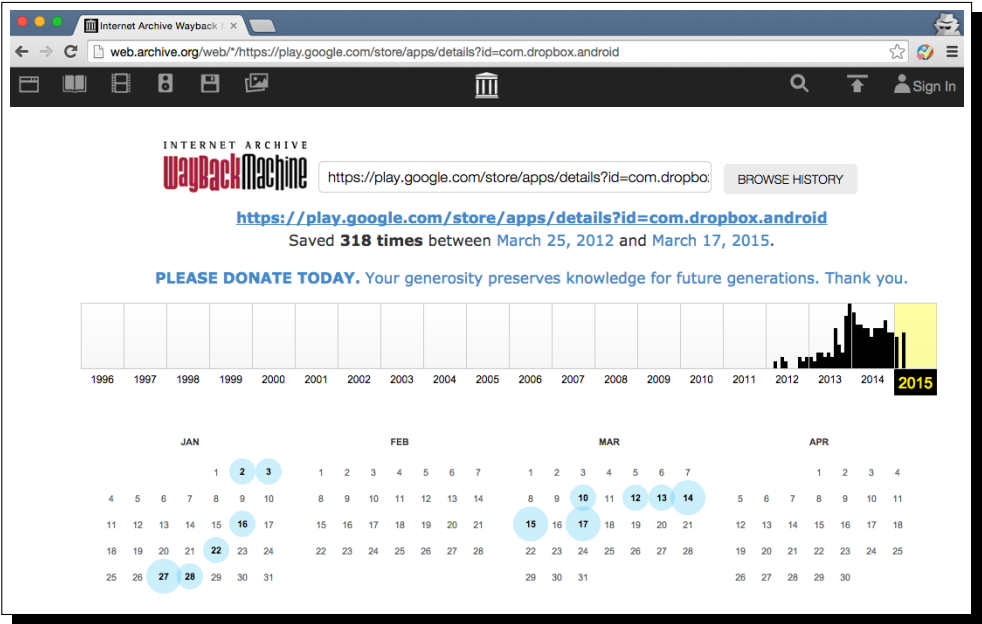


Fig. A.21 Distribution of counting of unique categories by featured developers

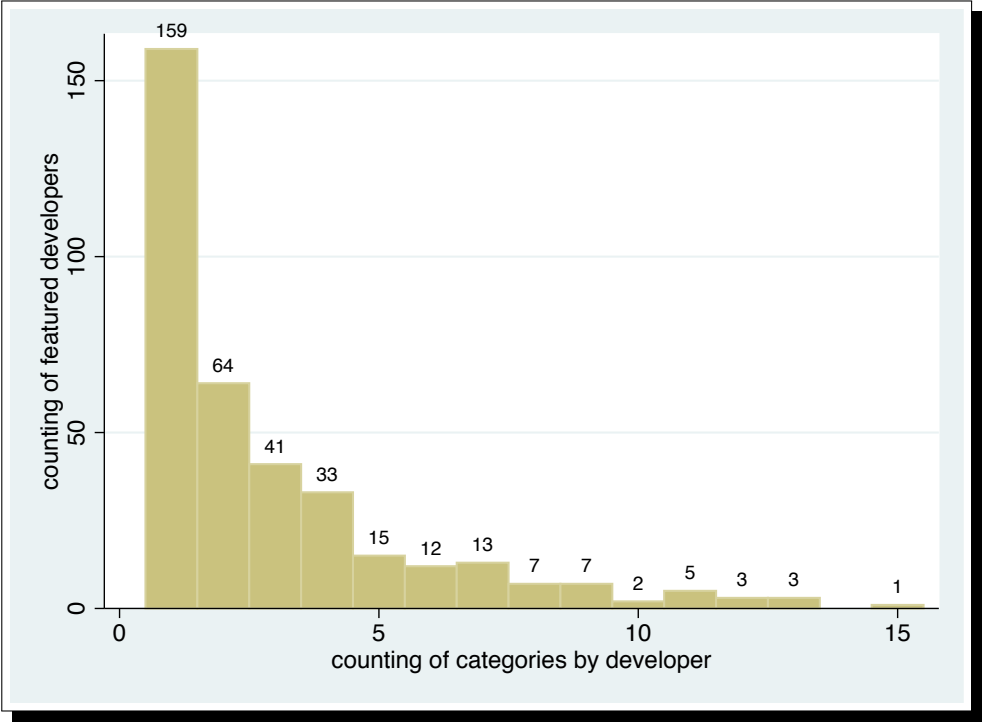
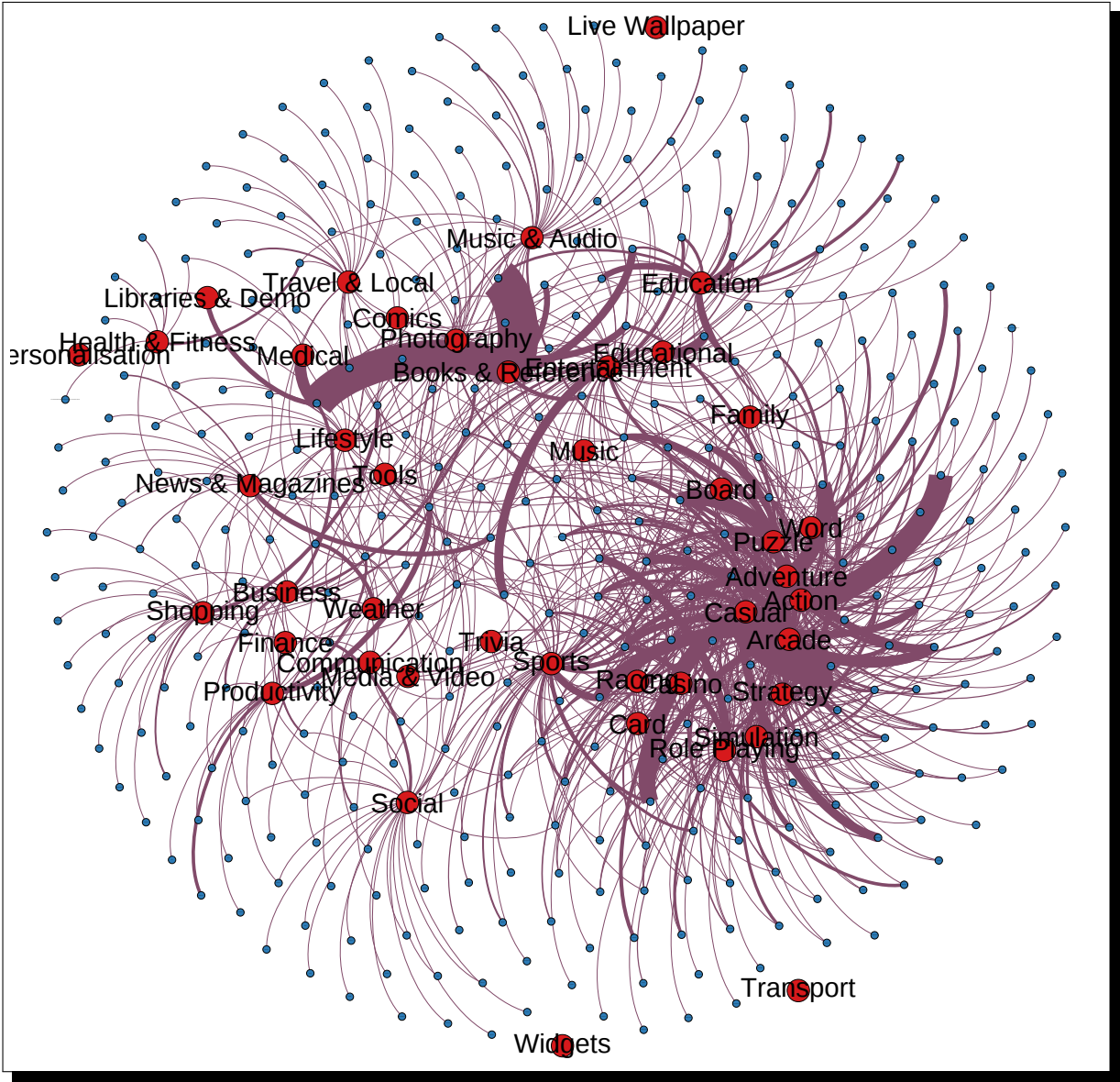
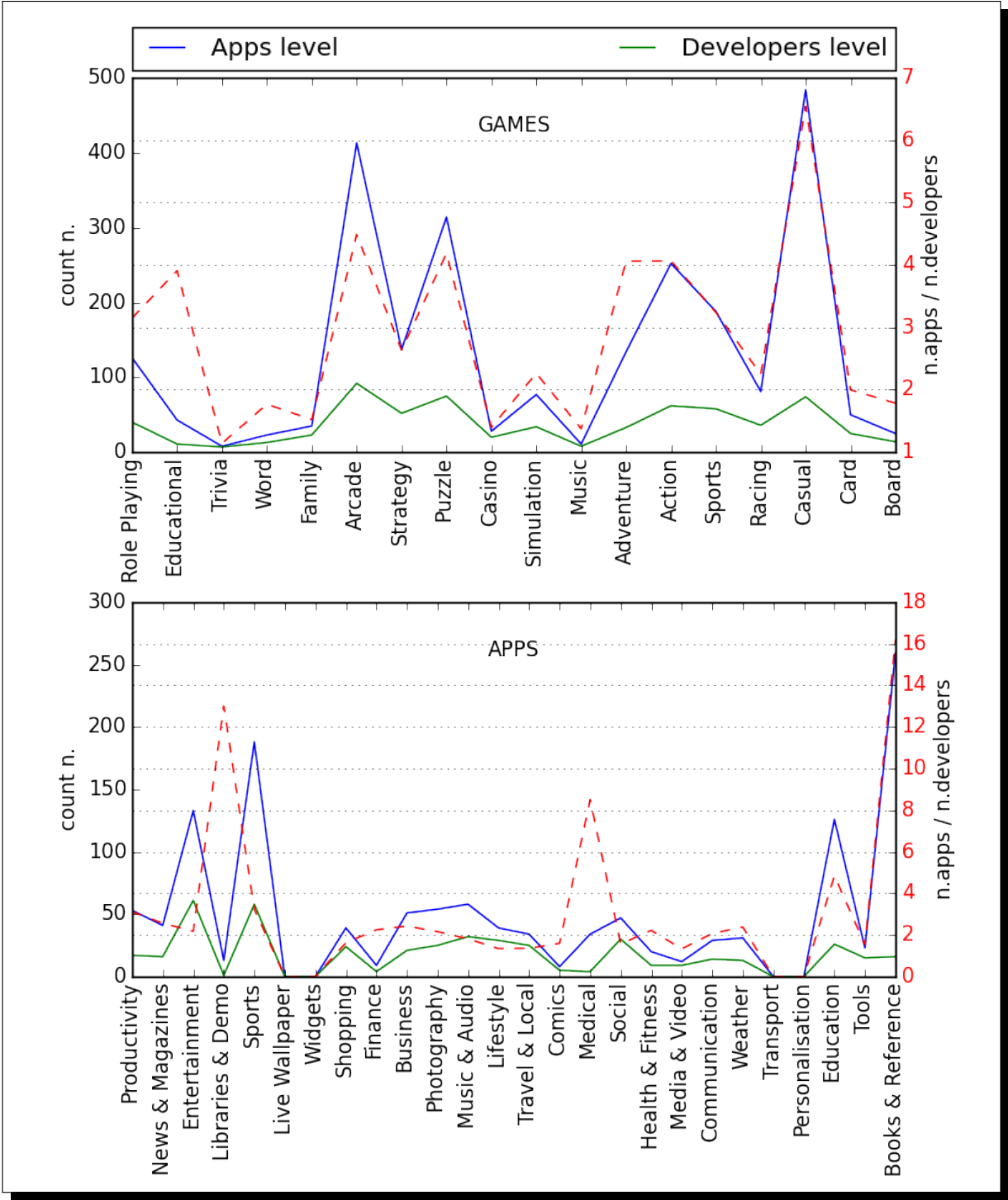


Fig. A.22 Network of featured developers and app categories



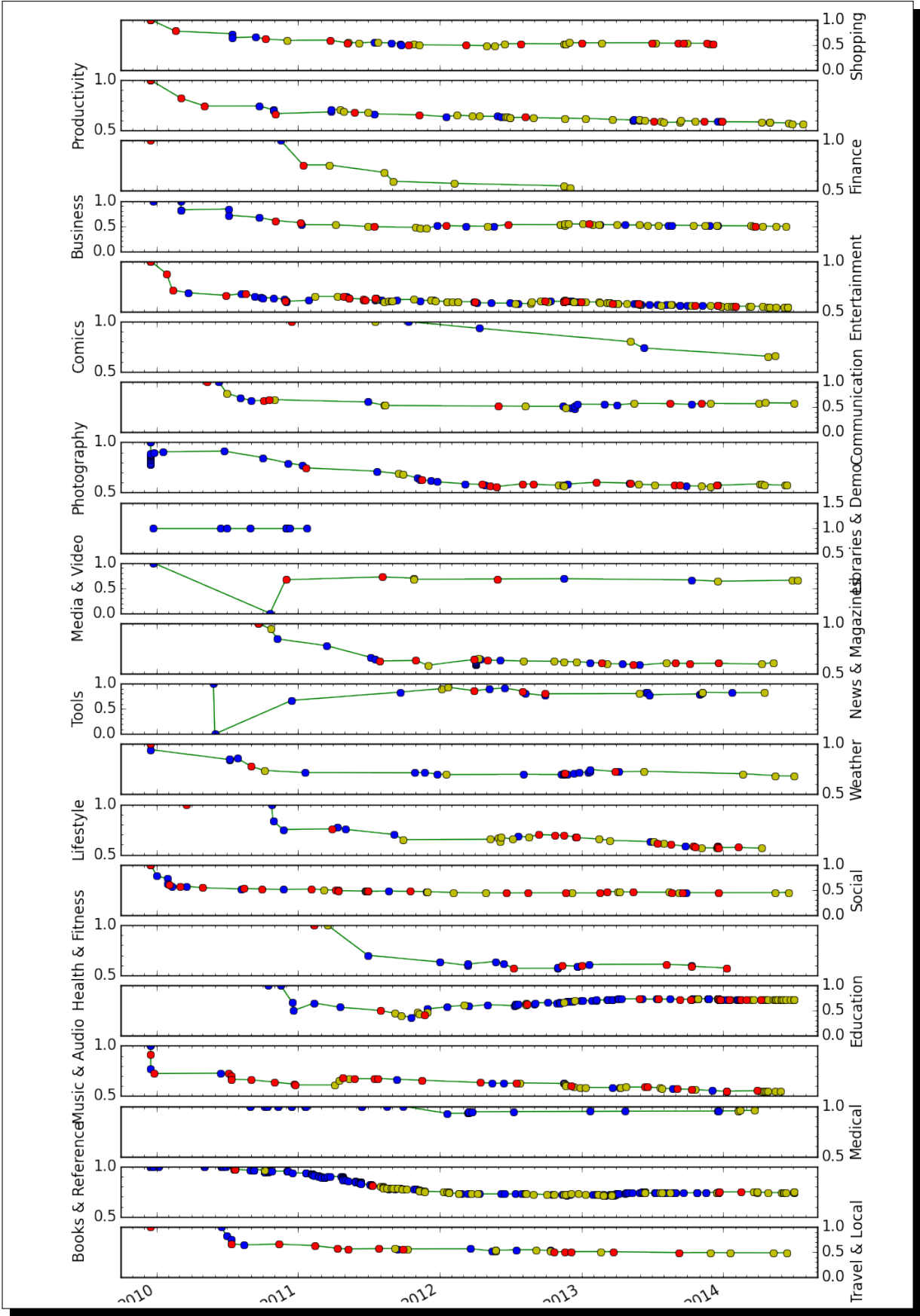
This diagram shows the relationship between developers and the category of their apps. Each *blue dot* represents a developer and each *red dot* represents the category to which apps are developed for. There is a *link* if a developer has been developed a app for the category. The *weight of link* represents the number of apps that a particular developer have in that category. The developer who focus on one category will try to stay away from the center and the developer who diversifies will stay crowded in the middle.

Fig. A.23 Distribution of featured developers and their apps across app categories



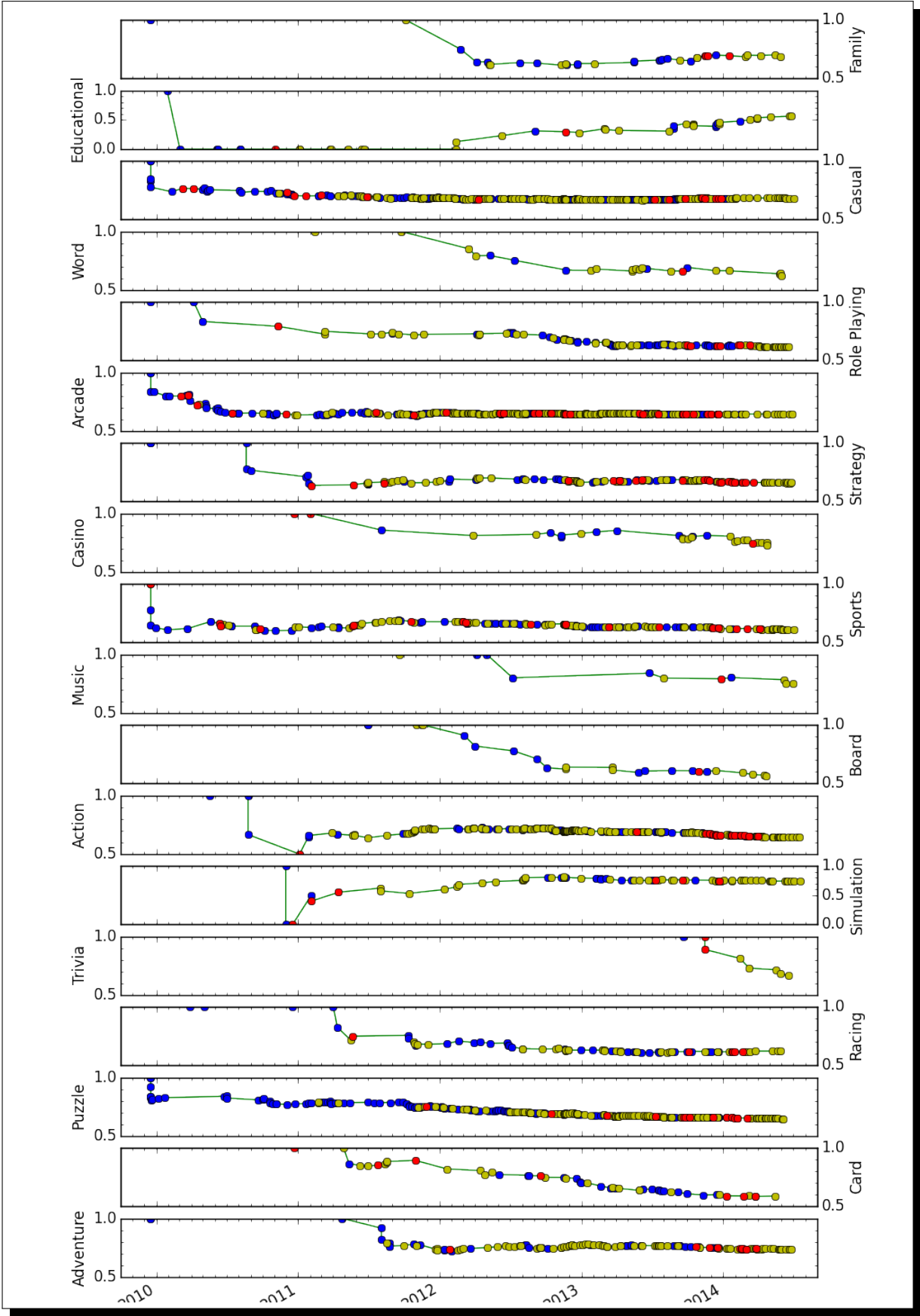
In *Google Play Store*, both game and non-game apps are not having equal distribution in each category. As expected, even all developers are featured, the number of apps (*blue line*) in each category is normally higher than the number of developers (*green line*), and this ratio (*red dash line*) is extremely high in certain categories like *Casual* in *GAMES* and *Book & Reference* in *APPS*. It partially proves that some developers prefer to focus on small number of apps and some other developers prefer to have large number of apps to diversify the opportunity.

Fig. A.24 Uniqueness change of apps by featured developers in non-game categories



The uniqueness is calculated in the distinction of permission use (see more in Section 8.2.2). The dot color represents the developer's featuring status when releasing the app. *Blue dot* means the developer was not featured when releasing that app. *Red dot* means this app was the first app when its developer was featured. *Yellow dot* represents the app released after developer was featured.

Fig. A.25 Uniqueness change of apps by featured developers in game categories



The uniqueness is calculated in the distinction of permission use (see more in Section 8.2.2). The dot color represents the developer's featuring status when releasing the app. *Blue dot* means the developer was not featured when releasing that app. *Red dot* means this app was the first app when developer was featured. *Yellow dot* represents the app released after developer was featured.

Fig. A.26 Distribution of Permission Requested by Apps

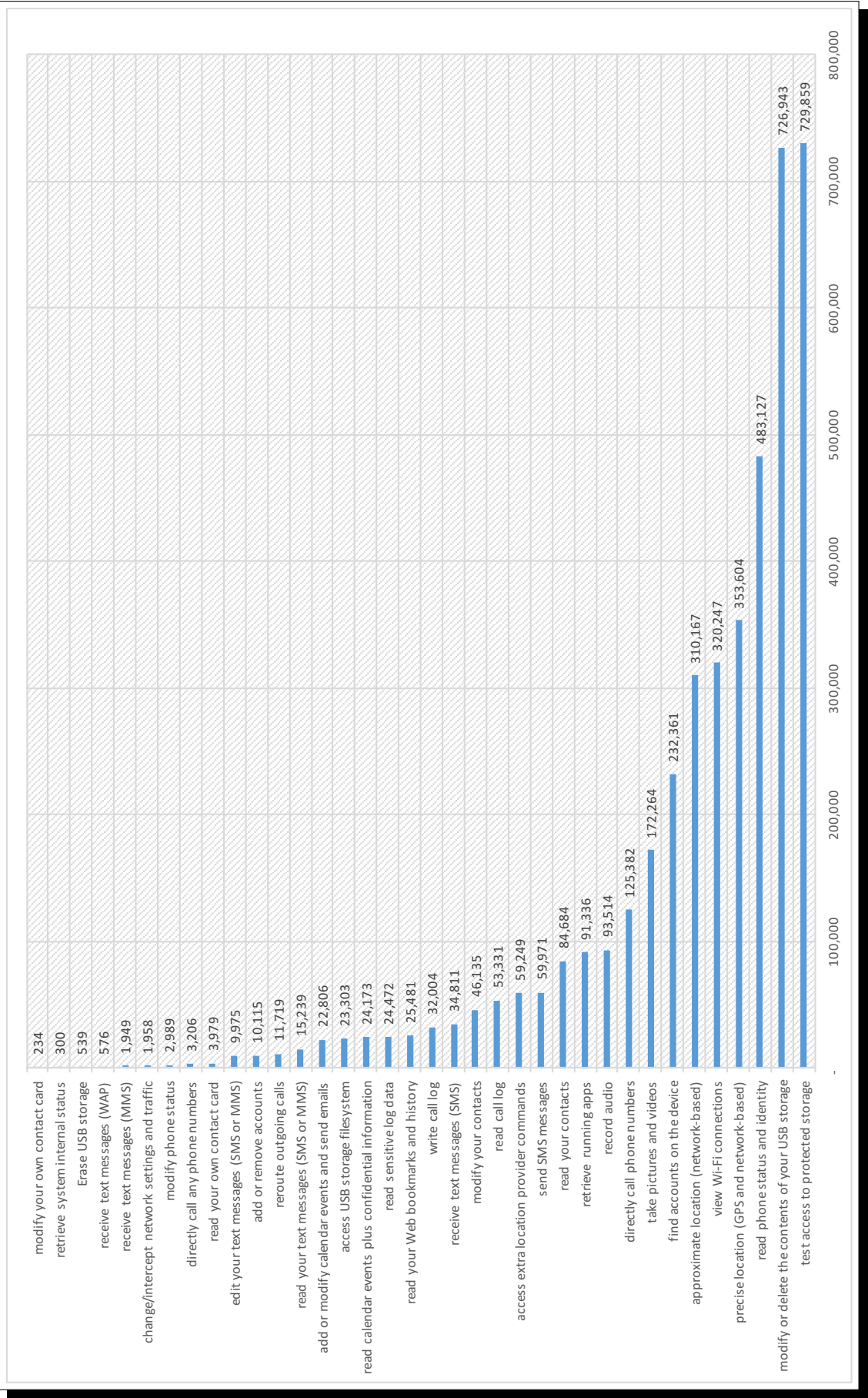


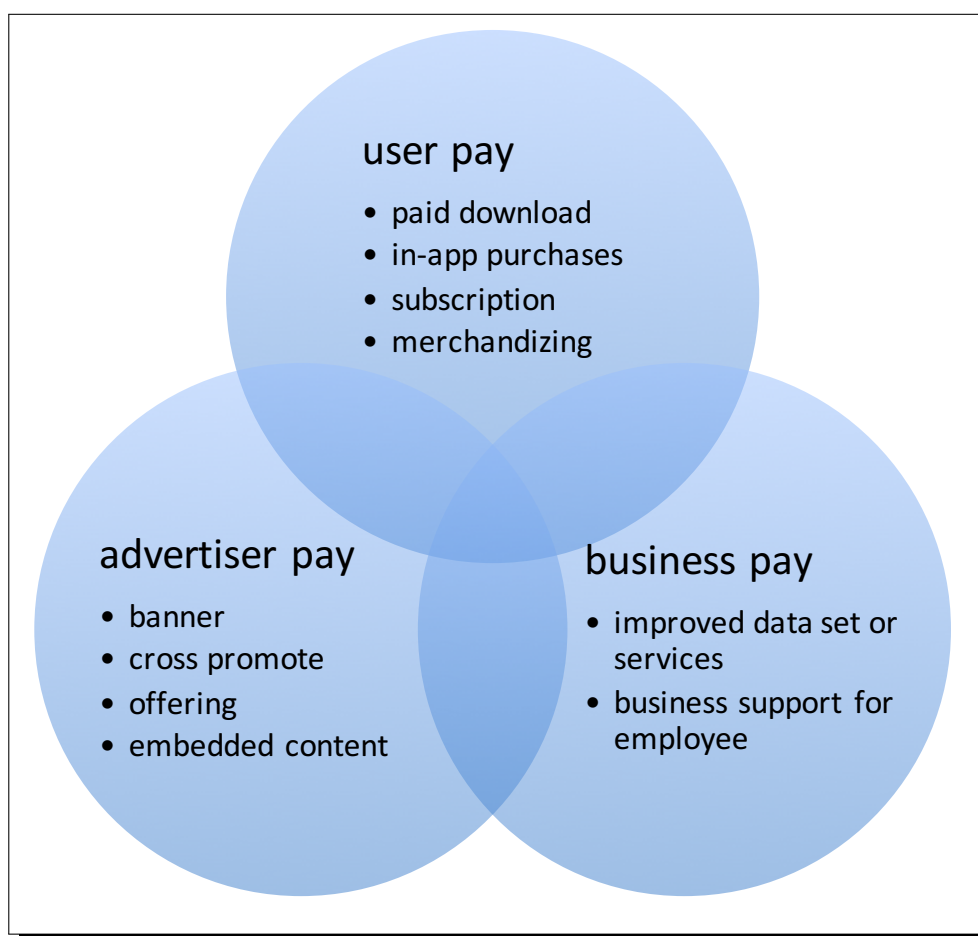
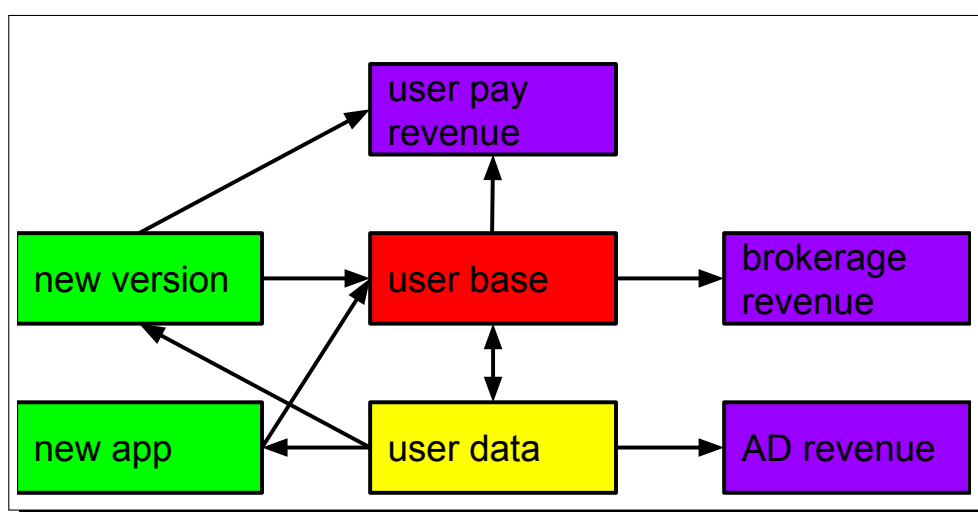
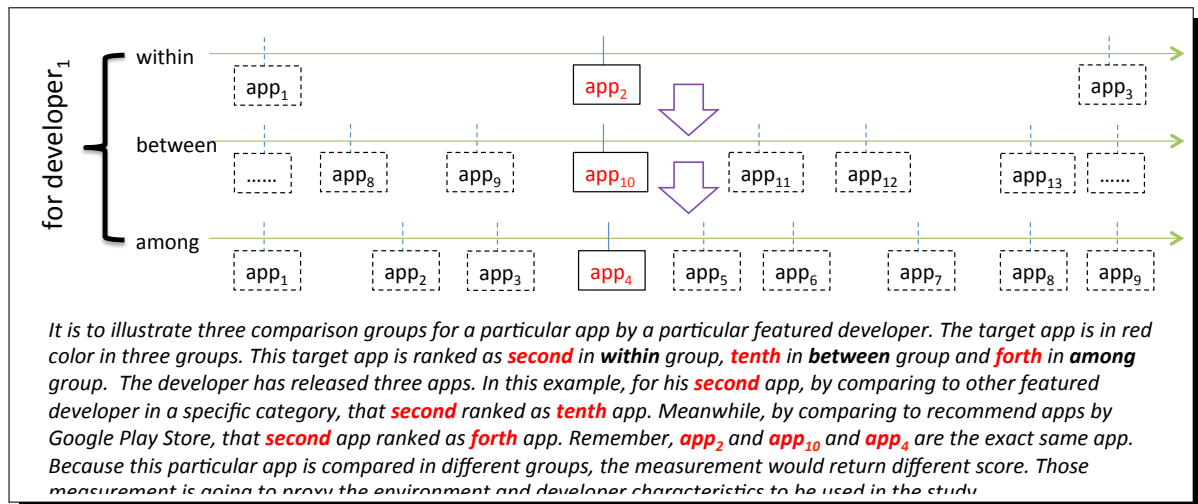
Fig. A.27 Monetization of user data in app economy**Fig. A.28** Value creation with user data

Fig. A.29 Comparison groups

(a) An illustration of time order in **among** group

(b) An illustration of three comparison groups for a particular app

It demonstrates how three comparison groups are selected and set to a target. It also shows how time events are set and measured. More details about comparison groups can be found in Section 8.1

Fig. A.30 Unit diagram of multilevel analysis in the study

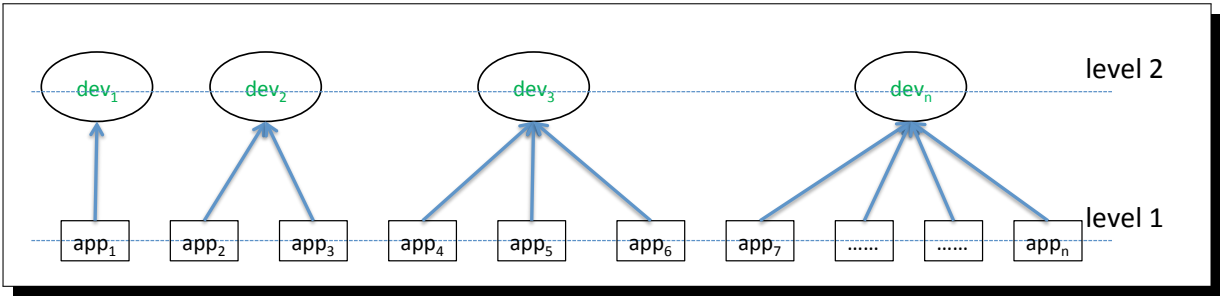


Fig. A.31 Market Performance Measurement

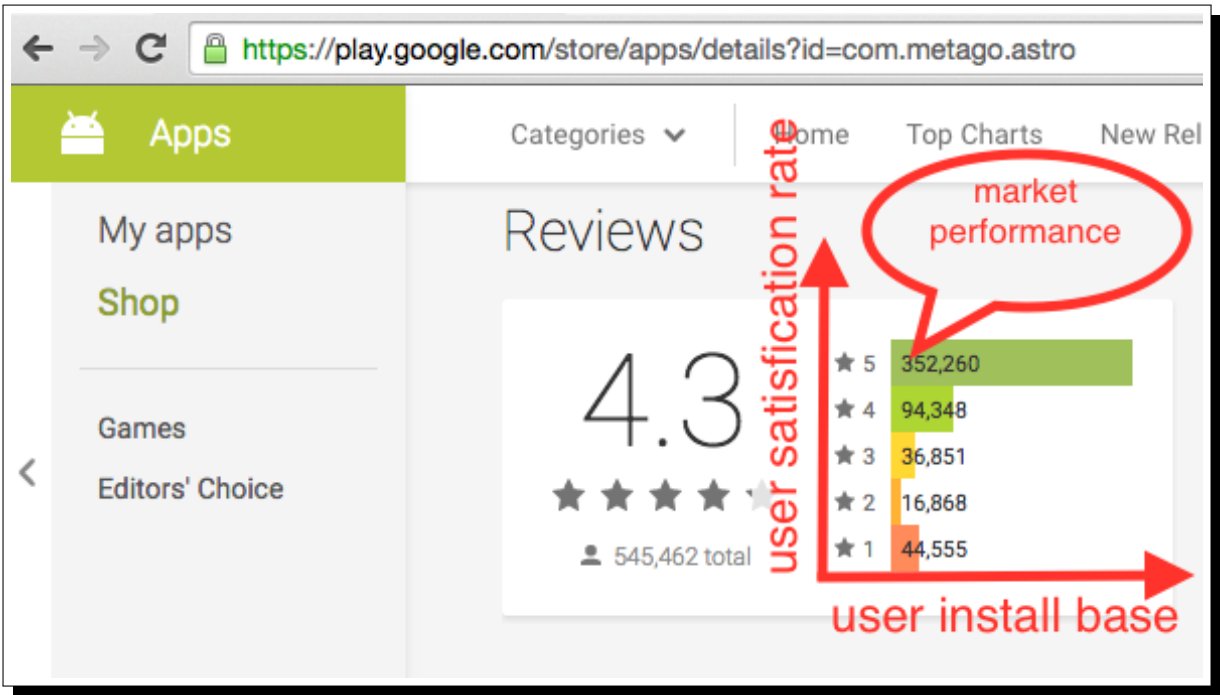


Fig. A.32 Uniqueness

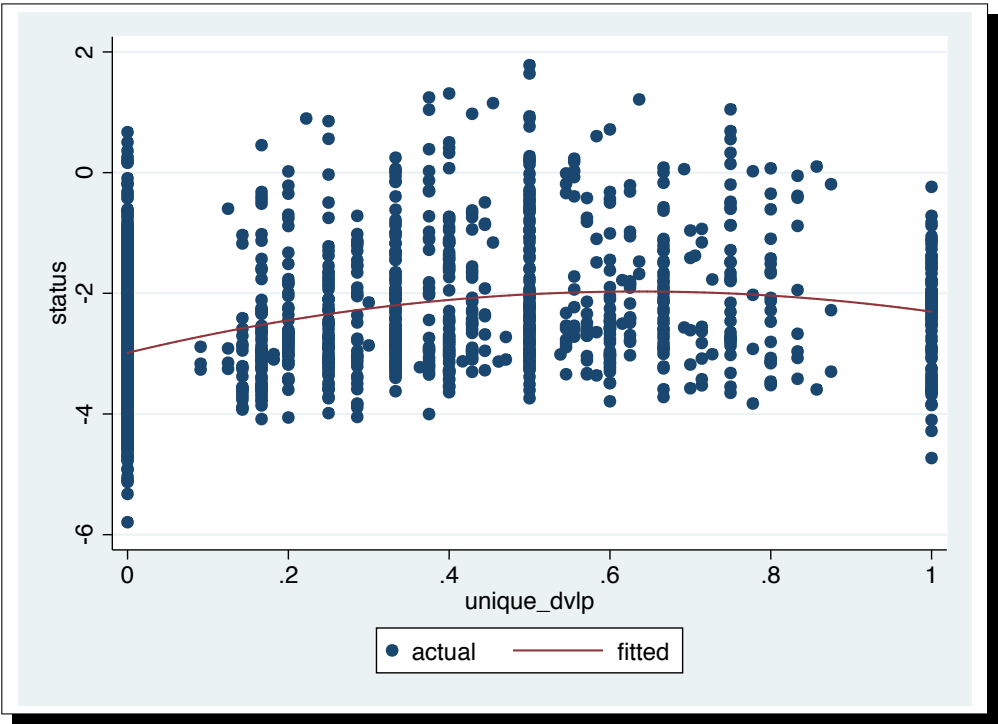


Fig. A.33 Diversification

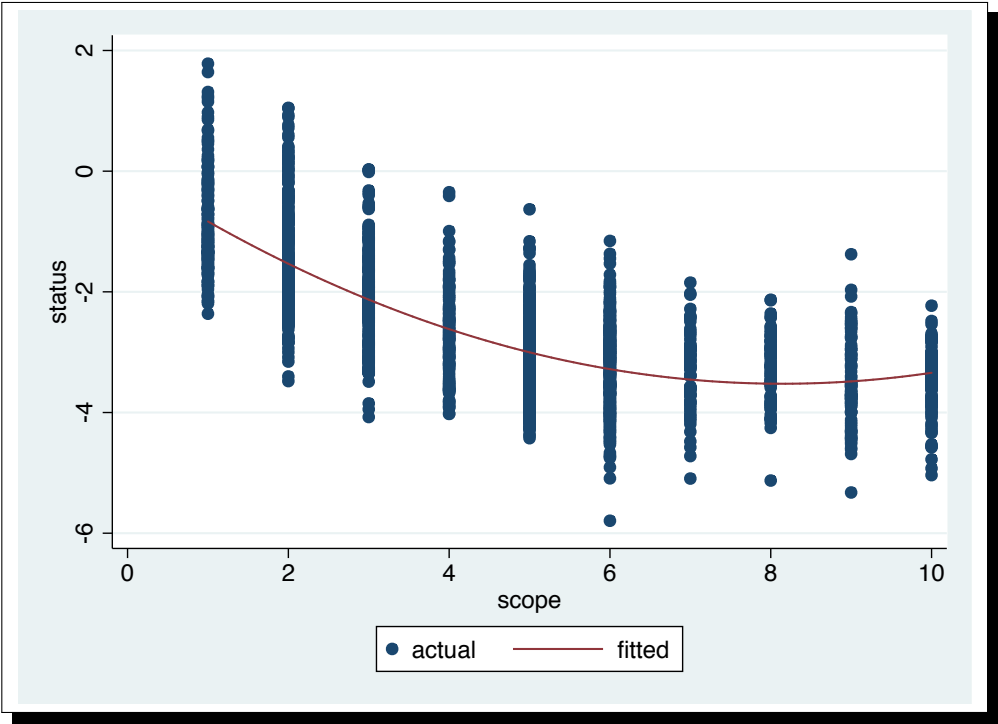


Fig. A.34 Following

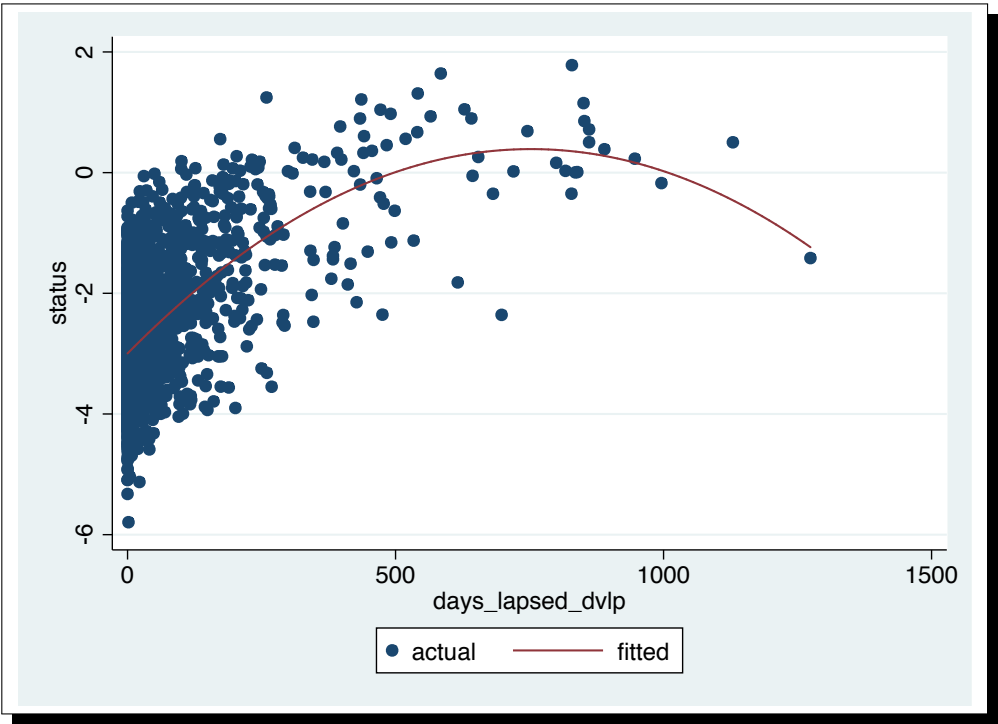


Fig. A.35 Benchmark on following featured vs. non-featured developers

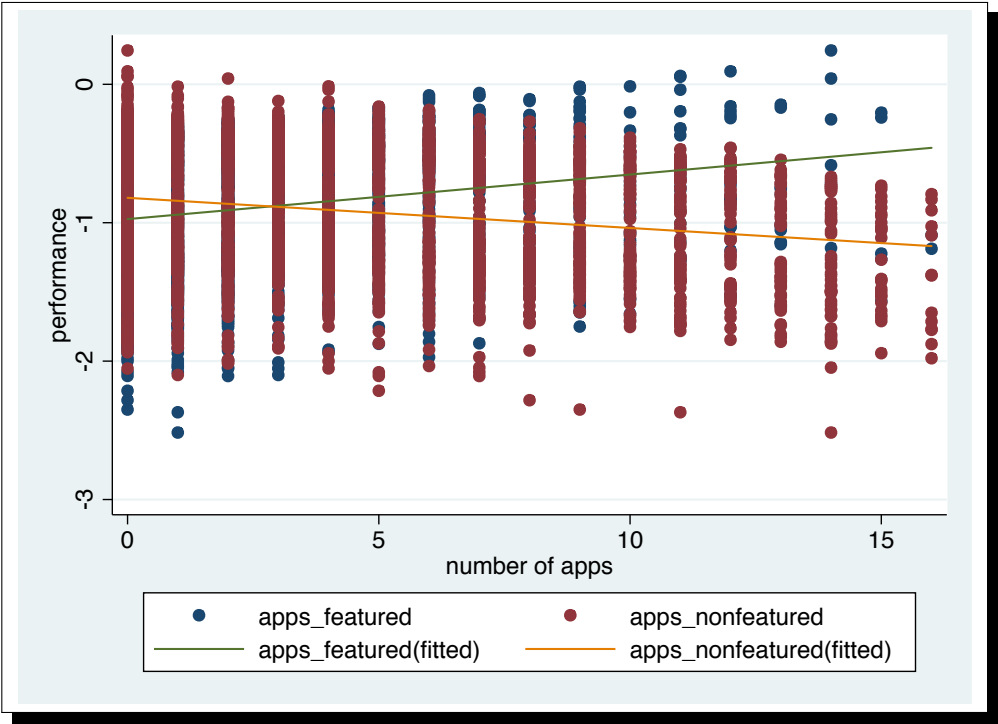


Fig. A.36 Curvilinear relationship to following featured developers

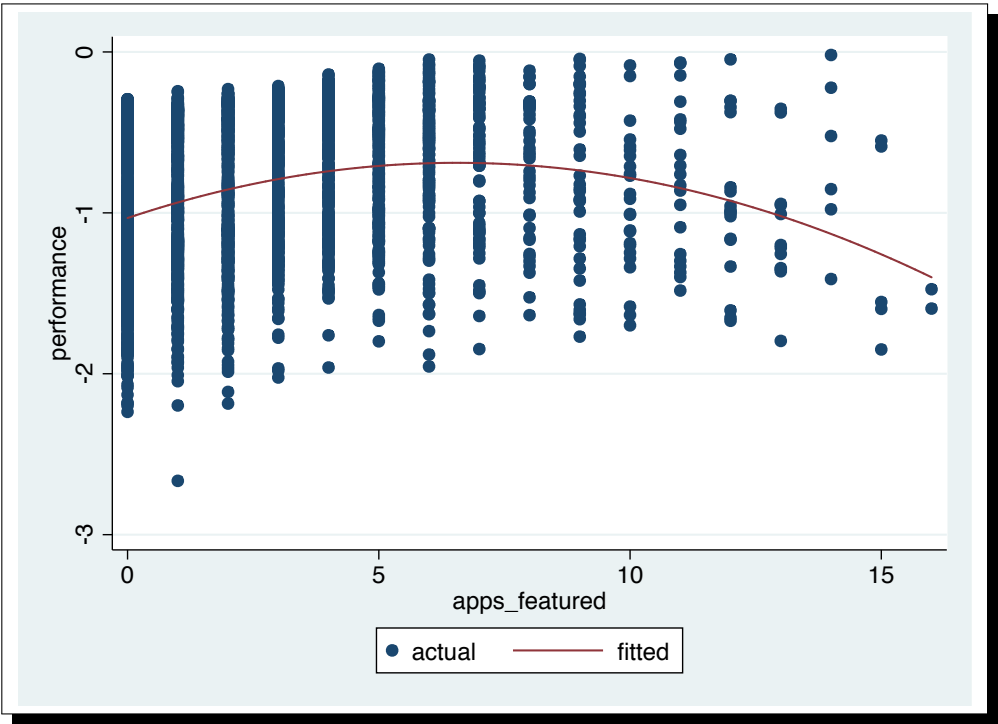


Fig. A.37 Benchmark on different selection status after following non-featured developers

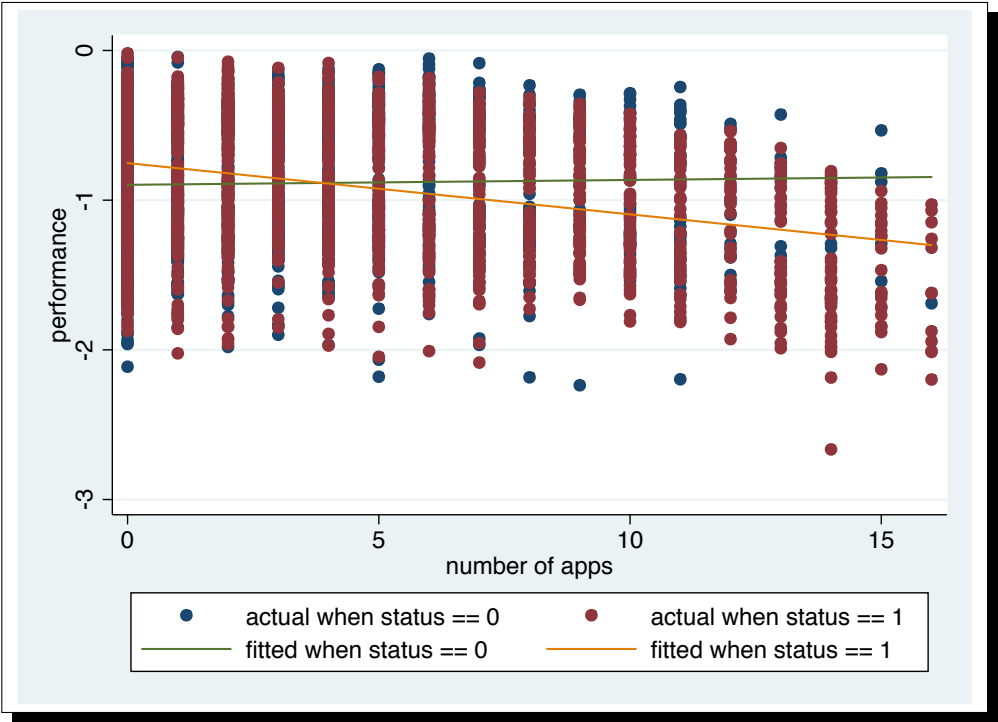
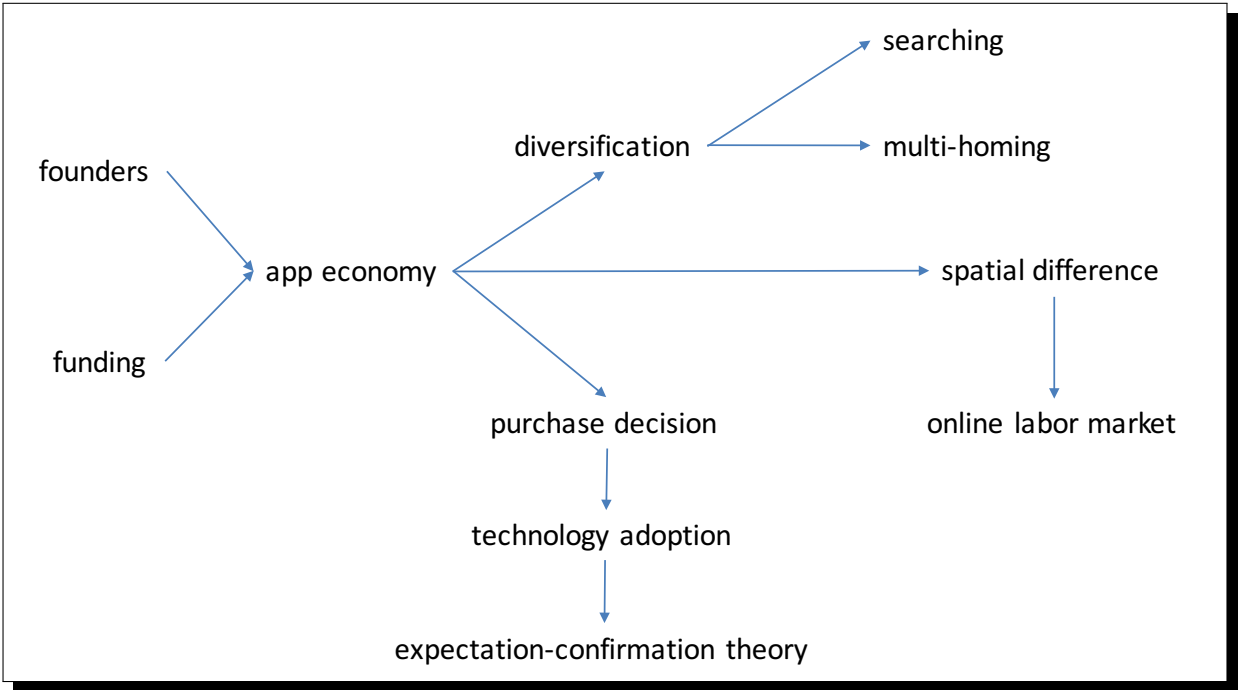


Fig. A.38 Future works around app economy from this thesis



A.6 Terms Used In The Thesis

Android

Android is a broad term referring to anything related to *Android* technology and an market. For example, *Android* device, *Android* OS, *Android* SDK, *Android* Market, *Google Play Store*, etc.

Android device

It is a hardware running *Android* OS. *Android* devices are manufactured by companies, such as HTC, Samsung, Sony, etc. *Android* devices are like PC hardware in desktop computing.

Android OS (Operating System)

It is a software system driving the operation of *Android* devices. It operates *Android* device hardware to function in a proper way. It also provides an infrastructure for other software to communicate or interact with each other. *Android* OS is always technologically complex and hardware dependent, so *Android* SDK is provided to app developers to abstract the functionality of *Android* OS. *Android* app has to build above *Android* OS by using *Android* SDK. *Android* OS is open sourced to the public already, but is mainly maintained by *Google*. *Android* OS is like Microsoft Windows OS or Mac OS or Linux OS in desktop computing.

Android SDK (Software Development Kits)

SDK is a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system or similar development platform. SDK can be considered as an abstract interface to the black box of *Android* OS. With SDK, developers do not need to worry about the underground changes of *Android* OS, but just need to follow the instructions of *Android* SDK. *Android* OS is always updated at the same time as *Android* SDK. Therefore, in most cases, *Android* SDK is equivalent to *Android* OS. *Android* SDK is mainly developed and maintained by *Google*.

Android app

It is the actual product users are normally interacting with directly on the *Android* device. *Android* app is created and maintained by its developer. The developer can be anyone who follows the basic requirements of development. The *Android* app is running above *Android* OS. *Android* devices bought by users are normally comes with some pre-installed *Android* apps. *Android* apps are traded on the market place and can be

installed, updated and uninstalled on *Android* devices running *Android* OS. The *Android* app is like Microsoft Office software which is separates from Microsoft Windows OS in desktop computing.

Google Play Store or Android Market

Google Play Store is the central market place to distribute *Android* apps. Developers publish and sale apps on *Google Play Store* and users buy and download apps from *Google Play Store*. Unlike *Android* OS and *Android* SDK who are open sourced to public, *Google Play Store* is proprietary owned and operated by *Google*. Developers can be featured in *Google Play Store* and users can review or comment the quality of apps to other users. *Google Play Store* keeps those review and featuring history traceable. *Google Play Store* was initialed called *Android Market* before 6th March 2012. Therefore, both *Google Play Store* and *Android Market* refer to the same thing but in different period. To make it simple, this dissertation uses *Google Play Store* rather *Android Market* as much as possible.

Google Play Store Services API

API is short for “Application Program Interface”. It is a proprietary background service by *Google* and API package for *Android* devices. It was initially closed combined in *Android* SDK, but started to be independent from *Android* SDK since 26 September 2012. By using this API, developers can access *Google*’s services. Unlike *Android* OS and *Android* SDK, *Google Play Store Services API* is not open sourced and is fully controlled by *Google*.

developer

Developer is the human force working on *Android* device, *Android* OS, *Android* SDK and *Android* app. However, as this study is focused on *Android* apps, therefore, developer is specifically referring to the creator of *Android* apps in *Google Play Store*.

user

User is the end user of apps in *Google Play Store*.

featured and non-featured

To easier user selection, *Google* would periodically review the quality of apps by developers and then decide whether to feature apps and developers in *Google Play Store*. Featured developers would have a special badge on their apps’ profile, like “Top Developer”, etc. Featured apps would attract great attention from users as *Google* will display them in the “show window” of *Google Play Store*. “Top Developer” is the featuring studied in this dissertation and this is lift-time, meaning that once a developer is featured, all of

his apps will display the special badge. Developers who are featured so far are called non-featured developers. More details can be found on *Google Play Store*'s website ¹.

Android Studio IDE (Integrated Development Environment)

IDE is a software application that provides comprehensive facilities to software development. An IDE normally consists of a source code editor, build automation tools and a debugger. *Android Studio* is an IDE specifically designed for *Android* app development released on 15th May 2013. Before this, a third-party IDE called *eclipse*² was suggested for *Android* app development.

Android platform

It refers to either the *Google Play Store* in a market context or *Android* OS in a technology context.

iOS

iOS is the main competitor and benchmark to *Android* in this study.

OHA (Open Handset Alliance)

The OHA was established on 5th November 2007, led by *Google* with 34 members, including mobile handset makers, application developers, some mobile carriers and chip makers. Its flagship is *Android*. Part of its efforts are to promote a unified *Android* platform. OHA members are contractually forbidden from producing devices that are based on incompatible forks of *Android*. The consortium has expanded into 84 firms to develop open standards for mobile devices. Member firms include *Google*, *HTC*, *Sony*, *Dell*, *Intel*, *Motorola*, *Qualcomm*, *Texas Instruments*, *Samsung Electronics*, *LG Electronics*, *T-Mobile*, *Sprint Corporation*, *Nvidia* and *Wind River Systems*, and etc.

Android market

It is a term which is the same as *Google Play Store*.

market operator

Google

platform owner

Google

good developer

Good developers refers to high quality developers. In this thesis, it specifically refers to developers who are featured by *Google*.

¹<https://developer.android.com/distribute/googleplay/about.html>

²<https://eclipse.org/downloads/>

ordinal or defective developer

Ordinal or defective developers refers to low quality developers. In this thesis, it specifically refers to developers who are not featured by *Google*.